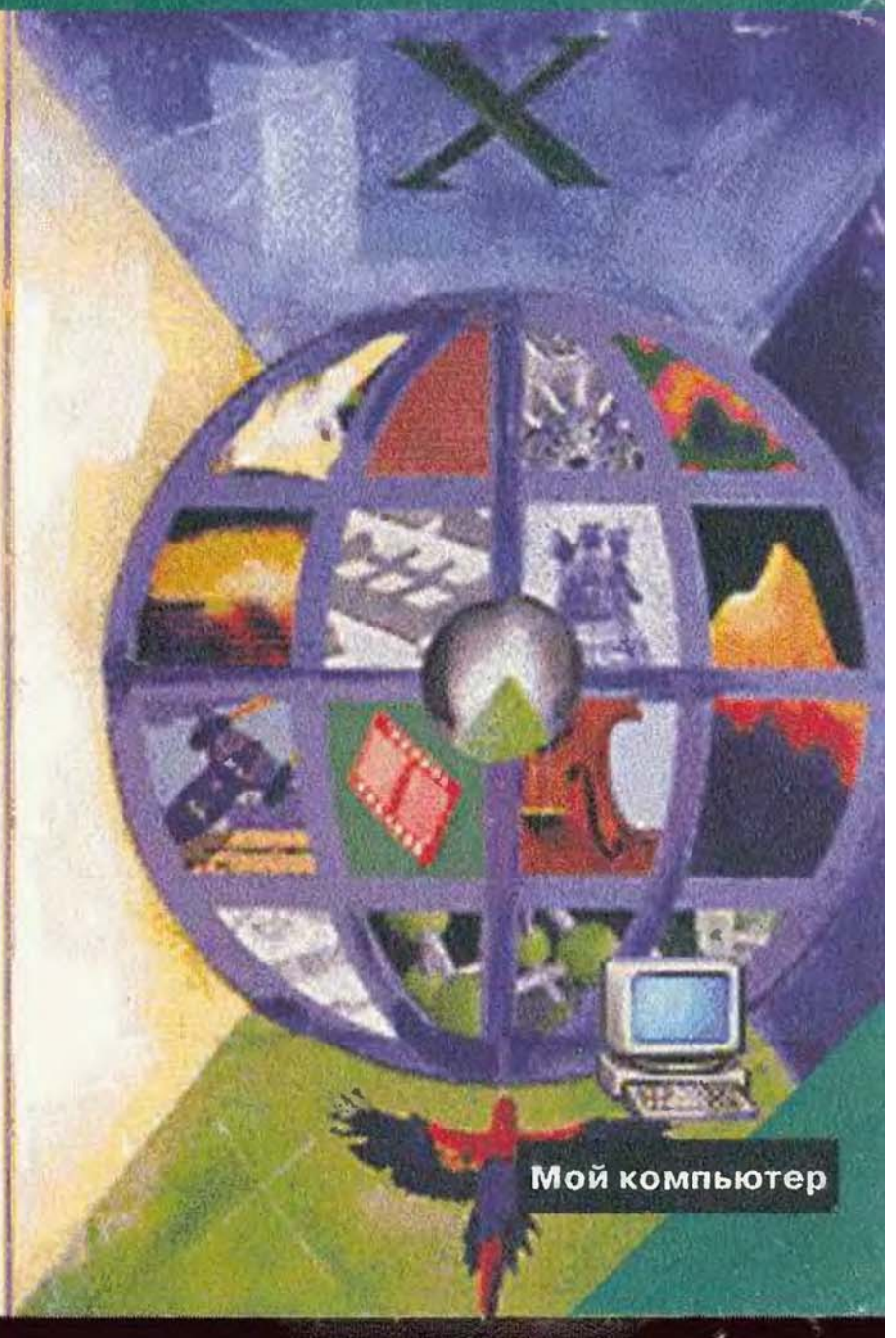
Иван Дегтярев

Язык программирования CLARION 5

Неофициальное руководство пользователя по созданию приложений для Internet

- Описание
- Синтаксис
- Функции
- Clarion Internet Connect
- Хитрости
- IBC Library
- Дизайн web-приложений
- HTML



Серия книг «Мой компьютер»

Иван Дегтярев

Язык программирования Clarion 5.0: Неофициальное руководство пользователя по созданию приложений для Internet

Москва

Майор
Издатель Осипенко А. И.
2002

УДК 004.5
ББК 32.973.26-018.2
Д261

Серия основана в 2001 году Осипенко А. И.

Дегтярев И. А.

Д261 Язык программирования Clarion 5.0: Неофициальное руководство пользователя по созданию приложений для Internet / Иван Дегтярев. - М.: Майор, 2002. - 208 с. - (Серия книг «Мой компьютер»).
ISBN 5-901321-44-8

Введение в программирование на языке Clarion и подробное руководство по созданию web-ориентированных приложений для Internet (Clarion Internet Connect позволяет создать приложение, способное работать в Web с использованием любого обозревателя Internet с поддержкой Java).

УДК 004.5
ББК 32.973.26-018.2

ISBN 5-901321-44-8

Язык программирования Clarion

Универсальный язык

Язык программирования Clarion предназначен для записи высококачественных коммерческих деловых программ применительно к компьютерам.

Язык является привычным, комфортным и имеет свойства предшествующих языков с некоторыми свежими новыми идеями. Синтаксис понятный и современный, привлекающий также для профессионального программирования.

Clarion развивался на предпосылке, что компьютеры необходимы для коммерческих применений. Самое большое применение для мини компьютеров — обычные программы, написанные для (и обычно посредством) компании, которая использует их. Самые обычные коммерческие применения для решения на настольных ЭВМ— это электронные таблицы, текстовые процессоры, управляющие программы для Баз Данных.

Компании хотят обычные программы и компьютеры по низкой цене.

Микрокомпьютеры имеют такую цену и достаточно мощны для решения обычных программ. Причина, по которой так мало деловых приложений для настольных ЭВМ в том, что высококачественные программы трудно писать, а программы, которые легко писать недостаточно хороши.

Язык Clarion может быть расширен операторами и функциями, написанными на других языках программирования. Модули расширения языка (language extension module (LEM)) должны удовлетворять требованиям и быть обработаны специальной утилитой.

Многие библиотеки сегодня могут быть использованы как модули типа LEM.

Формат программы и операторов

Исходный модуль на Clarion есть файл с операторами на языке Clarion. Большие программы обычно содержат несколько модулей, которые можно редактировать, компилировать и составлять. Программы могут быть прерваны перестановкой групп операторов, которые вызывают процедуры и функции, хранящиеся в других модулях.

Процедуры используются для операторов, которые выполняются больше, чем один раз или для операторов, которые имеют простое назначение и могут быть ясно описаны. Имя процедуры становится новым оператором; она выполняется, если ее вызвать по имени. Функции используются для утверждений, которые вырабатывают символьные или числовые значения.

Функция выполняется, если назвать по имени выражение, как бы если она была переменной. Clarion-программа постоянно находится в программном модуле. Процедуры и функции могут также находиться в программном модуле. Первый оператор в программном модуле есть **PROGRAM**. Дополнительные процедуры и функции могут находиться в **MEMBER**-модуле.

MEMBER-модуль - исходный оператор для описания файла **MEMBER**.

Программы, процедуры и функции, содержащие операторы объявления, состоят из последовательности **CODE**-

операторов, выполняемых операторов, местных подпрограмм.

Директивы компилятора, которые дают инструкции компилятору, могут встречаться везде. Некоторые операторы составные, поэтому они содержат другие операторы. Составные операторы и операторы, которые они окружают, называются структурами. Описательные предложения определяют константы, переменные, экраны, файлы, отчеты и табличную память. Выполняемые операторы представляют программные действия, операции, указанные в описательных операторах.

Местные подпрограммы как процедуры, но они не имеют параметров.

Местные подпрограммы не содержат описательных операторов и не могут вызываться вне программ, процедур или функций.

Формат предложений

Предложения содержат ключевые слова, операторы, метки, опознавательные знаки и специальные символы (иногда называемые пунктуацией). Ключевые слова являются словарем языка. Оператор метки идентифицирует положение его относительно других операторов; опознавательные знаки используются для шаблонов и контроля печати, и специальных символов, изображающих связь других компонентов предложения.

Программа содержит необязательный параметр **MAP**, в котором объявлены модули, процедуры и функции, используемые программой. Данные, объявленные в программе, являются глобальными, поэтому могут быть использованы всеми процедурами и функциями. Данные, объявленные в процедуре или функции не могут использоваться вне этой процедуры или функции.

MEMBER-модуль, который использует глобальные данные, должен назвать программный модуль, который содержит глобальные данные.

Формат операторов

Clarion-оператор не должен начинаться в первой позиции строки, которая резервируется для операторов меток. Предложения ограничиваются концом строки или «;». Если одна строка содержит более чем одно предложение, предложения должны отделяться «;». Последнее предложение ограничивается концом строки. Операторы за точкой с запятой игнорируются.

Длинные предложения могут занимать более, чем одну строку; нужно ограничивать каждую продолжающуюся строку вертикальной чертой.

Продолжаемые предложения могут быть ограничены после оператора метки, инициатора, терминатора, закрывающего ограничителя или коннектора (исключая символ подчеркивания, который встречается внутри предложения метки). Продолжаемое предложение не может быть ограничено внутри символьной строки. Длинная символьная строка должна быть разорвана на более короткие знаками конкатенации и продолжена перед или за оператор конкатенации.

Операторы структур (составные операторы и предложения, которые они окружают, ограничиваются точкой или **END** предложением). Для лучшей читабельности, предложения внутри структуры могут быть отделены.

Комментарии обозначаются восклицательным знаком и могут появиться в любой строке. Комментарии могут содержать какие-либо символы, они игнорируются компилятором и ограничиваются концом строки. Пустые строки могут быть вставлены везде и также игнорируются.

Оператор МЕТКА

Оператор метка может находиться в программе, процедуре или функции. Метка в описательном предложении называется именем данных. Имя данных может быть переменной, процедурой, экраном. Метка выполняемого оператора используется как указатель в GOTO-операторе.

Оператор метка начинается в колонке 1 и отсылает к следующему оператору той же самой строки или к следующему оператору в программе, процедуре или функции. Операторы внутри оператора метки не могут начинаться в столбце 1.

Операторы метки содержат до 12 символов, состоящих из букв от A до Z, цифр от 0 до 9 и символ подчеркивания. Первым символом оператора-метки должна быть буква. Операторы метки являются нечувствительными в данном случае. Например, имя данных может быть объявлено как **Printline**, затем упоминается как **PRINTLINE** или **Printline**. Предложение (оператор) метки не может быть резервируемым словом.

Резервируемые (служебные) слова

Служебные слова составляют основу языка для создания внутренних форм синтаксиса и структур. Резервные слова испортят Компилятор, если они будут использоваться в операторах-метках.

Здесь список резервных слов Clariona:

```
AND
DO
EXECUTIVE
INCLUDE
OMIT
RETURN
TIMES
```

```
BREAK
EJECT
EXIT
LOOP
OR
ROUTINE
TO
BY
ELSE
FUNCTION
MEMBER
OROF
SIZE
UNTIL
CASE
ELSIF
GOTO
NOT
PROCEDURE
SOURCE
WHILE
CYCLE
END
IF
OF
PROGRAM
THEN
XOR
```

Заметим, что наибольшее число ключевых слов Clarion, особенно используемых в предложениях объявления, не могут быть резервируемыми и могут быть использованы в операторах-метках.

Специальные символы

Специальные символы инициации, ограничения, разделения и соединительные ключевые слова, операторы-метки и опознавательные знаки служат для оформления предложений. Язык Clarion пользуется для этого специальными символами.

Initiators

! инициирует комментарий

? инициирует метку приравнивания поля

@ инициирует опознавательный знак

Terminators

; ограничивает предложения

CR ограничивает предложение (возврат каретки)

- ограничивает структуру

| ограничивает продолжаемую строку

ограничивает длинное неявное имя переменной

$ ограничивает числовое неявное имя переменной

" ограничивает символьное имя переменной

Delimitors

() включает параметр или параметр списка

[] включает индекс или список индексов

" включает символьную константу

{} включает повторение счета в символьной константе

<> заключает код ASCII в символьной константе

Connectors

. соединяет целое и дробь числовой константы

, соединяет параметры в списке

: соединяет приставку и имя данных

- соединяет мнемонику в операторе метки

Операторы, которые также являются соединительными

+ оператор сложения

- оператор вычитания или унитарный минус

* оператор умножения

/ оператор деления

% оператор коэффициентов

. экспоненциальный оператор

< меньше чем

> больше чем

= оператор равенства или оператор присваивания

<> не равно

& оператор конкатенации

Компилирование Clarion-программы

Clarion-компилятор читает исходный модуль и продуцирует файл процессора. Файл процессора содержит псевдокоды Clariona, которые представляют исходный модуль в компактной форме. Clarion-псевдокод — машинный язык для «воображаемого» Clarion-компьютера, который был предназначен для Clariona. Этот «воображаемый» компьютер использует операционные коды и операнды, что соответствует операторам Clariona.

Процессор-утилита выполняет (интерпретирует) файлы процессора.

Транслятор-утилита читает файлы процессора и создает стандартные объектные файлы DOS (с расширением .OBJ). Эти объектные файлы затем связываются (используя DOS LINK или другие компоновщики программы) с Clarion-библиотекой, чтобы произвести DOS выполняемые файлы (с расширением .EXE). Clarion-библиотека также используется для создания утилит Процессора, гарантируя, что программа, выполняемая Процессором, работает также, как оттранслированная программа в DOS.

Компилятор создает файл Процессора для каждого модуля в программе. Процессор собирает все процессорные файлы, чтобы выполнить программу. Транслятор продуцирует объектный файл для каждого процессорного файла в программе. Объектные файлы затем все объединяются вместе в Clarion-библиотеке, чтобы создать выполняемый файл.

Компилятор, Процессор и Транслятор создают группу файлов, которые используют имя программы или имена файлов с различными расширениями.

Когда компилируется модуль, Компилятор загружает программный символический файл и переводит в адреса глобальных данных. Эта техника разрешает Процессору выполнить программу немедленно после того, как модуль был откомпилирован. Поэтому, если глобальные данные в программном модуле изменились, рекомендуется исправить все его глобальные адреса.

Для того, чтобы выполнить повторно компиляцию, Компилятор предлагает «поток компиляции» для компиляции программного модуля. В этом случае поток компиляции может быть сделан быстро, потому что компилятор и программный символический файл уже находятся в памяти. Поток компиляции продуцирует суммарный файл ошибок (с расширением .SUM), который представляет текстовый файл, содержащий ошибки компиляции, найденные в каждом модуле.

Оператор PROGRAM

Оператор **PROGRAM** должен быть первым в программном модуле. Оператору **PROGRAM** может предшествовать **TITLE** или **SUBTITLE**.

Метка оператора **PROGRAM** должна быть допустимым оператором метки. Первые восемь символов метки используются как имя файла для символического файла (.SYM) и файла процессора (.PRO), создаваемых Компилятором. Если метка пропускается, исходное имя файла используется для этих файлов. Поэтому, если метка опускается, исходное имя файла должно приспосабливаться к правилам допустимых операторов метки.

Программный модуль — это исходный файл с операторами, состоящий из программы, оптимальной **MAP**-структуры и оптимальных процедур и функций. Объявления в программе общие для всех процедур и функций в программном модуле и членах модуля. Программа с членами, процедурами или функциями должна содержать **MAP**-структуру.

Если глобальные данные в программном модуле изменились, все его члены-модули должны быть откомпилированы заново. Компилятор предлагает оптимальный «поток компиляции» для компиляции программы с модулями.

Оператор MEMBER

MEMBER-оператор должен быть первым оператором в **MEMBER**-модуле. Тем не менее, **MEMBER**-оператору могут предшествовать **TITLE** или **SUBTITLE** директивы компилятора.

Параметром **MEMBER**-оператора является имя программы, к которой принадлежат **MEMBER**-модули. Компилятор использует имя программы с расширением .SYM как файл спецификаций для программного символического файла.

Программный символический файл содержит адреса оператора метки, который Компилятор использует для разрешения глобальных ссылок в **MEMBER**-модуле.

Если параметр в **MEMBER**-операторе опускается, **MEMBER**-модуль становится «универсальным **MEMBER**-модулем», который можно включать в любую программу.

Файл процессора продуцируется для универсального **MEMBER**-модуля только в потоке компиляции.

MEMBER-модуль является исходным файлом операторов, содержащим процедуры и функции для программы. Для того, чтобы быть включенным в программу, **MEMBER**-модуль должен быть назван в операторе **MODULE** в **MAP**-структуре.

Структура MAP

MAP — структура предусматривает размещение программы или модуля. Она включает **MAP**-оператор и операторы, следующие за ним, ограниченные точкой или **END**-оператором. **MAP**-структура используется в программном модуле для названия модулей, глобальных процедур и функций.

MAP-структура не может находиться в **MEMBER**-модуле. **MAP**-структура может содержать **PROC**, **FUNC**, **MODULE**, **AREA** и **OVERLAY** операторы.

Оператор PROC

PROC(имя процедуры)

PROC операторы называют по имени процедуры программы. Параметры имени процедуры должны быть оператором метки оператора **PROCEDURE**. Процедуры являются глобальными и могут вызываться из программ, любых функций и любых процедур.

PROC-оператор, который вызывает процедуру, принадлежащую программному модулю, должен появиться в структуре, впереди первого **MODULE**-оператора.

PROC-оператор, который вызывает процедуру, принадлежащую **MEMBER**-модулю, должен появиться в **MODULE**-структуре, вызывающей этот **MEMBER**-модуль. Процедура не может быть вызвана, если она не объявлена в **PROC**-операторе.

Оператор FUNC

FUNC(имя функции), тип данных

FUNC операторы называют функции программы. Параметр имени функции должен быть оператором метки в операторе **FUNCTION**. Атрибут «тип данных» поставляет тип данных возвращаемой величине. Параметр «тип данных» должен быть **LONG**, **REAL** или **STRING**.

Функции являются глобальными и могут вызываться из программы, какой-либо процедуры или какой-либо функции. Оператор **FUNC**, который называет функцию, принадлежащую **MEMBER**-модулю должен появиться в **MODULE** структуре, вызывающей этот **MEMBER**-модуль. Функция не может быть вызвана, если она не объявлена в **FUNC**-операторе.

Структура MODULE

MODULE(имя модуля)

MODULE-структура вызывает **MEMBER**-модуль с его процедурами и функциями. Она включает оператор **MODULE** и предложения, следующие за ним до тех пор, пока не будет ограничения в виде (.) или **END**-оператора.

Параметр имени модуля должен быть символьной константой, определяемый для имени файла (без расширения) исходного модуля. Компилятор использует параметр имени модуля с расширением .CLA для всех модулей для

потока компиляции. Компилятор использует параметр имени модуля с расширением .PRO для загрузки модулей для выполнения.

Структура **MODULE** может встречаться только внутри **MAP**-структуры.

Структуры **MODULE** содержат **PROC** и **FUNC** операторы, называющие процедуры и функции, располагающиеся в модуле. Чтобы быть загруженным Процессором, модуль должен быть назван в программе в **MAP**-структуре.

Компилятор помещает программный модуль после обработки его процессором в той же самой директории, как его исходный файл. Компилятор помещает **MEMBER**-модуль после обработки его процессором в той же самой директории, как его программный модуль. Когда процессор загружает .PRO и .LEMS, они все должны быть в той же самой директории, как и программный модуль.

Атрибут BINARY

Атрибут **BINARY** идентифицирует модуль как изображение его в памяти, записанное на языке ассемблера или другом компилирующем языке.

Изображение модуля в памяти есть готовый для загрузки, настраиваемый двоичный модуль, продукция **EXE2BIN** DOS-команды.

Процессор использует параметр имени модуля оператора **MODULE** с расширением .BIN, чтобы загрузить в память представление модуля.

Структура AREA

AREA-структура определяет пространство оверлея. Оно включает оператор **AREA** и операторы, следующие за ним до того, как они не будут ограничены **END**-оператором или точкой. Оператор **AREA** не имеет параметров или атрибутов.

Структура **AREA** содержит **OVERLAY**-структуры, которые содержат **PROC** и **FUNC** операторы.

Размер пространства оверлея, определяемый **AREA**-структурой так велик, как самый большой оверлей в **AREA**-структуре.

Площадь оверлея — пространство в памяти, где оверлеи загружаются. Оверлей содержит один или более модулей и объявляется в **OVERLAY**-структуре. Только один оверлей может быть резидентным (загруженным в памяти) в площади оверлея в одно время. Когда оверлей-процедура или функция (процедура или функция, вызываемая посредством **PROC** или **FUNC** оператора в **OVERLAY**-структуре) вызывается, оверлей загружается в память (если она еще не резидентна).

Оверлей-процедура или функция не может вызывать процедуру или функцию из другого оверлея, который использует ту же самую площадь оверлея, или данных, находящихся вне различных оверлеев в одном и том же пространстве.

Структура OVERLAY

OVERLAY-структура определяет оверлей. Она включает оператор **OVERLAY** и операторы, следующие за ним до оператора **END** или (.). Оператор **OVERLAY** не имеет параметров или атрибутов.

Структура **OVERLAY** содержит один или более структур **MODULE**, каждая из которых содержит один или более операторов **PROC** или **FUNC**. Если оверлей-процедура или оверлей-функция вызывается, полный оверлей, в котором она находится, загружается в память (если ее еще там нет). Оверлей, который загружается, содержит все процедуры и функции в каждом модуле, названном в оверлее (во всяком случае они называются в **PROC** или **FUNC**-операторах). Когда оверлей загружен, его переменные получают их на-

чальные значения. Представления модулей в памяти (объявленных как **MODULE**-операторы с атрибутом **BINARY**) не могут подвергаться процессу оверлея.

Оператор PROCEDURE

Метка PROCEDURE(список параметров)

Оператор **PROCEDURE** — первый оператор процедуры.

Оператор **МЕТКА** — имя процедуры и должен следовать правилам операторов меток. Процедура замещает библиотеку процедур с тем же именем, генерирующим сообщение об ошибках.

Список параметров представляет ряд переменных, отделенных запятыми и заключенных в круглые скобки. Имена переменных в списке параметров должны быть объявлены внутри процедуры.

Процедура выполняется, если ее вызвать со списком параметров.

Оператор FUNCTION

Метка FUNCTION(список параметров)

Оператор **FUNCTION** является первым оператором функции.

Оператор **МЕТКА** является именем функции и должен следовать правилам меток оператора. Функция замещает функцию в библиотеке с таким же именем, если генерируется сообщение об ошибке.

Список параметров представляет набор переменных, отделенных запятыми и заключенных в круглые скобки. Имена переменных в списке должны быть объявлены внутри функции. Функции выполняются, если вызвать их со списками параметров в выражении. Функция затем возвращает одно значение в выражение. Тип данных возвращаемой величины — **LONG**, **REAL** или **STRING** как данные

оператора **FUNC** в **MAP**-структуре. Функция не может быть выполнена, если ее имя не появится как параметр в **FUNC**-операторе. Выполнение начинается в первом операторе после **CODE** оператора.

Параметры операторов PROCEDURE и FUNCTION

Операторы **PROCEDURE** и **FUNCTION** сопровождаются списком параметров, состоящих из имен переменных, отделенных запятыми и заключенных в круглые скобки. Каждое имя переменной в списке параметров оператора **PROCEDURE** или **FUNCTION** должно быть объявлено в процедуре или функции. Параметры, объявленные с оператором **EXTERNAL** вызываются внешними параметрами. Другие параметры вызываются локальными параметрами.

Локальные параметры

Локальные параметры являются «проходящими мимо величины», поэтому они копируются с переменной или выражением в ту же самую позицию списка параметров вызываемой подпрограммы. Поэтому, когда местные параметры изменяются, их вызывающая копия не меняется.

Локальный параметр не нуждается в подходящем типе данных его вызывающей копии.

Внешние параметры

Внешние параметры являются «проходящими мимо адресов» и локальными или глобальными к программе, вызывающей процедуре или функции.

Изменение внешнего параметра меняет его вызывающую копию, так как есть только одна копия параметра. Функции или выражения в списке параметров вызывающей подпрограммы являются промежуточными величинами.

Внешние промежуточные величины могут быть изменяемы, но изменения могут не чувствоваться.

Метки **RECORD**, **HEADER**, **FOOTER**, **DETAIL** и **TABLE** структур могут быть использованы в списке параметров вызывающей программы, но они трактуются как структура **GROUP**, вызываемая процедурой или функцией.

Метки **SCREEN**, **REPORT**, **DOS**, **FILE**, **KEY**, **INDEX**, **TABLE**, используются в списке параметров вызывающей программы, но должен быть описаны как атрибут в **EXTERNAL**-операторе, таком как **EXTERNAL**, **FILE** или **EXTERNAL**, **KEY**. Параметр **TABLE** трактуется как **GROUP**, если его соответствующий оператор **EXTERNAL** не имеет атрибутов. Иначе, с **TABLE**-атрибутом, он трактуется как **TABLE**. **EXTERNAL**-атрибут, который не подходит по типу данных его вызывающей копии, произведет остановку во время прогона программы.

Оператор CODE

Оператор **CODE**-директива Компилятора, который отделяет операторы объявления от выполняемых операторов внутри программы, процедуры или функции. Операторы, предшествующие **CODE**-оператору, являются операторами объявления и называются «data section». Операторы, следующие за **CODE**-оператором, являются выполняемыми операторами и называются «code section». Оператор, следующий за **CODE**-оператором является первым выполняемым оператором, когда программа, процедура или функция вызывается.

Оператор ROUTINE

Оператор **ROUTINE** — первый оператор местной (локальной) программы. Местные программы должны располагаться в конце «Code section» программы, процедуры или функции. Оператор **МЕТКА** называет локальную программу

и он используется с **DO** оператором для выполнения программы.

Локальные программы ограничиваются концом исходного файла операторов или словами **ROUTINE**, **PROCEDURE** или **FUNCTION**. Оператор **EXIT** может использоваться внутри местной программы для возвращения управления оператору, следующему за оператором **DO**. EXIT-оператор — необязательный, так как программы автоматически заканчиваются после последнего предложения.

Оператор END

END-оператор дает ограничение для операторов структур.

Директивы компилятора

Директива TITLE

TITLE('заголовок модуля')

Заголовок печатается в 1-м столбце 1-й строки листинга Clarion-компилятора. **TITLE**-директива дает этот заголовок, но директива сама не печатается в распечатке. **TITLE** должен стоять в начале исходного файла впереди **SOURCE**, **PROGRAM** или **MEMBER** операторов.

Заголовок остается на том же самом месте каждой страницы распечатки. **TITLE**-директивы впереди **SOURCE**-оператора во включаемых файлах (называемых **INCLUDE**-операторами) игнорируются.

Директива SUBTITLE

SUBTITLE ('подзаголовок')

Подзаголовок печатается в первом столбце 3-й строки листинга Clarion-компилятора. Директива компилятора **SUBTITLE** дает этот заголовок, но сама не печатает в листинге. **SUBTITLE** должна стоять в начале исходного файла

впереди **SOURCE**, **PROGRAM** или **MEMBER**-оператора. Подзаголовок может быть установлен **SELECT** оператором, который начинает новую страницу в распечатке. **SUBTITLE** директивы впереди **SOURCE** оператора во включенных файлах (называемых **INCLUDE**-операторами) игнорируются.

Директива EJECT

Форма 1: EJECT('подзаголовок')
Форма 2: EJECT

EJECT-директива начинает новую страницу в Clarion-распечатке.

Форма 1 EJECT устанавливает подзаголовок и начинает новую страницу в распечатке. **Форма 2** начинает новую страницу с тем же самым подзаголовком, как на прошлой странице. **EJECT** директива сама не печатает в распечатке.

Директива OMIT

OMIT('ограничитель')

OMIT-директива специфицирует блок исходных строк, чтобы освободить их от компиляции. Эти исходные строки могут содержать комментарии или программные операторы, которые не компилируются. Блок **OMIT** идет от строки **OMIT** до строки, инициируемой «terminator»-ом (ограничителем). OMIT не может использовать оператор метки и должен быть первым оператором строки. В **OMIT**-блок включается ограничивающая строка.

Директива SOURCE

Директива **SOURCE** — первый оператор исходного файла, который включается в программу или **MEMBER**-модуль (используя оператор **INCLUDE**).

Тем не менее, **SOURCE**-директиве могут предшествовать **TITLE** или **SUBTITLE** операторы. Компиляция исход-

ного файла, представленная SOURCE-директивой, выдает листинг компиляции, но не файл процессора.

Директива INCLUDE

Форма 1: INCLUDE ('имя файла')
Форма 2: INCLUDE ('имя файла', имя структуры)

INCLUDE-директива специфицирует исходный файл операторов, который включается в программный модуль или член модуля. Параметр имени файла — символьная константа, специфицирующая имя файла с необязательным путем и расширением. Если путь пропускается (не указан), используется путь текущего директория. Если расширение пропускается, .CLA поддерживается. SOURCE директивы во включаемом файле игнорируются. TITLE и SUBTITLE директивы впереди SOURCE-директивы в подключаемом файле игнорируются. В **Форме 2**, параметр имени структуры обеспечивает включение оператора метки из исходного файла операторов. Только структура из оператора метки ограничивающего периода будет включаться в модуль.

Атрибут LIST

Атрибут LIST директивы INCLUDE просит Компилятор составить список исходных операторов так, как они встречаются. Если LIST-атрибут пропускается, включаемые исходные операторы не появляются в распечатке.

Объявление переменных

Операторы объявления используются для определения переменных. Переменные представляют ячейки памяти, которые содержат числовые или символьные величины. Оператор, объявляющий переменную, поддерживает её имя, формат, длину, начальное значение.

Оператор, объявляющий переменную, состоит из оператора метки (имя переменной), оператора ключевого сло-

ва (тип данных) с необязательным параметром и необязательный атрибут ключевые слова с необязательными параметрами. Все описательные операторы должны предшествовать CODE-оператору программы, процедуры или функции.

Обозначения шаблона

Знаки шаблона представляют формат, который используется, когда числовая величина пересылается в шаблон строки. Если формат слишком мал, чтобы содержать числовую величину, шаблон строки заполняется звездочками. Все знаки шаблона начинаются со знака («@»). Размещение «B» в последней позиции шаблона означает «пробел, когда нуль». Поэтому «нуль» продуцирует символьный шаблон со всеми нулями.

Знаки шаблонов используются как параметры **STRING**, **ENTRY** и **MENU** операторов и как параметры других процедур и функций. Есть шесть типов знаков шаблона: цифровой, научное представление, дата, время, форматирования и символьные.

Числовой и денежный шаблоны

Знак цифрового шаблона может производить широкое множество фиксированно-десятичных форматов. Целая часть числа может быть сгруппирована в тысячи запятыми, точками, пробелами и не сгруппирована совсем. Целая и дробная часть числа могут быть разделены точкой (десятичной точкой), запятой или пробелом. Неиспользуемый высший разряд числа может быть заполнен пробелами, нулями или звездочками.

Отрицательные числа могут быть сформатированы с ведущим знаком минус или замыкающим знаком минус или он может быть заключен в скобки. Знак доллара или другого денежного символа может находиться перед или за чис-

лом. И, наконец, число может быть отформатировано как состоящее из одних пробелов, когда его значение есть нуль.

Знак числового шаблона — @N, сопровождаемый указателями для размера, места, знака и денежного обращения.

Размер указателя указывает длину (в символах) конечной — 23 строки шаблона и формат целой части числа. Длина должна быть целой в интервале от 1 до 255.

Отсутствие формата размера отделяет запятыми группы в целой части между тысячами. Размещение подчеркивания слева от длины подавляет группирование и заполняет бесполезные высшие позиции пробелами. Размещение нуля слева от длины подавляет группирование и заполняет бесполезные высшие позиции (разряды) нулями.

Размещение звездочки слева от длины заполняет бесполезные высшие разряды звездочками.

Примеры:

```
@N9  4,550,000
@N_9  4550000
@N09  004550000
@N*09  **4550000
```

Если группировка не подавляется (с подчеркиванием или нулем слева), символ группировки (отличный от запятой) может быть специфицирован справа от длины. Подчеркивание группируется пробелами, а точка группирует с точками.

Примеры:

```
@N9_  4  550  000
@N9.  4.550.000
```

Указатель «позиции» специфицирует десятичные позиции числа и символ, используемый для отделения целой и дробной частей.

Количество десятичной позиции должно быть меньше, чем длина, указанная в размере указателя и в любом случае не должна быть больше 15.

Точка слева от позиций специфицирует десятичную точку как разделитель и знак ударения (') специфицирует запятую. Точка устанавливает запятую, знак ударения устанавливает точку как отсутствие группирующего символа для целой части.

Примеры:

```
@N_9.2  4550.75
@N_9'2  4550,75
@N9'.2  4,550.75
@N9.2  4,550.75 (такой как выше)
@N9.'2  4.550,75
@N9'2  4.550,75 (такой как выше)
@N9_'2  4550.75
```

Указатель знака специфицирует формат отрицательного числа. Знак минус слева от размера указателя продуцирует ведущий знак минус. Знак минус справа от указателя продуцирует замыкающий знак минус. Заключение размера и позиций указателя в круглые скобки продуцирует заключенное в круглые скобки отрицательное число.

Примеры:

```
@N-9.2  -4,550.75
@N9.2-  4,550.75@N(10.2)  (4,550.75)
```

Указатель денег — это или знак доллара ($), или если символьная строка заключается в тильдах (~).

Указатель денег должен быть также первым или последним указателем (исключая для «B» указателя — пробел, когда ноль) в цифровом шаблоне.

Если денежный указатель первый, денежный символ «плывет» к левой цифре высшего порядка числа.

Примеры:

```
@N$9.2  $4,550,75
@N~DM~10.2  DM4.500,75
@N11.2~KR~  4.500,75  KR
@N$(11.2)  $(4,550.75)
```

Примеры знаков шаблонов, используемых во всем мире для денежного обращения.

```
США @N$12.2  $24,500.00
Франция @N~F~12_'2  F  24.500,00
Италия @N~Lit.~15'2  Lit.540.000,00
Объед.Кор-во @N~&~12'2  &  24,500.00
Норвегия @N~KR~12'2  KR  24.500,00
Германия @N~DM~12'2  DM  24.500,00
Финляндия @N~MK~12'_2  MK  24.500,00
Швеция @N~KR~19'2  KR  24.500,00
```

Научное представление шаблонов

Научная система счисления является техникой для выражения очень больших или очень малых чисел.

Формат для научной системы счисления есть **@ Em.n**, где **m** — общее число символов в конечном шаблоне строки, и **n** — количество слева от десятичной точки.

«B» как последний символ шаблона означает «пробел когда ноль».

Примеры:

```
@  E9.0  1,967,  865  .20e  +  007
@  E12.1  1,967,  865  1.9679e  +  006
@  E12.1  -1,967,  865  -1.9679e  +  006
@  E12.1  000000032  3.2000e  -  008
@  E12.1B  0
```

Шаблоны дат

Даты могут быть представлены в виде числовой переменной (обычно **LONG**) или как строка шаблона. Дата,

представленная как числовая переменная, называется стандартной датой. Когда числовая переменная пересылается в шаблон строки даты, полагают, что стандартная дата форматируется в соответствии с его шаблоном. Когда строка шаблона даты пересылается в числовую переменную, она деформатируется в соответствии с его шаблоном и преобразуется в стандартную дату.

Знак шаблона даты состоит из символов **@D**, следуемых с кодом формата даты, следуемых с необязательным указателем разделителя.

Формат кодов даты:

1. mm/dd/yy 3/16/88
2. mm/dd/yyyy 3/16/1988
3. *mmm dd,yyyy MAR 16,1988
4. *mmmm dd,yyyy MARCH 16,1988
5. dd/mm/yy 16/03/88
6. dd/mm/yyyyy 16/03/1988
7. *dd mmm yy 16 MAR 88
8. *dd mmm yyyy 16 MAR 1988
9. yy/mm/dd 88/03/16
10. yyyy/mm/dd 1988/03/16
11. yymmdd 880316- 25
12. yyyymmdd 19880316

** Форматы этих дат не могут быть использованы как параметры ASK или ENTRY операторов. (Нечисловые символы не могут входить в числовые поля).*

Указатель разделителя может использоваться для спецификации символа, иного, чем (/) как разделитель для форматов кодов числовых дат. Числовые формы 11 и 12 не имеют разделителей.

Указатели разделителя:

. продуцирует точку

- продуцирует тире

_ продуцирует пробелы

' продуцирует запятые

Примеры шаблонов дат 16 марта 2002 г.:

```
@D1  3/16/02  @D1.  3.16.02
@D2  3/16/2002  @D2-  3-16-2002
@D3  MAR  16,2002  @D1_  3  16  02
@D4  MARCH  16,2002  @D2,  3,16,2002
@D5  16/03/02  @D5.  16.03.02
@D6  16/03/2002  @D6-  16-03-2002
@D7  16  MAR  02  @D5_  16  03  02
@D8  16  MAR  2002  @D6_  16  03  2002
@D9  02/03/16  @D9.  02.03.16
@D10  2002/03/16  @D9-  02-03-16
@D11  020316
@D12  20020316
```

Шаблоны времени

Время может быть помещено в числовую переменную (обычно **LONG**) или шаблон строки. Время, помещаемое в числовую переменную, называется стандартным временем и представляет сотую секунду с полуночи. Когда числовая переменная помещается в шаблон строки времени, начинается отсчет стандартного времени и оно форматируется в соответствии с его шаблоном. Когда шаблон строки помещается в числовую переменную, она деформатируется в соответствии с его шаблоном и преобразуется в стандартное время.

Шаблон времени состоит из символов @T, следуемых с кодом формата времени, следуемых с необязательным указателем разделителя.

Коды формата времени:

1. `hh:mm  14:30`
2. `hh  mm  1430`
3. `*hh:mmxM  2:30PM`

4. hh:mm:ss 14:30:00

Этот формат времени не может быть использован как параметр **ASK** или **ENTRY** операторов. (Нечисловые ключи не могут входить в числовые поля).

Указатели разделителя:

. продуцирует точки

- продуцирует тире

_ продуцирует пробелы

' продуцирует запятые

Примеры знаков шаблонов времени, используемых время 2:15PM:

```
@T1  14:15 @T1.  14.15
@T2  1415 @T1_  1415
@T3  2:15PM @T3.  2.15PM
@T4  14:15:30 @T4.  14.15.30
```

Форматированные шаблоны

Отформатированный шаблон называется шаблоном для целых чисел «сделай сам». Знак шаблона для форматированного шаблона содержит символы **@P**.

Формат шаблона есть символьная строка, которая содержит < или # в позициях, где расположены целые.

Символ, отличный от < или # рассматривается как редактирующий символ и появляется в конечной строке шаблона. < означает «печатать в высших разрядах пробелы вместо нулей» и # означает «печатать в высших разрядах нули как нули». Заглавная **P** может быть включена в шаблон формата с использованием разделителей нижнего регистра (для примера, **@p<#:##PMp**). Также строчная **p** может быть включена в шаблон с использованием разделителей нижнего регистра (пример **@P << # pounds P**).

«B» как последний символ знака шаблона означает «пробел когда ноль».

Примеры отформатированных шаблонов:

```
@P<#/##/##P  20,987  2/09/87
@P<#/##/##P  120,987  12/09/87
@P<#:##  AMP  146  1:46  AM
@P<#:##  PMp  1,146  11  :  46  PM
@P###-##-####  P  246,732,453  246-73-2453
@P###/###-####  P  3,059,417,665  305/941-7665
@P###/###-####  P  9,417,665  000/941-7665
@  P4##A-#P  112  411A-2
@  P#'  <#"P  511  5'  11"
@  P<#  1b  <#  ozP  902  9  1b  2oz
```

Символьные шаблоны

Шаблон для строки обозначается как **@Sn**, где **n** определяет количество символов в строке. Шаблоны строки используются при обработке экранов с **ENTRY** и **MENU** полями операторов и при доступе к монитору и клавиатуре с **ASK**, **LOOK** и **SHOW** операторами.

Оператор EXTERNAL

Оператор **EXTERNAL** специфицирует параметр процедуры или функции, которые «проходят мимо адресов».

Параметр, который проходит мимо адреса, называется внешним параметром и принадлежит вызывающей программе, процедуре или функции.

Метка **EXTERNAL** — оператора должна появиться в списке параметров его операторов **PROCEDURE** или **FUNCTION**. Программа не может содержать **EXTERNAL**-операторы до тех пор, пока она не имеет параметров. В вызываемом операторе, параметр может быть именем переменной или промежуточной величиной (результат выражения или функции). Если вызываемый параметр - метка **RECORD**, **HEADER**, **FOOTER**, **DETAIL** или **TABLE**-структура, он трактуется как **GROUP**-структура. Изменение величины

внешней переменной меняет ее величину в вызываемой программе, процедуре или функции.

Промежуточные величины, которые возвращаются функциям, являются результатом выражения и временными. Когда промежуточная величина проходит через **EXTERNAL**, она работает как переменная в течение всей процедуры. Если внешняя переменная меняется, ее значение в промежуточной памяти изменяется, как значение переменной, но когда процедура заканчивается, промежуточная величина освобождается. Поэтому, изменения, сделанные во внешних промежуточнцх величинах, не обнаруживаются вызывающей программой или процедурой.

Атрибуты внешних допустимых массивов экранов, отчетов, файлов, ключей и таблиц проходят как параметры процедур и функций. Следующие ключевые слова могут быть используемы как атрибуты **EXTERNAL**-оператора:

```
DIM
FILE
KEY
SCREEN
DOS
INDE
REPORT
TABLE
```

EXTERNAL-атрибут допускает метку **EXTERNAL**-оператора для использования как ограниченного параметра.

Хотя эти атрибуты **EXTERNAL** удобны для обобщенных процедур работы с файлами, у них есть также недостатки. Если тип даты вызываемого параметра не подходит соответствующему **EXTERNAL**-атрибуту, программа останавливается с ошибкой.

Структура GROUP

GROUP-оператор позволяет многократно объявленным операторам ссылаться на простое имя переменной. **GROUP**-оператор и операторы, следующие за ним, называются **GROUP**-структурой. **GROUP**-структура ограничена точкой или **END**-оператором. **GROUP**-структуры могут использоваться, чтобы проставить размеры в ряде объявлений или сравнить составные переменные. Группы также помогают организовать сложные программы сохранением связанных данных вместе. **CLEAR** процедура может использовать нуль или пробел в объявлениях в **GROUP**-структуре. **GROUP**-переменные трактуются как строки в выражениях.

Структуры **GROUP** могут быть сгруппированы (структуры внутри структур). Из-за их внутреннего формата **SHORT**, **LONG** и **REAL** переменные не сопоставимы обычно, когда сравниваются как строки. Поэтому **GROUP**-структуры, содержащие эти типы данных не будут сравниваться или использоваться как ключи в файлах или табличной памяти. **DIM**, **OVER** и **PRE**-атрибуты могут использоваться с **GROUP**.

Неявные переменные

Неявные переменные не требуют предложений объявления. Clarion обеспечивает 3 типа неявных переменных (**LONG**, **REAL** и **STRING(32)**), которые создаются Компилятором в их первой ссылке и инициализируются в соответствии с их типом. Знак фунта в последней позиции переменной идентифицирует переменную как **LONG**; знак доллара обозначает **REAL**; знак цитаты обозначает 32-символьную строку **STRING**.

LONG и **REAL** неявные переменные инициализируются как нуль, а неявная строка инициализируется пробелами.

Имя неявной переменной, включая символ ограничения, не может превышать 12 символов.

Атрибуты переменных

Атрибут PRE

PRE атрибут обеспечивает приставку, которая может быть использована для различия идентичных имен переменных, которые встречаются в различных структурах. **PRE** может быть использована со следующими операторами:

```
DETAIL
FOOTER
MENU
REPORT
DOS
GROUP
RECORD
SCREEN
FILE
HEADER
REPEAT
TABLE
```

Параметр-префикс содержит до трех символов, содержащих буквы от A до Z, числа от 0 до 9 и подчеркивающий символ. Префикс должен начинаться с буквы. При использовании, префикс соединяется с именем переменной двоеточием. На переменную в префиксной структуре можно ссылаться без префикса, если имя переменной уникально.

Атрибут OVER

OVER атрибут позволяет ту же самую область памяти использовать более, чем в одном операторе объявления. **OVER** может быть использовано со следующими операторами, исключая **SCREEN** или **REPORT** структуры:

```
BYTE
GROUP
```

```
MEMO
SHORT
DECIMAL
HEADER
REAL
STRING
DETAIL
LONG
RECORD
TABLE
FOOTER
```

Местные переменные не могут быть объявлены через глобальные переменные, и переменные в **GROUP**-структуре не могут быть объявлены через переменные вне **GROUP**-структуры.

Переменные не могут быть объявлены через короткие переменные. Также, переменная в **OVER**-атрибуте не может иметь начальное значение.

Атрибут DIM

```
DIM (размерность,...,размерность)
```

DIM-атрибут объявляет массив. **DIM** может использоваться со следующими операторами, исключая **SCREEN** или **REPORT**-структуры:

```
BYTE
LONG
DECIMAL
GROUP
SHORT
REAL
STRING
EXTERNAL
```

DIM(10)-атрибут в предложении описания обозначает повторение этого определения 10 раз (но только в одном списке). Параметр «размерность» — числовая константа, ко-

торая специфицирует число повторений в этой размерности. Индексы внутри групп присоединяются к имени переменной, показывая, какой элемент повторения (массива) используется.

Так **X[i]** указывает i-ое повторение в X-массиве. **X[1]** указывает 1-й элемент и **X[10]** указывает 10-й элемент одномерного массива.

Массивы с многократной размерностью называются «вложенными». В двумерном массиве ряды расположены в памяти в следующем порядке:

```
[1,1], [1,2]... [1,m],
[2,1], [2,2], [2,m],
...
[n,1], [n,2]... [n,m].
```

DIM-атрибут может иметь до 4 размерностей. Размерность **GROUP** структуры — особый случай. Каждый уровень вложенности прибавляет индексы (слева направо) заново объявленным **GROUP** и его членам. Вложение трех одномерных массивов продуцирует переменную, нуждающуюся в трех индексах. Одного измерения массив внутри двумерного массива требует трех индексов. Количество размерностей **GROUP**-структуры описывается точно также как его **GROUP**-оператор. Сумма индексов не может превышать четырех.

Данные массива не могут превышать 65520 байтов. Индексы могут быть константами, переменными, функциями и выражениями.

Массивы как параметры процедур и функций

Массив не может быть объявлен параметром **PROCEDURE** или **FUNCTION**-оператором до тех пор, пока он является «внешним параметром массива».

DIM атрибут используется с оператором **EXTERNAL** для объявления внешнего параметра массива. Каждый необходимый размер массива должен быть специфицирован в **DIM**-атрибуте, но параметр «размер» не нуждается ни в чем другом, кроме 1, например: **label EXTERNAL, DIM(1,1)** объявляет двумерный внешний массив.

Параметры «размер» не диктуют объявление размера, так как массив «внешний» или другими словами, «описан везде».

Метка массива (сама) не может быть использована в выполняемых операторах. Компилятор надеется, что индексы будут специфицированы. К массиву можно обращаться из процедуры, идя к нему с внешним параметром размерности. Массив проходит с внешним параметром размерности, присоединяя пустые группы ([]) к имени массива.

Директивы компилятора

Директива EQUATE

Форма 1: Метка EQUATE(метка)
Форма 2: Метка EQUATE(константа)
Форма 3: Метка EQUATE(шаблон)
Форма 4: Метка EQUATE(ключ)

Директива **EQUATE** назначает метку другой метке, числовому или символьному значению, обозначению шаблона или символьного слова. Метка **EQUATE**-директивы не может быть резервным словом или ключевым словом, используемым в **DATA**-секции. Метка **EQUATE**-директивы не может быть такой как ее параметр.

В **форме 1 EQUATE** параметр (метка) может быть любым допустимым оператором метки. Эта форма **EQUATE** может быть использована для объявления заместителя метки (чаще называемого алиас).

В **форме 2** параметр (константа) может быть любой числовой или символьной константой. Эта форма может быть использована для объявления метки как сокращенной записи для **long** констант (таких как **Pi**) или сделать константу легче для определения и изменения.

В **форме 3** параметр (шаблон) должен быть знаком шаблона. Эта форма может быть использована для объявления метки как стенографии для обозначения шаблона (например, **SS_PIC** может быть использована для (**@P ### — ##- ## ##P**).

В **форме 4** параметр «**Ключ**» должен быть ключевым словом Clarion, отличным от **EQUATE**, **TITLE** или **SUBTITLE**. Эта форма **EQUATE** может быть использована для модификации Clarion-языка.

Директива SIZE

Форма 1: SIZE(метка)
Форма 2: SIZE(константа)
Форма 3: SIZE(шаблон)

SIZE-директива задает размер памяти, используемый параметром.

Компилятор возвращает **SIZE**-директиву с ее соответствующей величиной перед компиляцией оператора. **SIZE** может использоваться в программе, где имеется допустимое числовое значение. В **форме 1** параметр (метка) должен быть меткой переменной, объявленной предварительно. Размер переменной - в байтах.

В **форме 2** параметр (константа) может быть числовой или символьной константой. Размер символьной константы есть длина строки. Размер числовой константы есть размер самого маленького типа данных (используя **BYTE**, **SHORT**, **LONG** или **REAL**), который может его содержать.

В **форме 3** параметр (шаблон) есть допустимое значение шаблона.

Размер значения шаблона есть длина отформатированной строки, продуцируемой значением шаблона.

Выражения

Выражение — это формула, по которой вычисляется значение в языке Clarion. Выражение может появиться как источник присваивания оператора, параметра процедуры или функции, индекса размерности величины или как часть **IF**, **CASE**, **EXECUTE** или **LOOP** предложений.

Выражения содержат константы, переменные и функции, соединенные операторами. Выражение вычисляется выполнением его операторов в порядке приоритета. Каждый оператор продуцирует промежуточное значение, которое затем используется в последующих операторах. Порядок приоритета операторов ограничивается оператором типа и круглыми скобками (которые группируют выражения в подвыражения). Конечная промежуточная величина есть величина выражения. Числовое, символьное и условное выражение продуцирует числовую величину, символьную величину и логическую величину («правда» или «ложь») соответственно.

Простая константа, переменная или функция также составляют выражение. Константа, переменная или функция может быть использованы в любом контексте, который разрешает выражение.

Функции

Функция - это подпрограмма, которая возвращает промежуточное значение выражению. Функция используется в том же самом контексте, где и переменная; тем не менее, функция должна следовать со списком параметров. Функция, у которой нет параметров, следует с пустым списком параметров, например: **ERROR()**.

Есть два рода функций. Пользовательские функции начинаются оператором **FUNCTION** и появляются как параметр **FUNC** оператора в **MAP**-структуре. Атрибут **FUNC** оператора (**LONG**, **REAL** или **STRING**) дает тип данных возвращаемой величины.

Библиотечные функции являются встроенными в язык Clarion и не объявляются в **MAP**-структуре.

Список библиотечных функций:

```
ABS
BSHIFT
CONTENTS
FOREHUE
LOGE
RANDOM
SUB
ACOS
BXOR
COS
FORMAT
LOWER
RECORDS
TAN
AGE
BYTES
DATE
INRANGE
MAXIMUM
REFER
TODAY
ALL
CENTER
DAY
INSTRING
MEMORY
```

RIGHT
UPPER
ASIN
CHOICE
DEFORMAT
INT
MONTH
ROUND
VAL
ATAN
CHR
EOF
KEYBOARD
NAME
ROW
YEAR
BACKHUE
CLIP
ERROR
KEYCODE
NUMERIC
ROWS
BAND
CLOCK
ERRORCODE
LEFT
OMITTED
RUNCODE
BOF
COL
FIELD
LEN
PATH
SIN
BOR

```
COLS
FIELDS
LOG10
POINTER
SQRT
```

Числовые константы

Числовые константы являются фиксированными величинами, которые входят в предложения объявления атрибутов, и в выражения.

Например, в выражении 1+1, 1 — числовая константа. Десятичная числовая константа содержит необязательный знак «минус» впереди в целой части и необязательную десятичную точку в дробной. Форматировать символы, такие как запятые и знак доллара не разрешается в числовых константах, а числовая константа не может заканчиваться десятичной точкой.

В добавление к десятичным (основание 10), числовые константы могут быть двоичными (основание 2), восьмеричными (основание 8) или шестнадцатеричными (основание 16).

Двоичные константы содержат впереди необязательный знак минус, 32 цифры 0 и 1, и ограничиваются B или b.

Восьмеричные константы содержат впереди необязательный знак минус, цифры от 0 до 7, и ограничены 0.

Шестнадцатеричные константы содержат впереди необязательный знак минус, цифры от 0 до 9 и буквы от A до F (которые обозначают числа от 10 до 15), и ограничиваются H или h. Самый левый символ шестнадцатеричной константы не должен быть A или F.

Очень большие или очень маленькие константы могут выражаться в научной нотации (степени десяти). Научная нотация использует формат:

```
- #. ##### + ###
```

где — #.##### — знаковая десятичная величина; + ### — знаковая степень 10.

Цифровые константы никогда не хранятся как строки.

Числовые операторы

Числовой оператор комбинирует два арифметических операнда для создания промежуточной величины. Числовыми операторами являются:

+ сложение (A+B значит A plus B)

- вычитание (A—B значит A minus B)

* умножение (A*B значит A times B)

/ деление (A/B значит A divided B)

^ возведение в степень (A^B значит A in a degree B)

% деление модулей (A%B значит the remainder of A divided by B).

Числовая операция продуцирует как **LONG**, так и **REAL** промежуточную величину, зависящую от типов данных операндов. Деление или возведение в степень, однако, всегда продуцирует **REAL** промежуточную величину.

Операторы деления и экспоненты всегда продуцируют **REAL**.

Числовые выражения

Числовые выражения могут занимать как параметры в **CASE**, **EXECUTE**, **IF** и **LOOP** операторах, так и в операто-

рах присваивания. Цифровые выражения могут содержать оператор конкатенации, но не могут содержать логические операторы. Если оператор конкатенации есть конечный в числовом выражении, или элемент числового выражения есть символьная константа или переменная, конечная промежуточная величина преобразуется из символьной в числовую.

Символьные константы

Символьные константы располагаются в символах, изображающих слова, предложения, адреса, что входят как параметры в предложения объявления и атрибуты, и в выражения. Символьные константы заключаются в апострофы. Используют два последовательных апострофа для включения апострофа в строковую константу.

Непечатаемые символы могут быть включены в строковую константу, заключая их ASCII коды в угловые скобки (знаки < и >). Напомним, что двоичные, восьмеричные или шестнадцатеричные константы могут быть использованы внутри угловых скобок, где их представление проще. Используйте два знака (<<) для включения знака (<) в строковую константу.

Нет необходимости дублировать знак (>) в строковую константу, так как непарный знак (<) инициирует поиск знака (>). Имеется формат для включения непечатаемых символов в строковую константу: **<xxx,yyy,zzz>**, где **xxx**, **yyy** и **zzz** равны ASCII символьным кодам.

Строковые константы, содержащие последовательности простых символов, могут быть укорочены с использованием счетчика повторов. Счетчик повторов специфицирует, сколько раз символ может быть повторен в строковой константе. Счетчик повторов состоит из символа, который по-

вторяется, левых фигурных скобок, счетчика повторов и правых фигурных скобок.

Счетчик повторов — общее число последовательных повторов символа.

Также как апостроф и знак (<), используют две левых фигурных скобки для включения левой скобки в строковую константу. Строковая константа (два апострофа) определяет нулевую строку, которая не содержит символов. После того, как короткая строка расширится пробелами, строковая константа может быть использована для представления пробельной строки любой длины.

Вычисление числовых и символьных выражений

Числовое или символьное выражение вычисляется выполнением каждого оператора. Последовательность операторов управляется правилами предшествования операторов. Круглые скобки используются в выражении для группировки операций в подвыражения.

Подвыражения выполняются из внутренних к внешним скобкам. Операции во внутренних (из-за отсутствия скобок) скобках выполняются слева направо внутри их предшествующего уровня. Поэтому все операторы из предшествующего уровня выполняются слева направо, потом все операторы из второго уровня выполняются слева направо и так далее.

Здесь предшествующие уровни числовых и символьных выражений:

```
Уровень 1 ( ) заключенная в скобки группа
Уровень 2 - одноместный синус
Уровень 3 вызов функции параметры группы, которые
являются подвыражением
Уровень 4 ^ экспонента
```

```
Уровень 5 * / % умножение, деление, деление по
модулю
Уровень 6 + - сложение, вычитание
Уровень 7 & конкатенация
```

Операторы конкатенации

Оператор конкатенации размещает две строки в одной. Длиной результирующей строки является сумма длин двух операндов. Оператор строки всегда продуцирует строку с промежуточной величиной; тем не менее, один или оба операнда могут быть числовым типом данных. Если операнд имеет числовой тип данных, он преобразуется в дробь с необязательным знаком минус, представляющую строку цифр с необязательной десятичной точкой.

Следующий оператор иллюстрирует самую большую ошибку в использовании оператора конкатенации:

```
BOOK_TITLE = BOOK_TITLE & 'CCP_OUT'
```

Полагаем, что **BOOK_TITLE ---> string(40)**. Оператор конкатенации продуцирует строку в 48 символов (длина **BOOK_TITLE** + длина **'CCP_OUT'**). Когда оператор присваивания переносит 48 символов промежуточной величины к **BOOK_TITLE**, он усекает последние восемь символов, испортив оператор конкатенации. Правильное использование:

```
BOOK_TITLE = SUB(BOOK_TITLE,1,SIZE(BOOK_TITLE)-8) &
'CCP_OUT'
```

Символьные выражения

Символьные выражения встречаются как параметры и как источник присвоения операторов, когда адресат является символьной переменной.

Символьные выражения могут содержать числовые операторы, но они не могут содержать какие-либо логические операторы. Если числовой оператор является конечной

операцией, вычисляемой в символьном выражении, или если только элемент строкового выражения является цифровой константой или переменной, конечная промежуточная величина преобразуется из числовой в символьную величину.

Логические операторы

Логический оператор проверяет два операнда и продуцирует условия «true» или «false». Есть два вида логических операторов: условные и связи. Условные (< чем, > чем, = и их комбинации) сравнивают две величины. Операторы связи (И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ и НЕ) вырабатывают условия «false» или «true».

Когда величина используется как операнд связи, логическое выражение вычисляется как «true» для ненулевых и «false» для нуля. (Например, **0 AND 1** вырабатывает «false»).

Логическими операторами являются:

Условные:

= ! правда, если равно

< ! правда, если меньше чем

> ! правда, если больше чем

<> или = или **NOT=** ! правда, если не равно

~< или => или >= или **NOT<** ! правда, если не меньше чем

~> или =< или <= или **NOT>** ! правда, если не больше чем

Связи:

~ или **NOT** ! правда, если ложь; ложь, если правда

AND ! правда, если в обоих случаях правда

OR ! правда, если в одном из случаев или в обоих правда

XOR ! правда, если в одном из случаев, но не в обоих правда

Логические выражения

Логические выражения используются в **IF**, **LOOP**, **WHILE** и **LOOP UNTIL** операторах. Конечное условие выражения («правда» или «ложь») используется для ограничения выполнения последовательности операторов, которые следуют.

Условные операторы имеют самое высокое предшествование в логических выражениях. Все условные операторы имеют одинаковое предшествование и вычисляются слева направо. Предшествование операторов связи ограничивает группирование условных подвыражений. Неоднозначные составные выражения могут стать яснее с использованием круглых скобок.

Предшествование в логических выражениях следующее:

Уровень 1 — условные

Уровень 2 — **NOT** или ~

Уровень 3 — **AND**

Уровень 4 — **OR** или **XOR**

Постоянные выражения

Выражение «константа» вычисляется Компилятором и хранится как константа; поэтому выражение константа может использоваться везде, где допустима константа. Выражение «константа» формируется с использованием только констант и операторов. Выражение строки константы со-

держит строковую константу, строки равенств и оператора конкатенации (&).

Выражение цифровой константы состоит из цифровых констант и числовых операторов.

Выражение «константа» может содержать скобки, но первый символ выражения константы не может быть левой скобкой.

Функции в выражениях

В выражении функция вызывается как одноместный оператор. Параметры вычисляются слева направо, каждый в соответствии с правилами для числовых или символьных выражений.

В логическом выражении все условия проверяются перед концом работы. Это указывает на то, что когда логическое выражение содержит несколько вызовов функций, все вызовы функций выполняются. Функции, которые не только возвращают величины, могут быть написаны; например. они могут видоизменить содержание глобальной переменной. Если эта глобальная переменная появляется неожиданно в выражении, она может влиять на результат выражения неожиданным образом. Эти действия вызываются функциями в выражениях и называются «побочным эффектом».

Например:

```
A = X()+Y(B)
```

В вышеприведенном примере полагаем, что **B** — глобальная переменная и что функция **X** изменила величину **B**.

Можно получить различный результат, если выражение запишется так:

```
A = Y(B)+X()
```

Операторы присваивания

Операторы присваивания задают ячейке памяти значение, или в терминологии языка программирования, пересылают значение к переменной. Оператор присваивания, однако не просто пересылает содержимое одной ячейки памяти к другой ячейке. Формат переменной зависит от типа ее данных.

Значение одного типа данных должно быть преобразовано для того, чтобы переслать его к переменной с другим типом данных. В CLARION любую переменную можно переслать к любой другой переменной. Для любой комбинации типа данных существует правило преобразования данных.

Оператор присваивания обычно самый простой оператор в программе. В данном операторе применяется знак равенства, обозначающий «устанавливается равным». Таким образом, оператор **A=B** означает «A устанавливается равным B», или проще, «переслать B к A».

Получатель должен быть именем переменной, а источник должен быть выражением. Так как выражение должно быть выполнено до того, как его значение можно было переслать, оператор присваивания является первичным вычислительным инструментом программы. Например, оператор A=B+C устанавливает A равным сумме B плюс C.

Если источник и получатель являются различными типами данных (или оба являются шаблонными строками), оператор присваивания производит преобразование данных.

Операторы присваивания и действия

```
Метка получатель += источник
Метка получатель -= источник
Метка получатель *= источник
Метка получатель /= источник
```

```
Метка получатель ^= источник
Метка получатель %= источник
```

Различные операторы присваивания производят арифметические действия с переменными. В каждом из изображенных операторов приемник должен быть именем переменной, а источник должен быть выражением.

Наиболее часто оператор присваивания применяется для прибавления или вычитания переменной. Следующие операторы прибавляют единицу или вычитают единицу из переменной A:

```
A = A + 1
A = A - 1
```

Предлагается сокращенная форма оператора присваивания для выполнения подобных действий. Эти операторы также добавляют единицу к, или вычитают единицу из переменной A:

```
A += 1
A -= 1
```

Операторы означают «прибавь 1 к A» и «вычти 1 из A». Вот примеры всех оперативных операторов присваивания, показанные с их эквивалентной линейной формой:

```
A += 1 A = A + 1
A -= B A = A - B
A *= -5 A = A * 5
A /= 100 A = A / 100
A ^= 1+1 A = A ^ (1+1)
A %= 7 A = A % 7
```

Оператор CLEAR

Оператор **CLEAR** устанавливает в исходное состояние переменную, структуры **GROUP**, **RECORD**, **HEADER**, **DETAIL**, **FOOTER** или массив. Для численной переменной (**BYTE**, **SHORT**, **LONG**, **REAL** и **DECIMAL**) исходным

состоянием является нуль, а для **STRING** или **MEMO** — пробелы.

Структура устанавливается в исходное состояние путем установления в исходное состояние каждой из своих переменных. Весь массив устанавливается в исходное состояние путем применения имени массива с пустыми скобками, **CLEAR** устанавливает в исходное состояние только массив целых чисел. **CLEAR** не может применяться для установления в исходное состояние одного элемента массива.

Правила преобразования данных

Оператор присваивания не всегда дает требуемый результат.

Рассмотрим оператор: **I = 1.1**. Если **I** описывается как **LONG**, оператор устанавливает значение **I** равным **1**, а не **1.1**, как требовалось.

Переменные типа **LONG** не допускают выражения над дробными числами, следовательно, **LONG = REAL** вызывает усечение дробной части **REAL**.

Ниже показаны правила преобразования для каждой комбинации типов данных, которые возникают в случае несовпадения типов.

Комбинации, которые не требуют преобразования данных (например, **BYTE** к **BYTE**, или **SHORT** к **LONG**) всегда создают равные источник и получатель. Комбинации с многократными правилами преобразования, (например, **REAL** к **SHORT**, или **STRING** к **DECIMAL**), требуют более одного шага для преобразования.

Правило 1: SHORT или LONG к BYTE

Знаковый разряд источника игнорируется. Получатель получает 8 младших битов источника.

Правило 2: LONG к SHORT

Знаковый разряд получателя получает знаковый разряд источника.

Получатель получает младших 15 битов источника.

Правило 3: REAL к LONG

Знаковый разряд получателя получает знаковый разряд источника.

Получатель получает младшие 31 бит источника.

Правило 4: DECIMAL к REAL.

Получатель наследует знак, всю целую и всю дробную часть источника.

Правило 5: STRING к REAL

Формат источника должен включать знак числа, целую часть, десятичную точку, дробную часть. Завершающие пробелы игнорируются. Запятая, или любой другой символ в источнике сбрасывает получатель в нуль.

Правило 6: PICTURE к REAL

Источник деформатируется в соответствии с его маской. Любой не подходящий символ сбрасывает получатель в нуль.

Правило 7: REAL к DECIMAL

Для **DECIMAL(m,n)** получатель использует знак, **m-n** знаков для целой части и **n** знаков для дробной части.

Правило 8: BYTE, SHORT или LONG к STRING

Получатель — это возможный минус, если число отрицательное и следующее числовое значение.

Правило 9: REAL к STRING

Получатель — это возможный минус, возможная целая часть, десятичная точка и возможная дробная часть чис-

ла. Очень большие и очень малые числа выдаются в научном формате.

Правило 10: BYTE, SHORT, LONG, REAL, DECIMAL, или PICTURE в PICTURE

Получатель форматируется согласно своему шаблону. Шаблоны времени и даты предполагают стандартные обозначения времени и даты.

Все другие шаблоны предполагают любое численное значение.

Правило 11: STRING к STRING, STRING к PICTURE или PICTURE к STRING

Если получатель короче источника, источник усекается. Если получатель длиннее источника, источник дополняется пробелами.

Clarion Internet Connect

Создаем приложения для Web

Internet Nonnect — продукт компании TopSpeed, позволяющий решить проблему создания приложений для Web за один шаг!

Nlarion и Nlarion Internet Connect совместно создают приложение, способное работать локально (то есть под Windows 9x или Windows NT), а также в Web с использованием любого броузера с поддержкой Java.

Вы можете запустить Web-приложение на любой платформе, допускающей работу броузера с поддержкой Java.

Internet Connect и Среда разработки Clarion

Среда разработки функционирует на нескольких уровнях и поддерживает каждый уровень пользователя/разработчика.

Автоматический разработчик приложений для Windows или Web

Когда необходимо «простое» приложение для ведения базы данных, вы можете создать его буквально за минуты, используя Clarion. Секрет — в словаре базы данных.

Если генератор приложений «знает», какие файлы и таблицы вы желаете иметь в приложении и как они связаны, он может построить приложение. Поэтому все, что вы должны сделать — это выбрать один или несколько файлов и установить (если файлов несколько) связи: «один ко многим» или «многие к одному» (если эти связи существуют).

Application Broker (брокер приложений) затем может создать полноценное приложение, и, посредством простой пометки поля в одном из диалогов Мастера, вы можете трансформировать приложение в Web-приложение. Полученное приложение может выполняться локально или в Web, используя Clarion Application Broker (брокер приложений). Это может сделать каждый. Только сначала надо выбрать файл из списка.

Визуальная Среда разработки для Windows или Web

В Clarion размещение управляющего элемента в окне дает вам гораздо больше, чем в других средствах быстрой разработки приложений, которые обычно позволяют добавить интерфейсный управляющий элемент пользователя, но предполагают самостоятельное написание кода для обеспечения его функционирования. В Clarion вы добавляете шаблон (template), который содержит элемент, данные и исполняемый код. Это означает, что вы не должны писать код — одно нажатие размещает полный рабочий вариант решения: управляющий интерфейсный элемент для пользователя и код, который позволяет ему выполнить его задачу. Кроме того, каждый шаблон имеет свой собственный интерфейс для настройки. При просмотре свойств шаблона вы можете увидеть закладку «**Actions**».

Помечая поля, выбирая варианты из выпадающих списков или заполняя поля редактирования, вы можете задать поведение шаблона в точном соответствии с вашими потребностями. В процессе изучения вы еще неоднократно будете устанавливать «**Actions**» для шаблонов.

После того, как вы опишете поведение шаблона через его интерфейс, генератор приложений создаст код (исходный код на языке Clarion), который обеспечит это поведение. Используя шаблоны, вы можете выполнить устрашаю-

щие объемы программистской работы, не написав вручную ни единой строчки кода.

Этот пример распространяется на приложения, создаваемые для Web. Вся упомянутая функциональность используется при интерпретации вашего приложения броузером. Проверка корректности обновления записи при одновременном доступе к ней нескольких пользователей и ссылочной целостности автоматически включается в ваше приложение, и похожим образом выполняется при работе через Web. Дополнительные параметры для Internet позволяют вам управлять обработкой событий таким образом, что вы имеете возможность определять, при каких условиях событие произойдет на сервере.

Квалифицированные пользователи, не имеющие возможности писать программы, легко смогут все это делать.

Совершенный язык программирования

На базовом уровне Clarion вы можете создавать приложения, целиком написанные вручную. Clarion — это язык четвертого поколения. У него есть высший уровень абстракции, имеющий компактную и очень хорошо читаемую грамматику для баз данных, которая позволяет вам легко обработать запись или наборы записей.

Вы не обязаны знать язык Clarion для создания приложений, но знание того, как он работает, позволит вам понять, что делает ваше приложение, и поможет в создании лучших приложений.

Библиотека Internet Builder Class открыта и пригодна для приспособления к Web-проектов, написанных вручную.

Профессиональные разработчики по достоинству оценят язык Clarion. Он с самого начала проектировался как язык для разработки бизнес-приложений. В дополнение к сравнительно небольшому времени изучения, вы получите блестящую производительность. Компилятор TopSpeed пре-

вратит весь ваш код, писали ли вы его вручную или он был написан генератором приложений, в высокооптимизированный машинный код.

Установка

Системные требования

Вы можете запустить среду разработки Clarion на любой системе, удовлетворяющей минимальным системным требованиям для Windows 9x, 2000 или Windows NT, XP. Web-пригодные приложения должны быть откомпилированы в 32 разряда, поэтому вы должны использовать Windows.

- Windows 9x, 2000 рекомендуется 16 или больше мегабайт RAM.
- Windows NT, XP рекомендуется 64 или больше мегабайт RAM.
- Минимум от 7 до 13 мегабайт свободного дискового пространства, в зависимости от параметров **SetUp**, которые вы выберете.

Система клиента может быть запущена на любой платформе, которая поддерживает работу Java-совместимого броузера. Приложения, которые вы разработаете с использованием Clarion Internet Connect, будут успешно выполняться на компьютерах, удовлетворяющих минимальным требованиям для этих броузеров. Производительность зависит от скорости соединения в Internet, но большинство приложений хорошо выполняется через модемное соединение 28,8 Кб.

Рекомендуемые броузеры

Хотя Web-пригодные приложения должны работать под любыми Java-совместимыми броузерами, лучшие результаты показывают следующие броузеры:

Windows 9x/2000/NT

Microsoft Internet Explorer 3.0x или более поздние

Netscape Navigator 3.0 или более поздние

Netscape Communicator 4.0 или более поздние.

Unix

Существует много вариантов UNIX, и некоторые из них имеют броузеры, написанные специально для них. Их слишком много, чтобы перечислять их здесь.

Netscape Navigator 3.01 или более поздние.

Netscape Communicator 4.03 или более поздние.

Apple/Macintosh

Microsoft Internet Explorer 3.0x или более поздние.

Netscape Navigator 3.0 или более поздние.

Netscape Communicator 4.0 или более поздние.

OS/2

Netscape Communicator 4.0 или более поздние.

Замечание: Инсталлятор устанавливает JSL файлы в сжатом формате (Clarion.CAB, Clarion.ZIP). Инсталлятор брокера также поставляет файлы .CLASS для поддержки 16-битных броузеров и броузеров для не-Windows платформ. Если у вас есть пользовательская версия для инсталляции, вы можете извлечь файлы .CLASS из архива Clarion.ZIP, используя 32-битную утилиту разархивации, поддерживающую длинные имена файлов.

Server System

Вы можете запустить Брокер приложений и Web-приложение на Windows 9x, 2000, однако рекомендуется Windows NT, XP для размещения рабочего комплекса.

- Для Windows 9x, 2000 рекомендуется 32 мегабайт RAM и статическое соединение с Internet/Intranet.
- Для Windows NT, XP рекомендуется 64 мегабайт RAM и статическое соединение с Internet/Intranet.

Производительность зависит от скорости канала подключения вашего сервера с Internet и трафика, который вы предполагаете при использовании вашего приложения. Приложение может быть доступно через модемное соединение со скоростью 28,8 Кб, но рекомендуется ISDN и выше.

Программа установки

Программа установки разворачивает и копирует файлы Clarion на ваш жесткий диск. Для всех операционных систем она предоставляет вам возможность выбора установки различных компонент, таких как файлы примеров.

Начало установки

Для запуска программы установки в Windows:

1. Вставьте инсталляционный CD в Ваш CD-ROM.
2. Из меню **Start** выберите **Settings ⇨ Control Panel**.
3. Выберите **Add/Remove Programs**, затем нажмите кнопку **Install**.

Windows Wizard проведет вас по процессу инсталляции.

Параметры установки

После запуска **Setup** вы увидите набор экранов мастера, содержащих некоторое число параметров.

1. Выбирайте параметры для установки, предлагаемые экранами мастера (такие как диск и директория установки) и перемещайтесь от экрана к экрану, используя кнопку **Next**.

Программа установки установит основные компоненты Clarion Internet Developer's Kit в соответствующие поддиректории определенной вами директории. Программа установки Clarion устанавливает все файлы в указанную директорию и в поддиректории, лежащие ниже ее. Она не устанавливает файлы в другие директории, за одним исключением. Версия ISAPI Application Broker требует, чтобы файл CWISAPI.DLL был установлен в директорию \Windows\System.

В процессе инсталляции индикаторы выполнения процессов отображают, как программа установки копирует файлы.

Нажмите кнопку **OK**, когда программа установки закончит свою работу.

Регистрация набора шаблонов IBC

Для того, чтобы зарегистрировать шаблоны IBC из Internet Developer's Kit, зарегистрируйте ICONNECT.TPL в Clarion Template Registry. Файл устанавливается в поддиректорию **..\TEMPLATE**.

Для регистрации набора шаблона Internet Developer:

1. Выберите **Setup ⇨ Template Registry** из меню Clarion.

2. Нажмите кнопку **Register** из диалога **Template Registry**.

3. В диалоге **Template Registry** выберите файл ICONNECT.TPL из директории **..\TEMPLATE**, после чего нажмите **OK**.

4. Нажмите кнопку **Close**.

Брокер приложений

Application Broker (брокер приложений) обеспечивает вам связь с приложениями, созданными при помощи Inter-

net Connect. Internet Developers Kit состоит из двух частей: IBC Templates/Library и Application Broker.

Шаблоны (templates) делают Web-пригодными ваши Clarion-приложения, а Брокер приложений разрешает доступ к ним через Web.

При использовании Брокера приложений Clarion отсутствует необходимость в каком-либо дополнительном программном обеспечении для Web-сервера. При этом вы позволите конечным пользователям при помощи броузера манипулировать Web-приложениями, загруженными на сервере.

Запуск брокера приложений

Есть две версии Брокера приложений — в виде исполняемого модуля (.EXE) и версия ISAPI DLL (.DLL). Первый вариант инициируется запуском исполняемого модуля. Версия ISAPI вызывается по требованию.

1. Запустите брокер приложений двойным нажатием мыши на машине-сервере. По умолчанию он использует порт 80. Если вы запустили какой-либо другой Web-сервер, использующий порт 80, то можете определить другой порт (например 8080) для Брокера приложений — в параметрах установки или передавая порт в качестве параметра при вызове из командной строки.

Важно: Если вы определите номер порта в командной строке, вы не сможете изменить его в параметрах установки брокера. При запуске брокера приложений разрешите ему предупреждать вас, если он не может использовать порт, определенный по умолчанию, затем измените параметры порта, после чего он запомнит эти установки и не будет требовать определения порта при очередном запуске.

Использование версии на одно соединение

Internet Developers Kit содержит версию брокера приложений на одно соединение (CWBrokr1.EXE). Он содержит также ISAPI-версию брокера приложений на одно соединение.

Версия брокера приложений на одно соединение полностью совпадает по функциям с полной версией брокера приложений, за исключением того, что разрешает доступ только к одному IP-адресу за один сеанс. Это означает, что один пользователь способен запустить несколько копий одного исполняемого модуля или несколько других, открывая новые окна броузера.

Для того, чтобы обеспечить доступ другому пользователю, все приложения, запущенные первым, должны быть закрыты.

Пользователь должен выйти из приложения, чтобы приложение закрылось; простое закрытие броузера не закрывает приложения. Если пользователь закроет броузер, приложение не закроется, пока не истечет **timeout**. По умолчанию шаблоны устанавливают **timeout** для приложений 10 минут. Вы можете изменить этот параметр в **Global Internet Application Extection Template**.

Вы можете тестировать многопользовательский доступ (то есть, моменты в приложении, связанные с одновременным доступом к приложению более чем одного пользователя) с помощью однопользовательской версии брокера приложений, запуская два экземпляра броузера и запуска в каждом из них своей копии приложения.

Для запуска нового экземпляра вашего броузера:

В Internet Explorer: выберите **File ⇨ New ⇨ Window**.

В Netscape Navigator: выберите **File ⇨ New ⇨ Navigator-Window**.

Подключение к вашему приложению

Для запуска приложения из броузера обеспечьте Uniform Resource Location в следующем формате:

```
http://nnn.nnn.nnn.nnn/appname.exe.0
```

где **appname.exe** — имя вашего исполняемого файла, а **nnn.nnn.nnn.nnn** — Ваш IP-адрес (не забудьте добавить (.0) после имени исполняемого модуля), или

```
http://domain.ext/appname.exe.0
```

если ваш IP-адрес имеет имя, зарегистрированное на **Domain Name Server**.

Если вы запустили broker через порт, отличный от порта 80 (HTTP-порт по умолчанию), пользователи должны включить адрес порта в URL.

Примеры:

```
http://nnn.nnn.nnn.nnn:8080/appname.exe.0
http://domain.com:8080/appname.exe.0
```

Использование гиперссылок для запуска приложений

Вы можете также создать гиперссылки на другую Web-страницу для запуска Web-пригодных приложений.

Примеры:

```
<HREF="http://nnn.nnn.nnn.nnn/appname.exe.0">
CLickto start app</a>
```

или

```
<HREF="http://nnn.nnn.nnn.nnn:8080/appname.exe.0">
CLickto start app</a>
```

Firewalls и альтернативные порты

Firewalls и proxy-сервера могут быть сконфигурированы с ограничением доступа к портам, отличным от порта по умолчанию (80). Для обеспечения доступа пользователям, находящимся за firewall и proxy-серверами, у вас есть следующие возможности:

1. Использовать ISAPI-версию брокера приложений и запустить ваш web-сервер через порт 80.

2. Запустить брокер приложений через порт 80.

3. Поручить сетевому администратору удалить ограничение на порт 8080 (или на порт, который вы используете).

Версия ISAPI брокера приложений

Internet Connect содержит также Application Broker, который использует расширение сервера. ISAPI (Internet Server Application Programming Interface). Вы можете использовать эту версию с Microsoft Internet Information Server (IIS) или другими ISAPI-совместимыми web-серверами. Эта версия брокера приложений Clarion разработана для обеспечения секретности и использует шифрование Secure Socket Layer (SSL). Здесь применяется мощная технология для шифрования данных, передаваемых через Internet. Непосвященному невозможно расшифровать информацию. Когда вы видите золотую иконку «**замок**» в строке состояния вашего броузера — знайте, что вы находитесь на секретном уровне, и все отправляемые и получаемые данные шифруются.

Важно: Personal Web Server не обеспечивает SSL шифрование, поэтому секретность не будет обеспечена, когда вы используете ISAPI Брокер приложений — Personal Web Server. Именно это позволяет разрабатывать и запускать приложения на машине с Windows 9x, 2000 и размещать его позже на компьютере с Windows NT, XP под IIS.

Как он работает

Брокер приложений Clarion выполнен в виде ISAPI Server extention Dynamic Link Library (DLL). Запрос от клиента (броузера) производится к web-серверу, определяя базовый DLL и web-приложение для выполнения. После этого в память загружается брокер приложений, и он

поддерживает связь между клиентом и серверным приложением так же, как это делает версия с исполняемым модулем.

Когда ваше web-приложение производит секретный запрос (это определяется для процедуры в **Internet Options**), брокер приложений принимает запрос через уровень **secure sockets** и остается в секретном режиме до тех пор, пока процедура активна.

Шифрование надо использовать разумно. **Secure Sockets Layer** (SSL) замедляет производительность между HTTP-серверами и броузерами. По этой причине Internet Connection позволяет вам переключаться между нормальным и секретным режимами в любой момент работы в приложении. Использование шифрования только в те моменты, когда это необходимо, очень сильно повысит производительность.

Установка и подготовка к работе Clarion Application Broker (брокера приложений Clarion) ISAPI DLLs:

1. Установите Microsoft Information Server (IIS) или Personal Web Server (PWS).

2. Получите Secure Sockets Layer (PCT/SSL) Digital Certificate и установите его.

Важно: Вы можете использовать ISAPI DLLs без обеспечения SSL. Если вы не хотите использовать SSL, установите PWS-версию Брокера приложений и пропустите шаг 2. PWS не поддерживает SSL или digital certificat.

3. Установите ISAPI DLL-версию брокера приложений Clarion. Программа установки предлагает вам определить следующее:

Устройство для установки

Устройство на которое будет установлен брокер.

Корневая директория для установки

Корневая директория для установки. Это не должна быть корневая директория вашего web-site.

Пароль на брокер приложений

Это пароль, необходимый для доступа к настройке брокера. Вы должны его обеспечить.

IP-адрес или Domain Name сервера

Domain Name сервера или его IP-адрес.

Используемый сервер

Тип web-сервера, с которым будет работать программа установки.

Варианты для выбора — **Personal Web Server** или **Internet Information Server**. Если вы не хотите устанавливать **Digital Certificate**, выберите **PWS**.

Программа установки создает эти директории внутри корневой установочной директории:

```
\scripts
\sec_scr
\secure
\public
\exec
```

Инсталляционная программа также создает текстовый файл (CWISET.TXT), перечисляющий созданные директории и установки, которые вы должны сделать на вашем Web-сервере.

4. Используя утилиту IIS Internet Service Manager или PWS Administrator (из **Control Panel**), установите права доступа для директорий, перечисленных в CWISET.TXT.

Например, при использовании C: как устройства для инсталляции и **CWICICWeb** — как корневой инсталляцион-

ной директории, вы должны установить следующие права в IIS:

Directory	Alias	Rights
C:\CWICWeb\public	/cwpub	Read
C:\CWICWeb\secure	/cwsec	Read & SSL
C:\CWICWeb\scripts	/cws	Execute
C:\CWICWeb\sec_scr	/cwss	Execute & SSL
C:\CWICWeb\exec	none*	none*

Важно: Установки для SSL нужны только при использовании digital certificate.

5. Используя утилиту **User Manager** (для NT), предоставьте пользователю Internet Guest права **WRITE** и **DELETE** в директориях **\PUBLIC** и **\DELETE**.

Важно: При использовании Personal Web Server нет необходимости в установке прав доступа для пользователя.

6. Поместите ваше web-приложение в директорию C:\CWICWeb\Exec (или в директорию ниже ее). Вместе с ними вы должны поместить в ту же директорию все необходимые для работы приложения файлы .DLL (драйверы файлов, rintime библиотеки и др.).

Важно: 32-х-разрядная Clarion Runtime Library (CW2RUN32.DLL), 32-разрядный DOS-файл-драйвер CW2-dos32.dll) и 32-разрядный ASCII-файл-драйвер (CW2ASC-32..DLL) также необходимы, даже если вы не используете файлы этого типа в вашем .APP (если только приложение не скомпилировано как Local Link .EXE-модуль).

7. Разместите все необходимые для вашего приложения файлы пиктограмм в C:\CWICWeb\Public. В эту же директорию можете поместить и .GIF или .JPG файлы для вашего приложения. Web-приложение будет автоматически извлекать файлы изображений, но если они размещены здесь, то этот шаг может быть пропущен.

Пиктограммы, отображаемые в списках или на кнопках, должны быть размещены в этой директории.

8. Создайте вашу Web-страницу для вызова ваших Web-приложений, используя следующее соглашение для гиперсвязей:

```
<A HREF="http://domainname.com/cws/cwlaunch.dll/
AppName.exe.0">Click to start app</a>
```

Утилита установки для ISAPI Broker

Для запуска утилиты установки брокера приложений:

1. Обратитесь из броузера к следующему URL:

```
http:/nnn.nnn.nnn/cws/cwlaunch.dll/AppBroker/
http:/domainname.com/cws/cwlaunch.dll/AppBroker/
```

Важно: строка должна содержать завершающий бэкслэш.

Появляется диалог аутентификации пользователя Броузера.

2. Введите в **Quick** имя пользователя и пароль (смотрите параметры установки брокера приложений), затем нажмите кнопку **OK**.

Появится страница удаленного доступа установки брокера приложений. Эта страница содержит гиперсвязи для удаленного управления брокером.

3. Нажмите на гиперсвязь **Setup**. Это вызовет страницу удаленной настройки.

Файлы, размещаемые программой инсталляции установки брокера приложений Clarion (Application Broker)

CWLAUNCH.DLL

C:\CWICWeb\scripts

CWSECURE.DLL

C:\CWICWeb\sec_scr

CWISAPI.DLL

\WinNT\System32 или \Windows\System

CLARION.ZIP

C:\CWICWeb\public

CLARION.CAB

C:\CWICWeb\public\

***.class**

C:\CWICWeb\public

Важно: программа инсталляции устанавливает файлы JSL в сжатом формате (Clarion.Cab и Clarion.Zip) и в развернутом виде — для поддержки 16-разрядных Броузеров и браузерах для не-Windows-платформ.

Директории

Версия EXE

При запуске EXE-версии установки брокера приложений директория, из которой он запускается — это виртуальная корневая директория для исполняемых модулей, а директория **\Public** — это виртуальная корневая директория для всех получаемых файлов. Например, ссылка на HTML-документ в директории **\Public** должна адресоваться как:

```
http://nnn.nnn.nnn.nnn/index.htm
```

В то время как на исполняемый модуль, находящийся в той же директории, что и установки брокера приложений, ссылка определяется похожим URL:

```
http://nnn.nnn.nnn.nnn/appname.exe.0
```

Хотя эти файлы находятся в разных директориях, брокер приложений управляет маршрутизацией и не требуется никакой информации о директориях.

Если вы хотите хранить информацию в разных директориях, можете вызывать приложение, используя его относительный адрес. Например, в случае исполняемой версии брокера, запущенного в C:\CWICWeb и приложения с именем myapp.exe, размещенного в C:\CWICWeb\Exec\Appl, вы можете вызвать его, используя:

```
http://mydomain.com/exec/appl/appname.exe.0
```

Версия ISAPI

При использовании ISAPI-версии Application Broker, директории и их алиасы определяются в параметрах настройки и в установках web-сервера.

Если вы хотите хранить приложение в нескольких директориях, вызывайте приложение, используя его относительный путь. Если, например, мы используем ISAPI-версию брокера с C:\CWICWeb\Exec в качестве пути к исполняемым файлам, и приложение с именем myapp.exe, размещенное в C:\CWICWeb \Exec\Appl, то вызывать это приложение следует используя:

```
http://mydomain.com/cws/cwlaunch.dll/app1/
apname.exe.0
```

Параметры установки Application Broker (брокера приложений)

В Setup для установки брокера приложений вы можете определить следующие параметры:

Default Home Page

Документ, который предоставляется по умолчанию, если не определен конкретный URL. По умолчанию — это index.htm.

Public Directory

Директория, в которой размещены совместно используемые файлы (файлы HTML, изображения). Это также ди-

ректория, в которой создаются временные директории, содержащие HTML-файлы, представляющие приложения.

По умолчанию это /**Public**.

Port Number

HTTP-порт, к которому «привязан» Application Broker. Если он определен в командной строке Broker, то эта опция отключается и не может быть изменена.

Если порт отличен от 80 (HTTP порт по умолчанию), то клиенты (или ваши гиперсвязи) должны определять номер порта в URL. Например, если broker подключен к порту 8080, вы должны определить:

```
http://nnn.nnn.nnn.nnn:8080/appname.exe.0
```

Use Log File

Включение этого признака разрешает трассировку сервера. Для каждого запроса, сделанного клиентом через Application Broker, создается запись в файле трассировки. Эта запись содержит IP-адрес источника запроса, дату и время запроса и сам запрос. Запомните, что файл трассировки непрерывно растет.

Log File Name

Определяет имя файла трассировки.

Remote password

Определяет пароль, разрешающий удаленный доступ к Application Broker.

Max Simultaneous Connections

Определяет максимальное число одновременных подключений. Если пользователь пытается получить доступ к брокеру, превышая это число, ему возвращается предупреждение.

Use Debug Setup

Этот параметр определяет запуск приложений в режиме отладки. Для запуска отладчика у вас должен быть обеспечен доступ к машине-серверу. В этом поле определите путь к файлам отладчика и переназначения (redirection). Например:

```
C:\cw20\bin\cw2db32.exe  c:\cw20\bin\cw20.red
```

Удаленный доступ к брокеру приложений

Если вы обеспечите пароль для удаленного доступа в **Setup** брокера приложений, то можно будет, используя броузер, получать доступ к брокеру приложений с любой удаленной площадки. Это позволит вам модифицировать параметры установки, запрещать и разрешать доступ к приложениям, и смотреть статус последнего запроса.

Для получения доступа из броузера к брокеру необходимо:

1. Ввести следующий URL в поле **Address Location** вашего броузера и нажать **Enter**.

```
HTTP://domainname.ext/appbroker/
```

или

```
HTTP://nnn.nnn.nnn.nnn/appbroker/
```

где **domainname.ext** — это имя вашего сервера в домене, а **nnn.nnn.nnn.nnn** — его IP-адрес.

Обязательно ставить заключающий слэш (/).

Появится диалог для ввода пароля.

2. Введите CWIC в поле определения имени пользователя и пароль, который вы ранее определили.

Появится web-страница с гиперссылками на следующие функции:

Setup

Обеспечивает доступ к модификации параметров установки. Вы можете модифицировать любой из параметров, за исключением номера порта и пароля для удаленного доступа. Эти параметры могут изменяться только на сервере.

Suspend (Запрещение)

Позволяет запретить доступ к web-приложению, размещенному на сервере с брокером приложений. Если приложение находится в работе, сессия и приложение закрываются. Пользователи, работающие с программой, получают сообщение, что приложение закрыто для проведения обслуживания. Это позволяет проводить обновление версий приложений или файлов с данными.

Для запрещения доступа введите полное имя исполняемого файла (включая расширение), а затем нажмите кнопку **OK**. Если приложение, для которого вводится запрещение, находится в поддиректории, то нет необходимости указывать здесь ее имя. Web-страница возвращает список приложений, которые запрещены в настоящий момент.

Enable

Позволяет вам снова сделать доступными web-приложения, размещенные на сервере с брокером приложений и имеющие статус запрещенных.

Чтобы разрешить работу с этими приложениями, введите полное имя исполняемого файла (включая расширение), затем нажмите кнопку **OK**. Web-страница возвратит список все еще запрещенных приложений.

Status

Отображает текущий статус приложений, запущенных на вашем сервере с Application Broker, включая время доступа и IP-адреса клиентов, работающих с приложениями.

Порядок разработки и размещения приложений

1. Создайте приложение (или откройте существующее).

2. Превратите приложение в web-приложение, добавив Global Internet Application Extention Template.

3. Откомпилируйте приложение в 32-х-разрядном режиме (в local или standalone).

4. Установите и настройте Брокер приложений (CWBroker.exe, или CWBrokr1.exe, или версию ISAPI.DLL).

5. Разместите приложение и другие необходимые файлы (DLL, файлы данных, файлы изображений, и др.) в директорию, из которой будет вызываться Брокер приложений (или поддиректории).

Важно: даже если вы их не используете в своем приложении, необходимы (если только приложение не скомпилировано с параметром **Local Link**): 32-х разрядная Clarion Rintime Library (CW2RUN32.DLL), 32-х разрядный драйвер для DOS-файлов (CW2dos32.DLL) и 32-х разрядный драйвер для ASCII-файлов (CWASC32.DLL).

Исполняемые модули и необходимые для работы файлы должны располагаться в той же директории, что и приложение, или в директории, указанной в PATH (Windows 9x), или в system path для пользователя (account) Internet Guest (Windows NT). Другими словами, все .DLL, необходимые для работы приложения, запущенного на сервере, должны быть видны приложению.

6. Инсталляционная программа создает поддиректорию **\Public** в директории самого брокера. Эта директория использует Брокер приложений для передачи файлов (таких как Java Classes из Java Support Library, графические или другие HTML файлы) и не передает каких-либо файлов из своей собственной директории.

Это предотвращает «скачивание» файлов с данными или исполняемых модулей с вашего сервера. Ваше приложение на ходу создает HTML-файлы и автоматически размещает их во временных поддиректориях ниже **\Public**.

Важно: Программа установки размещает JSL-файлы в сжатом формате (Clarion.CAB или Clarion.ZIP) и в несжатом формате — для поддержки 16-ти-разрядных броузеров и броузеров для не-Windows платформ.

В момент работы для каждого клиентского соединения создается временная директория. По завершению соединения эти директории автоматически удаляются.

Изображения распаковываются в эту директорию, когда они не находятся директории **\Public**. Пиктограммы, отображаемые в списках (**LIST**) или на кнопках (**BUTTON**), должны быть размещены в директории **\Public**.

7. Запустите на сервере Application Broker. По умолчанию он использует порт 80. Если вы запускаете какой-либо другой web-сервер, который использует порт 80, то для Брокера приложений Вы можете определить другой порт (например 8080), создав ярлык с номером порта в качестве параметра командной строки. Вы также можете можете изменить порт, к которому привязывается Application Broker, через параметры настройки.

Важно: Если вы определяете номер порта в командной строке, то не можете изменить его в параметрах настройки. Если вы готовите к работе брокер приложений, то позвольте ему, если он не может использовать порт по умолчанию, предупредить вас, дав затем вам возможность изменить настройку порта и сохранить изменения.

В результате не придется определять номер порта при каждом запуске брокера приложений.

8. Предоставьте пользователям, которые хотят иметь доступ к приложению, URL в следующем формате:

```
http://nnn.nnn.nnn.nnn/appname.exe.0
```

или

```
http://domain.ext/appname.exe.0
```

где **appname.exe** — это имя вашего исполняемого модуля. Убедитесь, чтобы пользователь добавлял (**.0**) после имени исполняемого модуля.

Если вы запускаете Брокер приложений с использованием порта, отличного от 80 (порт HTTP по умолчанию), пользователи должны будут включать номер порта в URL. Например,

```
http://nnn.nnn.nnn.nnn:8080/appname.exe.0
```

Если ваше приложение имеет параметры, передаваемые через командную строку, добавьте знак вопроса и сам параметр:

```
http://nnn.nnn.nnn.nnn/appname.exe.0?myargument
```

Локальное тестирование

Хоть для правильного размещения web-приложения вы должны использовать постоянный IP-адрес, вы также можете использовать IP адрес, динамически присваиваемый в момент соединения по коммутируемым линиям. Это употребимо для создания одноразовых демонстраций через Internet. Для нахождения вашего динамического адреса вы можете запустить applet, включенный в операционную систему. Для локального тестирования (с брокером и броузером на одной машине), вы можете использовать «заглушку» IP адреса: 127.0.0.1.

В Windows запустите утилиту Winipcfg.exe (из директории Windows). Она покажет ваш адрес, как это показано ниже. Если Ваш TCP/IP привязан к сети и к модему, используйте выпадающий список для выбора модема.

В Windows NT используйте **Dial Up Neworking Manager** для того, чтобы узнать ваш IP-адрес. После дозвонки и

подключения к вашему провайдеру нажмите на пиктограмму в System Tray.

Проверка вашего TCP/IP соединения

Существует несколько способов проверки вашего TCP/IP соединения.

1. В приглашении DOS наберите:

```
PING 127.0.0.1
```

Это проверит TCP/IP соединение на вашем локальном компьютере и покажет время, затраченное на ответ.

2. Запустите ваш web броузер, введите следующий URL машины-сервера и тестовой страницы:

```
http://server.domain.com/index.htm
```

где **server.domain.com** — это IP-адрес вашего сервера.

Если вы присвоили вашему серверу порт, отличный от 80 (порт HTTP по умолчанию), вы должны будете включить номер порта в URL, как показано ниже:

```
http:// server.domain.com:XX/index.htm
```

где **XX** — номер порта, присвоенного Web-серверу.

Если у вас нет TCP/IP соединения или вы хотите проверить соединение на своей локальной машине, вы можете использовать следующий URL:

```
http://localhost/index.htm
```

или

```
http://127.0.0.1/index.htm
```

Это внутренний локальный loopback (заглушка) адрес.

Как создать Web-приложение

В Clarion вы можете создать приложение из словаря данных без необходимости ручного программирования. Все, что вы должны сделать — это создать **Data Dictionary** (Словарь Данных) и использовать **Application Wizard** (Мастер

Приложений) для создания полного Windows приложения — в минуты! При установленном **Internet Connect**, в **Application Wizard** появляется добавочный check box, который позволяет превратить создаваемое вами приложение в web-приложение. Это позволит вам создать web-приложение с помощью единственного дополнительного нажатия кнопки мыши.

Мастер Web-приложений

Создание гибридных Web/Windows приложений

Вы должны иметь установленную среду разработки Clarion. Предполагается, что вы установили Clarion в директорию C:\CW20, а Брокер приложений — в директорию C:\CWICWEB. Если вы используете другие директории, то должны будете соответственно изменить команды.

Создайте ваше первое Calrion приложение для Web.

1. В диалоге **Pick** нажмите кнопку **New...**

Появится диалог **New**.

2. Выберите закладку **Application**.

3. Выберите **C:\CW20\Examples\WebTutor** из списка **Directories**.

4. Наберите **WebOrder** в поле **File Name**.

5. Убедитесь, что не отмечен признак **Use Quick Start Wizard**, затем нажмите кнопку **Create**.

Появится диалог **Application Properties**.

6. Нажмите кнопку справа от поля ввода **Dictionary File**.

Появится диалог **Select Dictionary**.

7. Выберите **C:\CW20\Examples\WebTutor\WebOrder.dct** и нажмите кнопку **OK**.

Запуск Application Wizard

1. Отметьте признак **Application Wizard**, затем нажмите кнопку **OK**.

2. Нажмите кнопку **Next**.

3. Нажмите кнопку **Next**.

4. Нажмите кнопку **Next**.

5. Пометьте признак **Create an Internet enabled application**, затем нажмите кнопку **Next**.

Этот шаг внесет два свойства в приложение, создаваемое Application Wizard:

- Он сделает его 32-разрядным
- Добавит к приложению Internet extention templates (шаблоны-расширения для Internet).

6. Нажмите кнопку **Finish**.

Application Wizard создаст приложение.

Создание приложения

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов). Ваше первое web-приложение готово к размещению и запуску.

2. Нажмите кнопку **OK** в окне компилятора.

Размещение приложения

На последнем шаге был создан WebOrder.exe. Так как это приложение является web-пригодным, оно может работать как под Windows (как стандартный исполняемый модуль Windows), так и в Web (через Application Broker, используя browser). Теперь разместим приложение и все необходимые для его работы файлы.

Заметим, что мы просто размещаем его в другую директорию на той же машине, но процесс в точности такой же, как и при размещении на машине-сервере.

1. Откройте Windows Explorer (или Windows NT Explorer).

2. Скопируйте WebOrder.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\EXEC\WebTutor.

Запомните, что простое перетаскивание файлов в Explorer создает ярлык для исполняемых модулей. Если вы используете метод «перетащить и опустить», вы должны нажать правую кнопку мыши и выбрать команду **Copy** из выпадающего меню.

Важно: Мы располагаем файлы данных для примера в двух директориях.

Если у вас есть локальные файлы с данными, вы также должны разместить их.

3. Скопируйте перечисленные ниже файлы из директории C:\CW20\BIN в директорию C:\CWICWEB\EXEC\WebTutor:

```
CW2RUN32.DLL
CW2TPS32.DLL
CW2ASC32.DLL
CW2DOS32.DLL
```

Это — DLL, поддерживающие работу вашего приложения, включающие rintime-библиотеки и драйверы баз данных.

Этот шаг включен сюда, хотя это может и не быть необходимым при работе под Windows (на вашей машине для разработки приложений), так как пути к этим файлам содержатся в вашей переменной **PATH**. Однако NT-сервер работает по-другому. У каждого пользователя есть переменная **PATH** и размещение вместе с приложением DLL-файлов гарантирует, что каждый пользователь получит доступ как к файлам приложения, так и к файлам поддержки.

4. Запустите Application Broker двойным нажатием левой кнопки мыши на файле CWBrokr1.exe (или CWBroker.exe, если у вас полная версия Application Broker) расположенном в C:\CWICWEB.

5. Запустите ваш любимый browser.

Затем проверьте работу Application Broker и ваши установки TCP/IP, используя метод **LocalHost loopback** (заглушка).

6. В строке URL вашего броузера наберите:

```
http://localhost/btest.htm
```

или

```
http://127.0.0.1/btest.htm,
```

а затем нажмите **Enter**.

Важно: Если ваш broker настроен на порт, отличный от порта 80, добавьте его номер в URL.

Например:

```
http://localhost:8080/btest.htm
```

или

```
http://127.0.0.1:8080/btest.htm,
```

а затем нажмите **Enter**.

Если проверочная web-страница отобразится корректно, то ваш Application Broker установлен и работает правильно. Если нет, то вы должны вернуться к предыдущей главе и переконфигурировать настройки.

Теперь запустите приложение в броузере:

7. В строке URL вашего броузера наберите:

```
http://localhost/exec/webtutor/weborder.exe.0
```

или

```
http://127.0.0.1/exec/ webtutor/weborder.exe.0
```

а затем нажмите **Enter**.

Важно: Если ваш broker настроен на порт, отличный от порта 80, добавьте его номер к доменной части URL. Например:

```
http://localhost:8080/exec/webtutor/weborder.exe.0
```

Итак, запущено ваше первое web-приложение!

Сейчас вы можете исследовать это новое приложение и сравнить его работу в web и Windows. Вы заметите, что есть несколько незначительных отличий (из-за различий платформ), но в целом впечатления от работы приложения почти одинаковы.

8. Когда вы закончите, нажмите на гиперсвязь **Exit**. Это закроет приложение. Заметим, что browser отобразит синюю web-страницу с гиперсвязью для перезапуска приложений. Эта страница создается Application Broker автоматически, если только вы не определили страницу для возврата из приложения в **Global Internet Application Extension template** (в шаблоне).

Оставьте ваш броузер открытым с отображенной страницей перезапуска. Вы будете использовать эту страницу для перезапуска вашего приложения.

Оптимизация приложения

Web поднял одну проблему программирования — ограниченная пропускная способность. Очень важно использовать все доступные методы для уменьшения количества данных, передаваемых по сети. Многие пользователи подключаются к web по телефонным линиям, используя модем, что является относительно медленным сетевым подключением.

Clarion Internet Connect был разработан с учетом возможной низкой пропускной способности сети. Управляющие элементы Java, которые он создает, в основном не требуют обновления всей страницы при собственном

динамическом обновлении в броузере клиента. Эта форма «динамического HTML» требует передачи только небольшого количества данных. Это называется «частичным обновлением».

В этом случае обновляются только управляющие элементы, которые используются для ввода/ отображения вводимых данных. Поля ввода, строковые (string) элементы Java, Java элементы изображения (Image) и Java-списки (Listbox) обычно являются доступными для динамического обновления.

По той же причине (понижение трафика) многие управляющие элементы вызывают частичное обновление. Например, выбор новой записи в списке вызывает частичное обновление, позволяя большинству управляющих элементов отобразить текущие данные.

Есть однако несколько примеров, когда частичное обновление подходит к случаю, но не включено по умолчанию. Инициирование частичного обновления вместо полного, там где это употребимо — это один из лучших способов оптимизации Ваших web-приложений.

Существует много случаев, когда подходит вариант частичного обновления, но по умолчанию установлен режим полного обновления. Так делается потому, что шаблоны (templates) не могут предвидеть всех возможных вариантов и поэтому предпочитают более безопасное полное обновление более быстрому частичному.

Например, multi-sorted (сортируемый по нескольким ключам) список, который не имеет управляющих элементов на закладках (**Tabs**) обрабатывается быстрее, если вы используете **Individual Control Overrides** (индивидуальные настройки управляющего элемента) для определения частичного обновления для момента, когда выбирается новая закладка (**Tab**). При этом будут обновляться только данные в списке (**listbox**) вместо обновления всей страницы.

Давайте посмотрим на приложение, которое мы только что создали.

1. Переключитесь назад в ваш browser.

2. Нажмите на гиперсвязь перезапуска.

Внутри броузера появится приложение **WebOrder**.

3. Нажмите на гиперсвязь **Browser Customer Information File**.

В броузере появится «окно» **Browse the Customer File**. Заметьте, что окно содержит список (**listbox**) и две закладки (**Tabs**). Выбор закладки изменяет порядок сортировки в списке.

4. Нажмите на каждую из закладок и отследите поведение web-страницы.

Вы должны заметить, что для обновления списка была заменена вся страница.

Такое поведение присваивается по умолчанию всем листам (**sheets**) и закладкам (**Tabs**). Чуть ниже мы изменим это поведение.

5. Нажмите на синюю клавишу **X** в правом конце панели инструментов для закрытия окна **Browse**, затем на гиперсвязь **Exit** для выхода из приложения.

Покиньте ваш броузер, оставив открытой страницу перезапуска приложения.

Вы будете использовать это для перезапуска приложения после того, как сделаете некоторые изменения.

Internet Procedure Extention Template (шаблон расширения для Internet)

Теперь мы переопределим поведение управляющего элемента **SHEET** с целью оптимизации его производительности при работе в web.

Перед началом работы у вас должна быть загружена среда Clarion, и в ней открыт файл weborder.app.

1. В дереве приложения (**Application Tree**) выберите закладку **Template**.

Это отсортирует процедуры по типам шаблонов (**template**). Заметим, что четыре процедуры используют шаблон **Browse**.

2. Выберите процедуру **BrowseCustomer** и нажмите кнопку **Properties**.

Появится окно **Procedure Properties**.

3. Нажмите кнопку **Internet Options**.

4. Выберите закладку **Controls**.

5. Выберите управляющий элемент типа **Sheet** в списке **Individual Control Options** (индивидуальные параметры элемента).

6. Нажмите кнопку **Properties** (свойства), затем выберите закладку с названием **Events** (события).

7. Выделите событие **Accepted**, затем нажмите кнопку **Properties**.

Переопределение полного обновления (по умолчанию) на частичное обновление

1. Установите признак **Override default action** (переопределить действие по умолчанию), затем выберите из выпадающего списка **Partial page refresh** (частичное обновление страницы).

2. Нажмите кнопку **OK** во всех окнах, пока не вернетесь в дерево приложения (**application tree**) (4 раза).

3. Повторите последние девять шагов для трех других Browse процедур.

4. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к повторному размещению.

5. Откройте Windows Explorer (или Windows NT Explorer).

6. Скопируйте Weborder.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\EXEC\WebTutor.

На этот раз вы должны разместить только приложение, а файлы DLL не меняются.

Давайте запустим приложение и понаблюдаем, как изменилось его поведение после сделанных нами изменений.

Просмотр отличий

1. Переключитесь в ваш броузер.

2. Нажатием на гиперсвязь **Restart**, запустите приложение в броузере.

3. Нажмите кнопку мыши над гиперсвязью **Browse Customer Information File**.

4. Нажмите кнопку мыши над каждой из закладок и понаблюдайте поведение web-страницы.

Вы заметите, что список (**list**) переотображает данные без посылки всей страницы.

5. Выйдите из приложения.

Покиньте ваш browser, оставив открытой страницу перезапуска приложения. Вы будете использовать это для перезапуска приложения после того, как сделаете некоторые изменения

Добавление графики

Web создает для пользователя приятное, полное красок пространство. Многие web-сайты (sites) создаются с целью обеспечения одновременно содержания и притягатель-

ного интерфейса. Clarion Internet Connect поддерживает наиболее часто употребляемые в web-страницах методы использования цветов и графики.

Мы добавим фоновое (background) изображение к страницам на которых появляются окна приложения. Это обеспечит задний план приложения и позволит визуально отследить, что на экране относится к приложению, а что — нет.

Internet Application Extention Template — шаблон расширения для Internet-приложений. Сначала мы добавим фоновое изображение (у вас должна быть загружена среда Clarion, и в ней открыт файл weborder.app):

1. В дереве приложения, нажмите кнопку **Global**.

Появится окно **Global Properties**.

2. Нажмите кнопку **Extentions**.

3. Появится окно **Extentions and Control Templates**.

4. Выделите **Internet Application Extention**.

Появится стандартный Windows диалог для выбора файла.

5. Выберите Crumpled.gif, затем нажмите кнопку **OK**.

Добавится изображение к фону web-страницы. Это — изображение скомканного листа серой бумаги. Запомните, что этот файл с изображением должен быть тоже размещен.

6. В области **Window** нажмите кнопку, расположенную за **BackGround Color**.

Появится стандартный диалог для выбора цвета.

7. Выберите цвет **Silver** (серебряный), нажмите **OK**.

Добавится цветовой атрибут к HTML-представлению окна приложения.

Кроме добавления цвета, это будет препятствовать взгляду сквозь фоновое изображение.

8. Нажмите кнопку **OK** в окнах **Extentions and Control Templates** и **Global Properties**.

Создание, размещение и запуск приложения

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к повторному размещению.

2. Откройте Windows Explorer (или Windows NTExplorer).

3. Скопируйте Weborder.exe из C:\CW20\ Examples\ WebTutor в C:\CWICWEB\EXEC\WebTutor.

4. Скопируйте Crumpled.gif из C:\CW20\Examples\ WebTutor в C:\CWICWEB\Public..

5. Переключитесь в броузер и перезапустите приложение. Посмотрите на новый вид приложения.

Преобразование готовых Clarion-приложений в Web-приложения

Перенести в web существующее приложение Clarion так же легко, как создать новое web-приложение.

Превращение Clarion-приложения в web-пригодное:

1. В диалоге **Pick** нажмите кнопку **Open...**

Появится диалог **Open**.

2. Выберите закладку **Application**.

3. Выберите из списка директорий директорию C:\CW20\Examples\WebTutor, выберите WebTree.app, затем нажмите кнопку **Open**.

Появится **Application Tree** (дерево приложения).

4. В дереве приложения нажмите кнопку **Global**.

Появится **Global Properties** (глобальные свойства).

5. Нажмите кнопку **Extensions**.

Появится окно **Extentions and Control Templates** (шаблоны расширений и управляющих элементов).

6. Нажмите кнопку **Insert**.

7. Выделите строку **Internet Application Extension**, затем нажмите кнопку **Select**.

Тем самым добавляется шаблон Internet Application Extention, который автоматически добавляет шаблон Internet Procedure Extension к каждой процедуре приложения.

8. Нажмите кнопку **OK** в окнах **Extentions and Control Templates** и **Global Properties**.

Изменение на 32-бит

1. В дереве приложения нажмите кнопку **Project**.

Появится окно **Project Editor**.

2. Нажмите кнопку **Properties**.

Появится диалог **Global Options**.

3. В поле **Target OS** (операционная система) выделите Windows — 32-bit.

4. Нажмите кнопку **OK** в **Global Options** и окнах **Project Editor**.

Это все, что требуется, чтобы сделать web-пригодным имеющееся приложение!

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компиляции.

3. Откройте Windows Explorer (Проводник) (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\CW-20\Examples\WebTutor в директорию C:\CWICWEB\EXEC\ WebTutor.

5. Скопируйте все файлы пиктограмм (*.ICO) из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\Public.

Эти пиктограммы используются на кнопках панели инструментов и управляющих элементах Дерева. Они должны быть помещены в директорию \Public в том порядке, в котором броузер будет их изображать.

Запуск приложения

1. Стартуйте брокер приложения двойным щелчком на CWBrokrl.exe (или CWBroker.exe, если у вас полная версия брокера приложения) в директории C:\CWICWEB\.

Важно: Мы будем использовать исполняемую версию брокера приложения. ISAPI-версия работает похожим образом, лишь с незначительными различиями.

2. Стартуйте ваш броузер.

3. Теперь стартуйте приложение WebTree.exe в броузере. (http://localhost/exec/webtutor/webtree.exe.0)

Проверка приложения

Вы должны были заметить, что это приложение по виду несколько отличается от предыдущего. Оно использует панель инструментов и не использует меню. Это обычный интерфейс для приложений web.

1. Щелкните по кнопке **Orders**.

В броузере появляется «окно» **Browse Customer Orders** (просмотр заказов покупателей). Окно содержит **Дерево уп-**

равляющих элементов и две кнопки: для **Expand All** и **Contract All** («Раскрыть Все» и «Сжать все»).

2. Щелкните по кнопкам **Expand All** и **Contract All** и отследите, что будет происходить. Заметьте, что раскрытие и сжатие дерева обновляет всю страницу.

Для оптимизации этого процесса мы будем использовать те же приемы частичного обновления.

3. Выйдите из приложения (нажатием синей **X**).

Оставьте ваш броузер открытым с отображенной страницей рестарта. Вы будете этим пользоваться для перезапуска вашего приложения после внесения изменений.

Переопределение действия по умолчанию

1. Выделите процедуру **BrowseCustomer**, затем нажмите кнопку **Properties**.

Появится окно **Procedure Properties**.

2. Нажмите кнопку **Internet Options**.

3. Выберите закладку **Controls**.

4. Выделите управляющую кнопку в списке **Individual Control Options** (индивидуальные параметры управляющего элемента).

5. Нажмите кнопку **Properties**, затем выберите закладку **Events**.

6. Выделите событие **Accepted**, затем нажмите кнопку **Properties**.

7. Пометьте признак **Override default action** (переопределить действие по умолчанию), затем выберите **Partial page refresh** (частичное обновление страницы) из выпадающего списка.

8. Нажмите кнопки **OK** в окнах **Events** и **Individual Overrides**.

9. Выделите управляющий элемент — кнопку (**Contract**) в списке **Individual Control Options**.

10. Нажмите кнопку **Properties**, затем выберите закладку **Events**.

11. Выделите событие **Accepted**, затем нажмите кнопку **Properties**.

12. Пометьте признак **Override default action**, затем выберите **Partial page refresh** (частичное обновление страницы) из выпадающего списка.

13. Нажимайте кнопки **OK** во всех окнах до тех пор, пока не вернетесь в дерево приложения (4 раза).

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение снова готово к размещению.

2. Нажмите кнопку **OK** в окне компиляции.

3. Откройте Windows Explorer (Проводник) (или Windows NTExplorer).

4. Скопируйте Weborder.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\EXEC\WebTutor.

5. Скопируйте все файлы пиктограмм (*.ICO) из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\Public.

На этот раз вам нужно только разместить приложение, пиктограммы не изменились.

Давайте выполним приложение и посмотрим, как внесенные нами изменения повлияли на его работу.

Заметьте разницу

1. Переключитесь назад в ваш броузер.

2. Стартуйте приложение через броузер щелчком над гиперсвязью **Restart**.

3. Снова щелкните по кнопке **Orders**.

4. Щелкните по кнопкам **Expand All** и **Contract All** и посмотрите, что происходит теперь.

Вы должны заметить, что теперь дерево переотображает данные Дерева без пересылки всей страницы.

5. Выйдите из приложения.

Дополнительная техника программирования для Web

Изложенной информации о создании Web-приложений и преобразования существующих Clarion-приложений в Web-приложения достаточно для того, чтобы публиковать приложения баз данных в Интернет. Но это не полная информация о возможностях Internet Connect.

Переменные, используемые для координации в межзадачном пространстве

Добавление процедуры входа в систему

1. В дереве приложения установите курсор на процедуре **main** и нажмите кнопку **Properties** (свойства).

Появится окно **Procedure Properties**.

2. Нажмите кнопку **Embeds**.

Появится окно **Embedded Source** (код-вставка).

3. Подсветите строку **WindowManager Executable Code Section — Init-(),BYTE**.

4. Нажмите кнопку **Insert**.

Появится окно **Select Embed Type** (выберите тип вставки).

5. Подсветите строку **Call a Procedure** и нажмите кнопку **Select**.

6. В поле **Procedure to call** (вызываемая процедура) наберите **Login Window** и нажмите кнопку **OK**.

7. Установите значение **Priority** равным **1**.

Это обеспечит **LoginWindow** перед открытием окна.

8. Нажмите кнопку **Close** в окне **Embedded Source** и кнопку **OK** в окне **Pocedure Properties**.

Это добавит процедуру **LoginWindow** с типом **ToDo**.

Добавление окна входа в систему

1. В дереве приложения выберите процедуру **LoginWindow** и нажмите кнопку **Properties**. Появится окно **Select Procedure Type**.

2. Выберите **Window-Generic Window Handler** (общий обработчик окна) и нажмите кнопку **Select**.

Появится окно **Procedure Properties**.

3. Нажмите кнопку **Window**.

Появится окно **New Structure**.

4. Выделите **Window** и нажмите **OK**.

Разработка окна входа в систему

1. Выберите **Populate ⇨ Field**.

Появится **File Schematic**.

2. В списке с названием **Files**, расположенном в левой части, выделите **Global Data**, а затем в списке (справа) с названием **Fields** выберите **LoginName** и нажмите кнопку **Select**.

3. Для размещения текста приглашения введите имя и поля ввода, щелкните над окном кнопкой мыши.

4. Выберите **Populate ⇨ Control Template**.

Появится окно **Select Control Template**.

5. Выделите **CancelButton** и нажмите кнопку **Select**.

6. Для размещения кнопки **Cancel**, щелкните над окном кнопкой мыши.

7. Выберите **Populate ⇨ Control Template**.

8. Выделите **CloseButton** и нажмите кнопку **Select**.

9. Для размещения кнопки **Close** щелкните над окном кнопкой мыши.

10. Измените текст на кнопке **Close** на **OK**.

Добавление «Cookie»-кода для сохранения LoginName

1. Произведите двойной щелчок мышью на кнопке **OK** для доступа к пунктам вставки кода для этой кнопки.

2. Выделите пункт **Control Event Handling, After Generated Code, Accepted** и нажмите кнопку **Insert**.

3. Выберите шаблон (template) **SetCookie** и нажмите кнопку **Select**.

4. В поле **Cookie name** введите **LoginName**.

5. В поле **New Value** введите **LoginName** (или выберите глобальную переменную **LoginName** из **File schematic**.

6. Нажмите кнопку **OK**.

7. Нажмите кнопку **Close** в окне **Embedded Source**.

Добавление «Cookie»-кода для получения LoginName

1. Произведите двойной щелчок мышью над окном для получения доступа к точкам вставки встроенного кода для этого окна.

2. Выделите точку вставки кода **Accept Loop, Before CASE EVENT() handling** (цикл обработки оператора **Accept**, перед предложением **CASE EVENT()**) и нажмите кнопку **Insert**.

3. Выделите **Source** и нажмите кнопку **Insert**.

4. Наберите следующий исходный код:

```
IF EVENT() = Event:NewPage
!Если web-страница новая
LoginName = Broker.Http.GetCookie('LoginName')
!Получить cookie
DISPLAY
END
      IF LoginName
      !Если значение было установлено
            POST(Event:CloseWindow)
            ! закрыть окно
      END
```

Этот код «получает» cookie когда окно активно. Если код успешно находит cookie и устанавливает переменную **LoginName**, то он закрывает окно (до того как пользователь увидит его).

Это означает что пользователь должен регистрироваться при входе только один раз, а после этого сервер «распознает» пользователя при следующих входах в систему.

5. Покиньте редактор исходных текстов и сохраните изменения.

6. Нажмите кнопку **Close** в окне **Embedded Source**.

7. Выйдите из форматера окна (**Window Formatter**) и сохраните изменения.

8. Нажмите кнопку **OK** в окне **Procedure Properties**.

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer (проводник) (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\.

Запуск приложения

1. Запустите **Application Broker** двойным щелчком на файле CWBrokr1.exe или CWBroker.exe (если у вас полная версия Application Broker) из директории C:\CWICWEB\.

2. Запустите ваш броузер.

3. Запустите в броузере приложение **WebTree**. (http://localhost/exec/webtutor/webtree.exe)

Проверка приложения

Вы запускаете приложение в первый раз. Вам выдается предложение ввести имя для регистрации в системе. При следующем запуске предложение ввести имя не будет выдаваться, так как система прочитает cookie и значение глобальной переменной будет установлено равным значению cookie.

1. Введите имя, когда появится экран **Login** и нажмите **OK**.

2. Выйдите из приложения.

3. Перезапустите в броузере приложение **WebTree**.

Заметьте, что при втором запуске приглашение зарегистрироваться в системе не появляется.

4. Выйдите из приложения.

Покиньте ваш броузер, оставив открытой страницу перезапуска. Вы будете использовать это для перезапуска приложения после ввода изменений.

Встроенный HTML

Одна из наиболее мощных возможностей Internet Developer's Kit — это способность вставлять HTML-код в

7 Зак. 3215

HTML-страницы, порождаемые web-приложением. Когда вы встраиваете HTML-код (используя специальные точки для вставки, добавленные шаблоном Global Internet Application Template), он вставляется в определенные места в HTML, возвращаемый в броузер, выполняющий приложение.

Добавление динамического HTML, используя переменную

Мы написали код, необходимый для установки, восстановления входного имени пользователя и записи его в глобальную переменную. Теперь мы обеспечим отображение этого имени на web-странице ниже HTML-представления окна.

1. В дереве приложения выделите процедуру **Main** и нажмите кнопку **Properties**.

Появится окно **Procedure Properties**.

2. Нажмите кнопку **Embeds**.

Появится окно **Embedded Source**.

3. Выделите пункт для вставки кода **Internet-Before the Closing </BODY> tag**, затем нажмите кнопку **Insert**.

Появится окно **Select Embed Type**.

4. Выделите **Dynamic HTML** и нажмите кнопку **Select**.

5. В поле **Dynamic Text** наберите следующее:

```
'<<P>' & CLIP(LoginName) & 'is logged in <</P>'
```

6. Нажмите кнопку **OK** в окне шаблона.

7. В окне **Embedded Source** нажмите кнопку **Close**, затем в окне **Procedure Properties** кнопку **OK**.

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\.

Проверка приложения

1. Перезапустите в броузере приложение WebTree (щелкните по гиперсвязи **Restart**).

Если вы уже запускали приложение на этой мащине, то у вас не появится окно регистрации. Вместо этого сервер прочитает ваш «cookie» и занесет это значение в глобальную переменную **LoginName**. Переменная **LoginName** теперь будет отображаться на web странице ниже кнопок панели инструментов.

2. Выйдите из приложения.

Покиньте ваш броузер, оставив открытой страницу перезапуска. Вы будете использовать это для перезапуска приложения после ввода изменений.

Давайте произведем еще несколько изменений в приложении, используя встраиваемый HTML.

Добавление статического HTML

Ранее мы добавляли HTML-код, который составлялся с использованием комбинации текста и переменных. Далее мы будем использовать шаблон Static HTML для добавления HTML-кода, который будет оставаться статическим.

Мы будем использовать это для добавления ссылки (**Link**) в конце страницы, которая позволит пользователям отправлять E-mail для Web-мастера с комментариями или вопросами к приложению.

1. В дереве приложения выделите процедуру **Main**, затем нажмите кнопку **Properties**.

Появится окно **Procedure Properties**.

2. Нажмите кнопку **Embeds**.

Появится окно **Embedded Source**.

3. Выделите пункт для вставки кода **Internet-Before the Closing </BODY> tag** и нажмите кнопку **Insert**.

Появится окно **Select Embed Type** (выберите тип вставки).

4. Выделите **Static HTML** и нажмите кнопку **Select**.

5. В поле **HTML to Insert** наберите следующее:

```
<P> <A HREF="mailto:nobody@topspeed.com">
Comments?</A></P>
```

6. Нажмите кнопку **OK** в окне шаблона.

7. Нажмите кнопку **Close** в окне **Embedded Source** и кнопку **OK** в окне **Procedure Propreties**.

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\.

Проверка приложения

1. Перезапустите в броузере приложение WebTree (щелкните по гиперсвязи **Restart**).

Вы заметите новую связь (**Link**). Если вы щелкните по связи, то ваш броузер откроет вашего E-mail «клиента») с новым сообщением с предопределенным адресом.

2. Покиньте ваш броузер, оставив открытой страницу перезапуска. Вы будете использовать это для перезапуска приложения после ввода изменений.

Добавление условного HTML в исходный код Clarion

Третьим методом вставки HTML-кода в ваши web-страницы является использование во вставках исходного кода метода **Target.Writeln**. Это позволит вам использовать исходный код Clarion для написания HTML-кода. Одним из преимуществ использования исходного кода Clarion является возможность управлять тем, какой код HTML вы хотите написать. Другими словами, вы можете использовать управляющие структуры языка Clarion для управления тем, что пишется. Вы можете написать одну или другую строку, используя структуры языка **IF..THEN..ELSE** или **CASE**.

Мы будем использовать этот прием, используя структуру **EXECUTE**.

1. В дереве приложения выделите процедуру **Main** и нажмите кнопку **Properties**.

Появится окно **Procedure Properties**.

2. Нажмите кнопку **Embeds**.

Появится окно **Embedded Source**.

3. Выделите пункт вставки кода **Internet-Before the Closing </BODY> tag** и нажмите кнопку **Insert**.

Появится окно **Select Embed Type**.

4. Выделите **Source**, затем нажмите кнопку **Select**.

5. В **Embedded Source editor** (редактор вставок кода) наберите следующий исходный код:

```
Str1"='<<A  HREF=http://www'
Str2"='.com"><<IMG  SRC="'
Str3"='i.BORDER=0><</A'

EXECUTE  RANDOM(1,5)
Target.Writeln(CLIP(Str1")&'topspeed'
&clip(Str2")&SELF/Files.GetAlias('1.GIF')&Str3")
Target.Writeln(CLIP(Str1")&'icetips'
&clip(Str2")&SELF/Files.GetAlias('2.GIF')&Str3")
Target.Writeln(CLIP(Str1")&'finatics'
&clip(Str2")&SELF/Files.GetAlias('3.GIF')&Str3")
Target.Writeln(CLIP(Str1")&'flpathers'
&clip(Str2")&SELF/Files.GetAlias('4.GIF')&Str3")
Target.Writeln(CLIP(Str1")&'flamarlins'
&clip(Str2")&SELF/Files.GetAlias('5.GIF')&Str3")
END
```

6. Выйдите из редактора и сохраните изменения.

7. Нажмите кнопку **Close** в окне **Embedded Source** и **OK** в окне **Procedure Properties**.

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer(Проводник) (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\.

5. Скопируйте GIF файлы (*.gif) из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\EXEC\Public.

Проверка приложения

1. Перезапустите в броузере приложение WebTree (щелкните по гиперсвязи **Restart**).

Вы заметите новое изображение и связь (**Link**). Каждый раз при запуске приложения будет показываться реклама выбираемая по случайному закону.

2. Покиньте ваш броузер, оставив открытой страницу перезапуска. Вы будете использовать это для перезапуска приложения после ввода изменений.

Защита операции подгрузки с помощью Splash-окна

Для того, чтобы запустить приложение, приспособленное для Web, броузеру клиента должна быть доступна JSL (Java Support Library). Начинающие пользователи должны подгрузить Clarion.CAB (для Microsoft Internet Explorer) или Clarion.ZIP (для Netscape). Для большинства броузеров JSL подгружается единожды и сохраняется в кэше (до тех пор, пока пользователь не очистит его). Хотя JSL чрезвычайно компактна для той функциональности, какую она обеспечивает, ее подгрузка займет несколько минут. Имея это ввиду, мы будем использовать всплывающее окно для предупреждения начинающего пользователя о том, что идет подгрузка. Размещая кнопки Java в этом окне, мы можем воспрепятствовать работе пользователя до тех пор, пока не загрузится JSL и кнопки Java не будут инициализированы.

1. В **Application Tree** подсветите **Main** процедуру и нажмите клавишу **Properties**.

Откроется окно **Procedure Properties**.

2. Нажмите клавишу **Embeds**.

Откроется окно **Embedded Source**.

3. Подсветите **WindowManagerExecutable Code Section — Init-(),BYTE**.

4. Нажмите клавишу **Insert**.

Откроется окно **Select Embed Type**.

5. Выделите подсветкой **Source**, нажмите клавишу **Select**.

6. В редакторе **Embedded Source** введите следующее:

```
IF WebServer.Active THEN Splash
```

Это позволит процедуре **Splash** быть вызванной, когда приложение исполняется в Web.

7. Установите **Priority** равным единице.

8. Убедитесь, что эта вставка внесена в список перед вызовом процедуры **LoginWindow**, использующей клавиши «вверх» и «вниз».

Это обеспечивает то, что **Splash** процедура вызывается перед открытием других окон.

9. Нажмите клавишу **Button** в окне **Embed Source**.

10. Нажмите клавишу **Procedure**.

Откроется окно **Procedure**.

11. Подсветите **Splash**, нажмите кнопку **OK**.

Это связывает **Splash**-процедуру с **Main**-процедурой в **Application Tree**.

Последнее необходимо, если ваше приложение используется в **LocalMAPs**.

Переименование BUTTON в Java Button

Окно **Splash** содержит некоторый текст, клавишу и управление **IMAGE**. К клавише BUTTON «подвязан» управля-

ющий шаблон **CloseButton** с текстом, измененным на **Continue**. Так как кнопка создана как HTML кнопка по умолчанию, вы будете специфицировать ее иначе. Так как мы хотим, чтобы это была Java кнопка, она не будет доступна конечному пользователю до тех пор, пока JSL будет подгружаться.

1. В Application Tree подсветите **Splash** процедуру, нажмите кнопку **Properties**.

2. Нажмите кнопку **Internet Options**.

3. Выберите закладку **Controls**.

4. Подсветите **?Close** в списке **Individual Control Overrides**, нажмите кнопку **Properties**.

5. Выберите закладку **Classes**.

6. Пометьте признак **Override default Class**, выберите **WebJavaButtenClass** из выпадающего списка **ClassName**.

7. Нажмите **OK**.

Оставьте открытым окно **Internet Options**. Мы будем использовать его далее.

Центрирование поля Image в окне Splash

Управляющий элемент **IMAGE** окна **Splash** располагается по центру горизонтали окна. Ниже будет показано, как добавить некоторый HTML код, обеспечивающий отцентрованное положение **IMAGE** во время исполнения в Web.

1. Подсветите **?Image** в списке **Individual Control Overrides** (индивидуальные настройки управления), затем нажмите кнопку **Properties**.

2. Выберите закладку **Tab**.

Это окно позволит вам ввести HTML код перед и после управляющего элемента.

Код будет влиять на управляющий элемент только во время исполнения в Web.

3. В поле **HTML before control** введите:

```
<CENTER>
```

4. В поле **HTML after control** введите:

```
</CENTER>
```

5. Нажмите кнопку **OK** во всех окнах (3 раза) и вернитесь в **Application Tree**.

Создание и размещение

1. Выберите **Project-Make** (или нажмите кнопку **Make** панели).

Ваше Web приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\Clarion4\Examples\WebTutor в директорию директорию C:\CWICWEB\EXEC\WebTutor.

Проверка приложения

1. Перезапустите в броузере приложение WebTree (щелкните по гиперсвязи **Restart**).

Вы можете обнаружить, что **Splash** окно появилось перед другим окном.

2. Выйдите из приложения.

Покиньте ваш броузер, оставив открытой страницу перезапуска. Вы будете использовать это для перезапуска приложения после ввода изменений.

Применение частичного обновления к управляющим элементам Update

В приложениях под Windows программисты часто вставляют код для изменения одного элемента управления, когда значение другого меняется. Например, вы могли вставить код для изменения суммы переменных, когда меняется количество переменных. Код, подобный этому, дерево приложений для Web (**Webtree**) имеет в процедуре **UpdateItems**. Встроенный код связан **EVENT:Accepted** на каждом элементе управления. Другими словами, когда пользователь изменяет значение в управляющем элементе или выбирает другой управляющий элемент щелчком мыши, код исполняется.

Когда приложение исполняется через Web, управляющий элемент **ENTRY** обрабатывается в броузере по умолчанию. Другими словами, нет взаимодействия между броузером и сервером приложений, пока вы не меняете вручную опции для этого элемента управления. Далее мы будем изменять действие для трех управляющих элементов, чтобы обеспечить исполнение кодовой вставки на сервере для **Event:Accepted** для этих элементов.

1. В Application Tree подсветите **UpdateItems** процедуру и нажмите клавишу **Proreties**.

Откроется окно **Procedure Properties**.

2. Нажмите клавишу **Internet Options**.

3. Выберите закладку **Controls**.

4. Подсветите **?Item:ProdCode** в списке **Individual Control Overrides**, затем нажмите клавишу **Properties**.

5. Выберите закладку **Events**.

6. Выделите подсветкой **Accepted**, нажмите клавишу **Select**.

7. Пометьте признак **Override default action**, выберите **Partial** page **refresh** из выпадающего списка **Action on Event**.

8. Нажмите **OK** во всех окнах, пока не вернетесь в окно **Internet Options** (2раза).

9. Повторите последние 5 шагов для **?ITEM:Quantity** и **?ITEM:Price**.

10. Нажмите **OK** во всех окнах до возвращения в **Application Tree** (2 раза).

Создание и размещение

1. Выберите **Project-Make** (или нажмите кнопку **Make** панели).

Ваше Web приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\Clarion4\Examples\WebTutor в директорию директорию C:\CWICWEB\EXEC\WebTutor.

Проверка приложения

1. Перезапустите в броузере приложение WebTree (щелкните по гиперсвязи **Restart**).

2. Нажмите кнопку **Orders**.

3. Нажмите кнопку **Expand All**.

4. Подсветите одну из тем (зеленая линия).

5. Нажмите кнопку **Change**.

6. Измените итог (количество) в поле **Quantity**, нажмите **Tab**.

Отметьте изменения **Extended Total**. Если вы меняли поле **Price** или **Product Code**, поле **Extended Total** также изменится.

7. Выйдите из приложения.

Покиньте ваш броузер, оставив открытой страницу перезапуска. Вы будете использовать это для перезапуска приложения после ввода изменений.

Ограничение доступа к процедуре

Ограничивать доступ к процедуре возможно, используя встроенную в броузер поддержку аутентификации и возможности шаблона Internet Procedure Extension по парольной защите. При вызове процедуры, защищенной паролем, на экране появляется окно аутентификации броузера. От вас не требуется создавать окно сбора информации для входа. Парольная защита основана на области, имени пользователя и пароле. «Область» — это защищенная процедура.

Броузер запрашивает у пользователя имя и пароль. Затем они посылаются в программу для проверки. Если программа принимает пароль (то есть возвращает **TRUE** из метода **WebWindow.Validate Password**), предъявляется новая страница, в противном случае броузер повторяет запрос. После трех попыток броузер выдает сообщение, информирующее пользователя о том, что доступ запрещен. Эта страница автоматически возвращает пользователя к последнему активному месту в программе.

Важно: если страница уже посещалась в текущем сеансе, броузер обычно обеспечивает имя и пароль без опроса пользователя. Это свойство встроено в большинство броузеров.

Существует несколько методов парольной защиты. Мы будем использовать более передовой метод — переопределим метод **WebWindow.Validate Password**.

Добавление файла проверки

1. В Дереве приложения выделите процедуру **Update-Products**, затем нажмите кнопку **Properties**.

Появляется окно **Procedure Properties**.

2. Нажмите кнопку **Files**.

Появляется **File Schematic**.

3. Выделите **Other Files**, затем нажмите кнопку **Insert**.

Появляется **Insert File**.

4. Выделите **Userlist**, затем нажмите кнопку **Select**.

5. Нажмите кнопку **OK** на окне **File Schematic**.

Добавление вызова пароля

1. Нажмите кнопку **Internet Options**.

2. Выделите закладку **Advanced**.

3. Установите флаг **Restrict access to this** procedure (ограничить доступ к данной процедуре).

4. Нажмите кнопку **OK**.

5. Нажмите кнопку **Embeds** (вставки).

Появляется окно **Embedded Source**.

6. Выделите точку вставки кода Internet — **Password validation Code Section**, затем нажмите кнопку **Insert**.

Появляется окно **Select Embed Type**.

Вводя код в точке вставки Internet — **Password validation Code Section**, вы переопределяете принятый по умолчанию метод для проверки пароля.

Эта точка вставки помещает код внутри метода с двумя параметрами: **Именем пользователя** и **Паролем**, которые получены от броузера. Вы должны будете возвратить TRUE, если пароль правильный, и FALSE — если неправильный.

Это позволяет вам просматривать информацию в файле или использовать другой метод, выбранный вами для проверки пароля.

7. Выделите **Source**, затем нажмите кнопку **Select**.

8. В **Embedded Source editor** напишите следующий код:

```
USE:UserID  -UseName
     Get(Userlist.Use:KeyUserID)
     !lookup  UserName  in  file
        IF  ERRORCODE()  THEN  RETURN(false)
        !Return  false  on  invalid  user
           IF  USE:UserPassword=Password
           !Check  the  password
              RETURN(True)
        ELSE
        RETURN(False)
END
```

9. Выйдите из редактора текста и сохраните изменения.

10. Нажмите кнопку **Close** на окне **Embedded Source** и кнопку **OK** на окне **Procedure Properties**.

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer (Проводник) (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\CW20\Examples\WebTutor в директорию C:\CWICWEB\EXEC\WebTutor.

Проверка приложения

1. Перезапустите в броузере приложение WebTree (щелкните по гиперсвязи **Restart**).

2. Нажмите кнопку **Products**.

3. Нажмите кнопку **Insert**, чтобы добавить новый продукт.

Появляется окно аутентификации броузера.

4. В поле **UserName** напишите **Fred**.

5. В поле **Password** напишите **Wilma**.

Введенные вами значения находятся в файле **Userlist**. Этот файл был создан заранее с двумя пользователями. Заметьте, что в этом приложении нет процедуры для редактирования этого файла. Это обычный способ обращения с паролем пользователя, при котором только у системного администратора есть разрешение добавлять пользователей. Можете создавать процедуры для изменения этого файла, как вы это сочтете нужным.

6. Выйдите из приложения.

Ограничение редактирования по месту

Шаблоны ABC в Clarion4 позволяют вам использовать редактирование по месту (Edit-In-Place) в простом элементе выбора. Это свойство, однако, не поддерживается во время исполнения через Web. В этом случае вы должны иметь отдельную форму для изменений. Это простой метод — альтернатива между редактированием по месту при локальном исполнении под Windows и вызовом формы при исполнении в Web.

Если вы используете редактирование по месту и уточняете процедуру шаблоном управления **BrowseBox**, вы уже выполнили две трети работы. Шаблон, сгенерировавший код, или вызывает отдельную процедуру обновления, или

осуществляет редактирование по месту в зависимости от значения **BRWn.AskProcedure**. Установите значение **BRWn.-AskProcedure** в ноль, и получите редактирование по месту; установите в единицу и получите процедуру обновления (update).

Для того, чтобы использовать редактирование по месту для локального Windows и видоизменить при исполнении в Web:

1. Выберите процедуру **Browse**, затем нажмите клавишу **Properties**.

2. В разделе **UpdateButton** окна **Procedure Properties** установите флаг **Use Edit in Place** (использовать редактирование по месту).

Отметьте, что процедура замены всегда уточняется.

Далее, задайте код, чтобы установить обновление при редактировании по месту, когда исполняется в Windows и вызовите form при исполнении через Web.

3. Нажмите кнопку **Button**.

Откроется окно **Embedded Source**.

4. Выделите точку вставки кода **Window Manager Method-Init-BYTE()**, затем нажмите кнопку **Insert**.

5. Выделите **Source**, нажмите кнопку **Select**.

6. В редакторе **Embedded Source** введите следующий код:

```
IF WebServer.Active
BRW1:AskProcedure=1
END
```

7. Выйдите из редактора и сохраните изменения.

8. Нажмите кнопку **Close** в окне **Embedded Source** и **OK** в окне **Procedure Properties**.

Оставьте значения в **Property** в качестве значений по умолчанию.

Создание и размещение

1. Выберите **Project ⇨ Make** (или нажмите кнопку **Make** на панели инструментов).

Ваше web-приложение готово к размещению.

2. Нажмите кнопку **OK** в окне компилятора.

3. Откройте Windows Explorer (Проводник) (или Windows NTExplorer).

4. Скопируйте WebTree.exe из директории C:\Clarion4\Examples\WebTutor в директорию C:\CWICWEB\EXEC\WebTutor.

Проверка приложения

1. Перезапустите в броузере приложение WebTree (щелкните по гиперсвязи **Restart**).

2. Нажмите кнопку **Products**.

3. Нажмите кнопку **Insert**, чтобы добавить новый продукт.

Появляется окно аутентификации броузера.

4. В поле **UserName** напишите **Fred**. В поле пароля введите **Wilma**. Отметьте, что появилась форма **Update Products**.

5. Выйдите из приложения.

6. Запустите приложение «под Windows».

7. Нажмите клавишу **Products**.

8. Нажмите **Insert**, чтобы добавить новый продукт.

Отметьте, что **Edit-In-Place** доступна.

Выйдите из приложения.

Итак, у вас теперь должно быть достаточно опыта, чтобы создавать крепкие web-приложения для базы данных.

Шаблоны Internet Builder Class

Шаблоны IBC состоят из одного шаблона расширения (Extention template) **Global Application**, процедурного шаблона (**Procedure template**) и нескольких кодовых шаблонов (**Code temlates**).

Шаблон Global Internet Application Extention Template автоматически добавляет к каждой процедуре приложения шаблон расширения (Extention template). Это позволяет превратить ваше приложение в Web-приложение за один шаг.

Комбинация глобальных и процедурных настроек обеспечивает возможность детального определения поведения системы на разных уровнях. Для определения установки для всего приложения используется глобальный параметр (**Global option**).

Определив параметр для процедуры, вы производите настройку только этой процедуры. Многие из глобальных и процедурных настроек одинаковы, разница только лишь в области действия настройки.

Global Internet Extention Template

Global Internet Extention (шаблон Интернет — расширение) превращает ваше Clarion-приложение в Интернет-приложение. Он прививает приложению способность генерировать динамический HTML, когда оно выполняется через Application Broker (брокер приложений).

Этот шаблон позволяет определять параметры, используемые при генерации HTML-представления ваших окон и отчетов.

Вдобавок, он автоматически добавляет Internet Procedure Extention к каждой процедуре в приложении и к каждой впоследствии добавляемой к приложению процедуре.

Это позволит переопределять для определенной процедуры многие из глобальных параметров.

Этот шаблон (Global Internet Extention) позволит вам определить особенности появления и поведения приложения при выполнении в Web. Параметры, которые вы здесь определите, по «природе», являются глобальными, поэтому они воздействуют на все процедуры приложения.

На процедурном уровне вы можете переопределить большую часть этих параметров, используя настройки шаблона Internet Procedure Extention. В дополнение, некоторые параметры могут определяться на уровне обработки управляющих элементов. Комбинации этих трех уровней достаточно для обеспечения гибкости при разработке.

Важно: Ни одна из этих настроек не работает при запуске из Windows (как исполняемый модуль).

Настройки страницы

При работе через Web текущее окно приложения отображается внутри страницы HTML (Web-страницы). Установки страницы позволяют вам определить цвет фона и фоновое изображение для HTML-страницы.

Отцентрируйте окно на странице (Center Window on Page)

Установите этот флажок для центрирования HTML-представления вашего окна внутри Web-страницы. Это добавляет к Web-странице признаки **HTML <CNTER></CENTER>**.

Цвет фона (Background color)

Вы можете определить цвет для web-страницы. Определите цвет (**Color**), числовой (константный) эквивалент цвета (**Equate**) или выберите цвет в диалоге **COLORDIALOG**, нажав на кнопку (...). По умолчанию цвет не установлен (эквивалент-**COLOR:NONE**). Это означает, что для web-страницы используется цвет, заданный по умолчанию.

Фоновое изображение (Background image)

Можно определить изображение для последующего использования в качестве фона для web-страницы. Определите имя файла с изображением или выберите его в диалоге **FILEDIALOG**, нажав кнопку (...). По умолчанию изображение отсутствует.

Настройки окна

При работе через Web текущее окно приложения представляется HTML <**TABLE**>. Это позволяет устанавливать такие параметры <**TABLE**>, как цвет фона и ширина обрамления. Код, сгенерированный шаблоном, вызывает для установки этих параметров метод **WebWindow.SetPageBackGround**.

Цвет фона (Background color)

Вы можете определить цвет для web-страницы. Определите цвет (**Color**), числовой (константный) эквивалент цвета (**Equate**), или выбрать цвет в диалоге **COLORDIALOG**, нажав на кнопку (...) . По умолчанию, цвет не установлен (эквивалент-**COLOR:NONE**). Это означает, что для web-страницы используется цвет, заданный по умолчанию.

Совет: Вы также можете установить цвета для отдельных частей окна, таких как панель инструментов.

Фоновое изображение (Background image)

Можно определить изображение для использования его в качестве фона для web-страницы. Определите имя файла с изображением или выберите его в диалоге **FILEDIALOG**, нажав кнопку (...). По умолчанию изображение отсутствует.

Совет: Фоновое изображение повторяется (**tiles**) внутри элемента HTML <**TABLE**>, представляющего окно приложения, столько раз, сколько позволяет его размер. Для

сохранения небольшого трафика используйте маленькие изображения.

Толщина обрамления окна (Window border width)

Определите толщину обрамления окна для вашего приложения. По умолчанию она равна 2, что создает тонкое обрамление. Для того, чтобы убрать обрамление, определите значение толщины обрамления равным 0. Сгенерированный шаблоном (template) код вызывает для установки данного свойства метод **WebWindow.SetBorderWidth**.

Помощь

Включение Помощи в Интернет-приложения (Enable Help for internet)

Установите этот флажок для разрешения связей от кнопок Help в вашем приложении. (Кнопка **Help** — кнопка с атрибутом **STD:Hlp**).

Если Помощь включена, то кнопка **Help** будет вызывать web-страницу, базирующуюся на **Help ID** данного окна. Этот документ (помощь) открывается в окне броузера с названием «**_HELP**», что будет вызывать открытие нового окна броузера, или если фрейм уже имеет это имя, то документ (помощь) отобразится внутри этого фрейма. Код, сгенерированный шаблоном (template), использует для установки заданных вами параметров методы **WebWindow.SetHelpDocument** или **WebWindow.SetHelpURL**. Вы должны создать соответствующие HTML-страницы.

URL документов помощи (URL of help documents)

Расположение HTML-файлов Помощи. Например, если ваши HTML-файлы Помощи располагаются в отдельной поддиректории.

Стиль окна помощи (Help Window Style)

Вы можете определить собственный стиль для окна помощи.

Идентификаторы Help Ids являются связями внутри основного документа (Help Ids links within a base document)

Установите этот флажок, если ваша помощь разработана как единый документ с внутристраничными ссылками. Если признак не установлен, то **Help-кнопки** ссылаются на отдельные HTML-страницы.

Документ Помощь (Help Document)

Основной документ, содержащий внутристраничные ссылки.

Поле доступно только если проставлен признак «**Идентификаторы Help Ids являются связями внутри основного документа**» (**Help Ids are links within a base document**).

Компоненты окна (Window Components)

Нажмите кнопку для определения появления оконных компонент (таких, как **TOOLBAR**, **MENU** и области **Caption** (заголовок)).

Управляющий элемент

В этой вкладке (**tab**) вы можете определить установки по умолчанию для генерации HTML-кода, обеспечивающего работу каждого управляющего элемента вашего приложения.

Совет: В добавление к настройкам, произведенным здесь, вы можете установить параметры управляющего элемента в Интернет-параметрах процедурного шаблона.

Если управляющий элемент отключен (If control disabled)

Параметр определяет, что отображать в броузере, когда управляющий элемент отключен. Предоставляется пото-

му, что большинство управляющих элементов HTML не поддерживают отключения. Это устанавливает свойство **WebWindow.DisableAction**. Возможны следующие варианты:

- **Hide** (по умолчанию) — Прячет отключенный элемент.
- Спрятать, если невозможно отключить (**Hide if cannot disable**) — Прячет отключенный элемент, если нельзя отключить его на web-странице. Большинство управляющих элементов HTML не могут быть отключены.
- Показать (**Show**) — Отображает отключенный элемент. Он появляется обычным образом (т.е. доступным для обработки), но изменения, произведенные в этом поле, никак не отразятся в приложении.

Выпадающие списки (Drop listboxes)
Заменить на Java невыпадающий список

Позволяет заменить выпадающий список на загружаемый постранично список Java. Используйте эту установку, когда необходимо в выпадающем списке отображать более одной колонки.

Листы (Sheets) Толщина обрамления (border width)

Определите толщину обрамления для управляющего элемента **SHEET**. По умолчанию она равна 2, что создает тонкое обрамление.

Для того, чтобы убрать обрамление, определите значение толщины обрамления равным 0. Это устанавливает свойство **WebWindow.SheetBorderWidth**.

Варианты (Options) Толщина обрамления (border width)

Определите толщину обрамления для управляющего элемента **Options**. Это употребимо только для **OPTIONs** с атрибутом **BOXED**. По умолчанию толщина равна 2, что со-

здает тонкое обрамление. Для того, чтобы убрать обрамление, определите значение толщины обрамления равным 0. Это устанавливает свойство **WebWindow.OptionBorder Width**.

Группы (Groups)
Толщина обрамления (border width)

Определите толщину обрамления для управляющего элемента **Groups**. Это употребимо только для **GROUPs** с атрибутом **BOXED**.

По умолчанию толщина равна 2, что создает тонкое обрамление. Для того, чтобы убрать обрамление, определите значение толщины обрамления равным 0. Это устанавливает свойство **WebWindow.GroupBorderWidth**.

Многодокументный интерфейс

Совет: Для управления определенными пунктами меню или панели инструментов используйте для переопределения пункты из Интернет Параметров процедуры **Frame**.

Меню Frame

Этот раздел определяет способ обработки меню приложения. Это позволяет Вам определить, какие глобальные параметры меню отобразятся в «дочерних» окнах.

Включать в «дочерние» окна (Include on Child Windows)

Выберите вариант из выпадающего списка. Возможные варианты:

- Все пункты меню (**All Menu Items**) — Все пункты меню появятся в «дочерних» окнах.
- Нет пунктов меню (**No Menu Items**) — Пункты меню не появятся в «дочерних» окнах.

Игнорировать код в цикле ACCEPT процедуры Frame (Ignore code in frame's ACCEPT loop)

Установите этот флажок для того, чтобы игнорировать код для обработки пунктов меню внутри цикла **ACCEPT** в процедуре **Frame**.

Если признак не установлен, то весь код выполняемый в цикле **ACCEPT** процедуры **Frame**, автоматически выполняется в «дочерней» процедуре.

Панель инструментов процедуры Frame (Frame Toolbar)

Этот раздел описывает способ обработки управляющих элементов Панели инструментов приложения. Это позволяет вам определять, какие глобальные управляющие элементы Панели инструментов высвечиваются в «дочерних» окнах.

Include on Child Window (Включать в «дочерние» окна)

Выберите вариант из выпадающего списка. Доступны следующие значения:

- **All Toolbar Items** — Все пункты панели инструментов появятся в «дочерних» окнах.
- **Standard Toolbar Only** — В «дочерних» окнах появятся только «стандартные» (**Standard**) пункты панели инструментов. Это кнопки, добавляемые шаблоном **FrameBrowse Control**.
- **No Toolbar Items** — В «дочерних» окнах не появятся пункты панели инструментов.

Ignore code in frame's ACCEPT loop (Игнорировать код в цикле ACCEPT для Frame)

Установите этот флажок для игнорирования в цикле **ACCEPT** (для **Frame**-процедуры приложения) кода обработки пунктов панели инструментов. Если признак не установлен, то код, включенный в цикл **ACCEPT** (для **Frame**), автоматически попадает в «дочернюю» процедуру.

Horizontal Pixels per Char

Ширина символа, выраженная в числе точек для расчета размеров при создании апплетов Java и изображений.

Vertical Pixels per Char

Высота символа, выраженная в числе точек для расчета размеров при создании апплетов Java и изображений.

Важно: Определяемые здесь значения влияют на общий вид генерируемой HTML-страницы. Например, увеличивая значение **Vertical Pixels per Char**, вы увеличиваете высоту клеток HTML-таблицы.

Data for grid snapping

Число точек, рассматриваемое перед перемещением управляющего элемента. Определите значения для координат **X** и **Y**.

Если управляющий элемент находится в пределах этих значений, то он не перемещается.

Page to return on exit

При желании, вы можете определить HTML-страницу для возврата после окончания работы программы. Код, созданный шаблоном, вызовет метод **WebServer.Init** для установки свойства **WebServer.PageReturnTo**.

Time out (seconds)

Величина, определяющая максимальное количество времени бездействия (в секундах), после которого происходит закрытие приложения. По умолчанию — 600 секунд (10 минут). Для установки свойства **WebServer.TimeOut** сгенерированный код вызывает метод **WebServer.Init**.

Subdirectory for pages

Директория, в которой приложение создает временные директории (временная директория создается для каж-

дого активного соединения) для записи динамических HTML и графических файлов. Это также директория, в которой размещаются графические файлы. Если вы разместите графику в этой директории, то не будет производится ее запись во временную директорию. По умолчанию установлена директория /PUBLIC. Сгенерированный код вызывает для установки этого свойства метод **WebFilesManager.Init**. Не стоит устанавливать это свойство в момент работы.

Classes Local to Application Broker

Здесь определено, что файлы библиотеки поддержки Java расположены в поддиректории /PUBLIC основной директории брокера приложений. Если у вас несколько серверов, то может появиться желание определить единственный источник, откуда будут задействоваться эти файлы. В этом случае вы должны убрать этот флажок и назначить URL для файлов поддержки Java (Java Support Library). Это устанавливает свойство **WebServer.JavaClassLibrary**.

Use Long Filenames

Установите этот флажок для того, чтобы разрешить создание на Web-сервере файлов с длинными именами.

Классы

Закладка **Classes** предлагает вам определить, какие классы (объекты) используют шаблоны (**Templates**) для выполнения различных задач и модули, содержащие определения классов. Этот подход дает вам возможность использовать столько классов, сколько вы хотите из библиотеки IBC и столько, сколько хотите — из своих собственных классов.

Для изменения класса для определенного пункта или переназначения класса выделите его в списке и нажмите кнопку **Properties** (свойства).

Глобальные параметры компоненты окна

Заголовок

Это область в верхней части «окна» на HTML-странице. Это часть, представляющая заголовок окна.

Include caption

Установите этот флажок для отображения заголовка (**Caption**). Если флажок не установлен, то заголовок не используется. Это устанавливает свойство **WebWindow.CreateCaption**.

Background color

Вы можете определить цвет для использования в области заголовка.

Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из **COLORDIALOG**, нажав на кнопку (...).

По умолчанию установлен темно-синий цвет (**Navy Blue**) (эквивалент **COLOR:Navy**). Если цвет здесь не определен, то используется тот цвет фона в окне, который вы определяли ранее в установках параметров для окна (если вы это делали). Если не определяли и окно приложения имеет атрибут **COLOR**, то этот цвет отображается в броузере. Для определения данного свойства сгенерированный код использует метод **WebCaption.SetBackground**.

Background Image

Вы можете определить изображение для использования его в области заголовка в качестве фона. Определите имя файла с изображением или выберите его, используя диалог **FILEDIALOG**, нажав кнопку (...).

По умолчанию изображение не используется. Для определения данного свойства сгенерированный код использует метод **WebCaption.SetBackground**.

Совет: Изображение, используемое для фона, повторяется столько раз, сколько позволяет его размер внутри области HTML <**TABLE**>, представляющей область заголовка окна.

Обеспечьте небольшой размер изображения для того, чтобы сохранить трафик.

Alignment

Вы можете управлять выравниванием текста в области заголовка.

Варианты для выбора — это выравнивание влево, вправо и по центру.

По умолчанию — по центру. Это устанавливает свойство **WebCaption.Alignment**.

Font Family name

Позволяет вам определять шрифт для отображения на экране.

Запомните, что броузер может отображать только те шрифты, которые установлены на компьютере клиента. Однако большинство операционных систем поддерживают замену шрифтов и будут использовать для отображения наиболее близкий шрифт. Код, сгенерированный по шаблону, использует для установки данного свойства метод **WebCaption.SetFont**.

Font size

При желании, вы можете определить размер шрифта, отображаемого в области заголовка. По умолчанию не установлен (выбирается тот размер, который по умолчанию ис-

пользует броузер). Для определения данного свойства сгенерированный код использует метод **WebCaption.SetFont**.

Font color

Вы можете определить цвет шрифта для области заголовка.

Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из **COLORDIALOG**, нажав на кнопку (...).

По умолчанию не установлен и используется цвет шрифта, по умолчанию используемый броузером.

Меню

Это область меню в верхней или боковой части «окна» в HTML-странице.

Background color

Вы можете определить цвет для использования в области меню.

Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из **COLORDIALOG**, нажав на кнопку (...).

Если цвет здесь не определен, то используется тот цвет фона в окне, который вы определяли ранее в установках параметров для окна (если вы это делали). Если не определяли, и окно приложения имеет атрибут **COLOR**, то этот цвет отображается в броузере. Для определения данного свойства сгенерированный код использует метод **WebMenuBar.SetBackground**.

Background Image

Вы можете определить изображение для использования его в области меню в качестве фона. Определите имя файла с изображением или выберите его, используя диалог **FILEDIALOG**, нажав кнопку (...).

По умолчанию, изображение не используется. Для определения данного свойства сгенерированный код использует метод **WebMenuBar.SetBackground**.

Совет: Изображение, используемое для фона повторяется столько раз, сколько позволяет его размер внутри области HTML <TABLE>, представляющей область заголовка окна.

Обеспечьте небольшой размер изображения для того, чтобы сохранить трафик.

Alignment

Вы можете управлять расположением меню. Возможны следующие варианты:

- **Above Toolbar** (по умолчанию)
- **Left of Window**
- **Below Toolbar**.

Когда вы используете **Above Toolbar**, меню растягивается горизонтально через верхнюю часть HTML страницы. Когда вы используете **Below the ToolBar**, меню растягивается горизонтально через HTML-страницу под областью **Панели инструментов** (Toolbar arear). Когда вы используете **Left of Window**, меню располагается вертикально слева от <TABLE>, представляющего окно приложения.

Панель инструментов

Это область панели инструментов, находящаяся в верхней части «окна» в HTML-странице (под областью заголовка).

Background color

Вы можете определить цвет для использования в области панели инструментов. Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из COLORDIALOG, нажав на кнопку (...). Если цвет

здесь не определен, то используется тот цвет фона в окне, который вы определяли ранее в установках параметров для окна (если вы это делали). Если не определяли, и окно приложения имеет атрибут **COLOR**, то этот цвет отображается в броузере. Для определения данного свойства сгенерированный код использует метод **WebToolBar.SetBackground**.

Background Image

Вы можете определить изображение для использования его в области панели инструментов в качестве фона. Определите имя файла с изображением или выберите его, используя диалог **FILEDIALOG**, нажав кнопку (...). По умолчанию, изображение не используется. Для определения данного свойства сгенерированный код использует метод **WebToolBar.SetBackground**.

Create extra close button

Определяет, в каких случаях обеспечивать для окна кнопку **Close**. Эта кнопка — дополнение ко многим другим кнопкам окна. Она предназначена для замены системной кнопки **Close**, автоматически обеспечиваемой Windows, но не броузером. Если во всех ваших окнах уже есть кнопка закрытия, у вас нет необходимости определять еще такую же. Возможны следующие варианты выбора:

- **Never** — Никогда не создавать дополнительных кнопок **Close**
- **If window has system menu** — Создает кнопку **Close** тогда, когда **WINDOW** имеет атрибут **SYSTEM** и нет других кнопок.
- **If window has system menu** — Создает кнопку **Close** тогда, когда **WINDOW** имеет атрибут **SYSTEM**.
- **Always** — Всегда создает кнопку **Close**.

Image for close

Определяет пиктограмму отображения для кнопки **Close**. Определите имя файла с пиктограммой или выберите файл в диалоге **FILEDIALOG**, нажав кнопку (...). По умолчанию определен файл EXIT.ICO (маленькая голубая X) (поставляется с Clarion).

Клиентская область

Это область «окна» на странице HTML, представляющая клиентскую область приложения.

Background Color

Вы можете задать цвет, который будет использоваться для вашей клиентской области. Определите цвет (**Color**), цветовой эквивалент (**Color equate**) или выберите цвет нажатием кнопки (...) в **COLORDIALOG**. Если здесь цвет не был определен, то будет использоваться тот цвет, который был задан ранее в установках параметров для окна (если вы это сделали). Если вы не определили цвет ни там, ни здесь, и окно приложения имеет атрибут **COLOR**, то этот цвет и будет отображаться в броузере. Для определения данного свойства сгенерированный код использует метод **WebClientArea.SetBackground**.

Background image

Вы можете определить изображение, которое будет отображаться на экране как фон для клиентской области вашего приложения. Задайте имя файла изображения или выберите файл из **FILEDIALOG** нажатием кнопки (...). По умолчанию, изображения нет. Для определения этого свойства сгенерированный код использует метод **WebClientArea.SetBackground**.

Совет: Изображение, используемое для фона, повторяется столько раз, сколько позволяет его размер внутри

области HTML <TABLE>, представляющей область заголовка окна.

Обеспечьте небольшой размер изображения для того, чтобы сохранить трафик.

Переопределение классов (Class Overrides)

Override default class (Переопределение классов по умолчанию)

Чтобы переопределить класс IBC, установите этот флажок и определите **Class Name**, файл **Header** (INC.) и файл **Implementation** (.CLW) в полях, описанных ниже.

Class Name (Имя класса)

Определите имя класса, который будет использоваться, или имя класса по умолчанию, если Вы хотели бы переопределить класс по умолчанию.

Header file (файл заголовка)

Определите файл заголовка (файл, содержащий объявления класса) или выберите файл из **FILEDIALOG** нажатием кнопки (...).

Implementation file (файл реализации)

Определите файл реализации (файл, содержащий определения класса или исходный код) или выберите файл из **FILEDIALOG** нажатием кнопки (...).

Распределенный шаблон процедуры для Internet

Этот шаблон позволяет вам доработать появление и поведение процедуры для выполнении в Web. Установки, которые вы здесь определяете, локальны по природе, то есть они распространяются только на эту процедуру. Чтобы из-

менить глобальные установки (**Global Settings**), нажмите кнопку **Global** в генераторе приложений, затем — кнопку **Extensions**, и поменяйте установки для Internet-расширения приложения.

Чтобы поменять установки, нажмите кнопку **Internet Options** в окне **Procedure Properties**.

Важно: ни одна из этих установок не влияет на выполнения вашего приложения, когда оно работает локально как исполняемый модуль Windows.

Установки страницы

При работе под Web окно приложения изображается внутри страницы HTML (страницы web).

Установки страницы дают вам возможность задать цвет фона и изображение фона для страницы HTML. Для обеспечения этих свойств сгенерированный код использует метод **WebWindow.SetPageBackground**.

Override Global Settings (переопределение глобальных установок)

Чтобы переопределить установки страницы в шаблоне **Internet Application Global Extension**, установите этот флажок. Установка этого флажка делает доступными для определения другие установки.

Center Window on Page (центрировать окно на странице)

Установите этот флажок, для центрирования HTML-представления вашего окна внутри web-страницы. Это добавляет к web-странице признаки <CENTER></CENTER> HTML.

Background Color (цвет фона)

Вы можете задать цвет, который будет использоваться для web-страницы. Определите цвет (**Color**), цветовой экви-

валент (**Color equate**), или выберите цвет нажатием кнопки (...) в **COLORDIALOG**.

По умолчанию цвета нет (эквивалентно **COLOR:NONE**). Это значит, что используется цвет, установленный по умолчанию в броузере.

Background image (фоновое изображение)

Вы можете определить изображение, которое будет отображаться на экране как фон для web-страницы. Задайте имя файла изображения или выберите файл из **FILEDIALOG** нажатием кнопки (...). По умолчанию изображения нет.

Установки окна

При работе в Web окно приложения представлено посредством HTML <**TABLE**>. Поля ввода на этой закладке позволяют вам определить появление порции «окна» страницы HTML, которая отображается при выполнении приложения в Web.

Override Global Settings (переопределение глобальных установок)

Чтобы переопределить установки страницы в шаблоне **Internet Application Global Extension**, установите этот флажок. Установка этого флажка делает доступными для определения другие установки.

Background Color (цвет фона)

Вы можете задать цвет, который будет использоваться для окна вашего приложения. Определите цвет (**Color**), цветовой эквивалент (**Color equate**), или выберите цвет нажатием кнопки (...) в **COLORDIALOG**. По умолчанию цвета нет (эквивалентно **COLOR:NONE**), это значит, что используется цвет фона окна приложения. Для обеспечения этого свойства сгенерированный код использует метод **WebWindow.SetBackground**.

Background image

Вы можете определить изображение, которое будет отображаться на экране как фон для окна вашего приложения. Задайте имя файла изображения или выберите файл из **FILEDIALOG** нажатием кнопки (...). По умолчанию изображения нет. Для установки этого свойства сгенерированный код использует метод **WebWindow.SetBackground.**

Совет: Фоновое изображение повторяется (tiles) внутри элемента HTML <**TABLE**>, представляющего окно приложения, столько раз, сколько позволяет его размер. Для сохранения небольшого трафика, используйте маленькие изображения.

Window border width (ширина границы окна)

Определите ширину границы для окна вашего приложения. По умолчанию это 2, что создает тонкую границу. Чтобы границы не было, задайте границу ширины 0.

Помощь

Override Global Settings (переопределение глобальных установок)

Чтобы переопределить установки **Help** в шаблоне **Internet Applica tion Global Extension**, установите этот флажок. Установка этого флажка делает доступными для определения другие установки.

URL of help documents

Основное место размещения файлов HTML для вашей Помощи.

Например, ваши HTML-файлы помощи размещены в отдельной поддиректории.

Help Window Style (стиль окна Помощи)

При желании вы можете определить стиль окна Помощи.

Help Ids are links within a base document (идентификаторы Помощи являются ссылками внутри основного документа)

Установите этот флажок, если ваша Помощь задумана как единственный документ с внутристраничными ссылками (**anchors**).

Если он не установлен, кнопки Помощи ссылаются на индивидуальные HTML-страницы.

Help Document (документ Помощи)

Основной документ, содержащий внутристраничные ссылки. Это поле доступно, только если установлен флаг **Help Ids are links within a base document**.

Window Components (компоненты окна)

Нажмите эту кнопку, чтобы задать установки, определяющие появление компонентов окна (то есть, **TOOLBAR**, **MENU** и области **Caption**). Эти установки переопределяют любые соответствующие глобальные установки.

Return if launched from browser (возврат, если запущен из броузера)

Закрывает процедуру, когда она выполняется в web. Эффективно делает процедуру недоступной при работе в Web.

Управляющие элементы

To Override Global settings

Чтобы переопределить установки управляющего элемента в шаблоне **Internet Application Global Extension**, установите флажок слева от описания параметра. Установка этого флажка делает доступными для определения другие установки.

General
If control disabled

Определяет, что будет отображаться в броузере, когда управляющий элемент окна отключен. Такая возможность обеспечена из-за того, что большинство управляющих элементов HTML не поддерживают отключение. Это устанавливает свойство **WebWindow. DisabledAction**.

Возможности для выбора:

- **Hide** (по умолчанию) — Прячет все отключенные управляющие элементы.
- **Hide if cannot disable** — Прячет любой отключенный управляющий элемент, когда его нельзя отключить на web-странице. Большинство управляющих элементов HTML не могут быть отключены.
- **Show** — Показывает все отключенные управляющие элементы. Они нормально появляются (то есть, будут появляться как действующие), но изменения, сделанные в них, не будут действенны в разрабатываемом приложении.

Drop listboxes
Replace with Java non-drop list

Это позволяет вам заменить выпадающий список на странично загружаемый список (**Listbox**) Java. Используйте этот параметр, если ваш выпадающий список требует для отображения больше одной колонки.

Sheets Border width

Задайте толщину рамки для управляющих элементов **SHEET**. По умолчанию, это 2 — для тонкой границы. Определите толщину равной 0, чтобы рамки не было. Здесь устанавливается свойство **WebWindow.SheetBorderWidth**.

Options Border width

Задайте толщину рамки для управляющих элементов **OPTION**.

Применяется только для элементов **OPTION** с атрибутом **BOXED**.

По умолчанию это 2 — для тонкой границы. Определите толщину равной 0, чтобы рамки не было. Здесь устанавливается свойство **WebWindow.OptionBorderWidth**.

Groups

Border width

Задайте толщину рамки для управляющих элементов **GROUP**.

Применяется только для элементов **GROUP** с атрибутом **BOXED**. По умолчанию это 2, что создает тонкую рамку. Определите толщину равной 0, чтобы рамки не было. Здесь устанавливается свойство **WebWindow.GroupBorderWidth**.

Individual Control Overrides

В этом месте вы можете переопределить появление и поведение каждого управляющего элемента в окне. Выделите элемент, подлежащий изменению, и нажмите кнопку **Properties**. Смотрите раздел **Individual Overrides for a Control**.

Многодокументный интерфейс

Здесь определяется порядок обработки меню приложения (**Menu**) и панели инструментов (**Toolbar**).

Совет: для управления специфичными пунктами меню и панели инструментов установите переопределение для MDI в Интернет-параметрах процедуры **Frame**.

Merge Frame Menu

Установите этот флажок для подавления меню **Frame** при выполнении этой процедуры.

Merge Frame Toolbar

Установите этот флажок для подавления панели управления **Frame** при выполнении этой процедуры.

Для процедуры **Frame** имеются также дополнительные параметры.

Formatting (форматирование)
Override Global Settings

Чтобы переопределить установки для форматирования в шаблоне **Internet Application Global Extension**, установите этот флажок.

Установка этого флажка делает доступными для определения другие установки.

Horizontal Pixels per Char

Число точек, принимаемое как ширина символа при вычислении размера для создания апплетов и изображений Java.

Vertical Pixels per Char

Число точек, принимаемое как высота символа при вычислении размера для создания апплетов и изображений Java.

Delta for grid snapping

Число точек, которое учитывается перед перемещением управляющего элемента. Задайте значение для X и значение для Y.

Каждый раз при перемещении, пока элемент находится внутри этого интервала, он не перемещается.

Важно: заданные числа влияют на общий вид созданной страницы HTML.Например, увеличение значения **Vertical Pixels** per **Char** делает ячейки таблицы HTML более высокими.

Security

Transfer over a secure connection

Если установлен этот флажок, данные пересылаются с использованием **Secure Socket Layer** (SSL). Это обеспечивает безопасные транзакции на уровне процедуры. Помните, что шифрование заметно замедляет выполнение программы. Вам следует обеспечивать безопасность только для процедур, которые пересылают важные данные.

Важно: Эта возможность требует инсталляции секретной версии Брокера приложения.

Restrict Access to this procedure

Установите этот флажок для парольной защиты процедуры и разрешения двух нижеописанных полей.

Password

Задайте пароль или выберите переменную из схемы связей файлов (**file schematic**) нажатием кнопки (...). Постоянный пароль обеспечивает простую защиту.

Индивидуальные переопределения настроек для управляющего элемента

Состав пунктов для переопределения настроек управляющих элементов основывается на типе управляющего элемента и его атрибутах. Каждое переопределение описывается здесь с пометками о возможных условных параметрах.

Override Global Settings (переопределить глобальные установки)

Установите этот флаг слева от названия для переопределения установок управляющего элемента в шаблоне **Internet Application Global Extention template**. После установки этого флага откроются другие параметры настройки.

Отображение

If control is disabled (если элемент отключен)

Определяет, что нужно отображать в броузере, когда управляющий элемент окна отключен. Этот параметр определяется потому, что большинство элементов HTML не поддерживают отключения. Этим устанавливается свойство **IC:CurControl. DisabledAction**. Возможны следующие варианты:

- **Hide** (по умолчанию) (спрятать) — Прячет любой отключенный управляющий элемент.
- **Hide if cannot disable** (спрятать, если невозможно отключить) — Прячет любой отключенный управляющий элемент, когда невозможно его отключение.
- **Show** (показать) — Отображает любой отключенный элемент. Он появляется как обычно (то есть доступный для редактирования), но все изменения, сделанные в нем, не имеют эффекта в приложении.

Hide if launched from browser (спрятать, если запущен из броузера)

Установите этот флаг для того чтобы спрятать управляющий элемент при работе приложения в web. Это позволяет вам отключать отображение некоторых данных или ограничивать функциональность для web-версии, не удаляя ее из Windows-версии.

Autospot Hyperlinks (Автораспознаваемые гиперсвязи)

Этот признак доступен для элементов **List** и **STRING**. Если он установлен, то отображаются те данные, которые содержат корректную гиперсвязь (то есть начинающиеся с http:, https:, ftp:, malto:, news:, telnet:, wais: или gopher:) для перехода по ссылке.

Allow dynamic updates (позволить динамическое обновление)

Этот параметр доступен для элементов типа **STRING**. Если он установлен, то строковый управляющий элемент создается на HTML-странице как строковый управляющий элемент Java, который обновляется при каждом частичном обновлении страницы.

Важно: Элемент типа **STRING** с переменной в качестве атрибута для **USE** автоматически становится элементом Java и не нуждается в установке этого признака. Он применим только для статических **STRING**-элементов, которые изменяются через присваивание свойств (например, **?String1{PROP:Text}='New Text'**).

Image Options (параметры изображения)
Update Image dynamically (обновлять изображение динамически)

Этот признак доступен для элементов типа **IMAGE**. Если он установлен, то управляющий элемент создается на HTML-странице как управляющий элемент Java типа **Image**, который обновляется при каждом частичном обновлении страницы.

Важно: Элемент типа **IMAGE** с переменной в качестве атрибута для **USE** автоматически становится элементом Java и не нуждается в установке этого признака. Он применим только для статических IMAGE-элементов, которые изменяются через присваивание свойств (например, **?String1{PROP:Text}='New.gif'**).

Alternative text (альтернативный текст)

По желанию можно предоставить текст для отображения, пока загружается изображение. Это добавляется к признаку (**tag**) HTML **IMG ALT=**. Альтернативный текст отображается, пока графический файл передается в броузер (до отображения изображения), или вместо изображения, если пользователь отключил отображение изображений в настройках броузера.

Border Width (толщина обрамления)

Этот параметр доступен для элементов **SHEET**, **OPTION** (если с параметром **boxed**) и **GROUP**. Определяет толщину границы управляющего элемента. По умолчанию установлено значение 2, определяющее тонкую границу. Для полного отключения границы определите значение 0.

HTML

Одной из наиболее мощных возможностей шаблонов (template) IBC является возможность встраивания HTML-кода в HTML-страницы, порождаемые web-приложениями. Эта возможность позволяет вам добавлять HTML — код перед или после элементов на результирующей web странице. Этот код не работает при запуске приложения через Windows.

Используя встроенный HTML, вы можете писать любой HTML-код, поддерживаемый броузером. Вы можете вставлять ваши собственные JavaScript, Java апплеты, элементы ActiveX, файлы Shockwave и другие объекты.

По желанию, вы можете установить признак **Remove Default HTML generation** для подавления генерации кода HTML для управляющего элемента.

События

В этой закладке вы можете переопределить для управляющего элемента обработку событий, заданную по умолча-

нию. Эта закладка применима только для тех элементов, которые порождают события.

Каждый управляющий элемент имеет по умолчанию некоторое поведение. Оно определяет, как обрабатываются его события. Например, действием по умолчанию для командной кнопки (**command button**) является передача страницы серверному приложению и возврат свежей web-страницы.

Способность переопределять обработку событий, заданную по умолчанию для режима выполнения приложения в броузере, позволяет вам оптимизировать приложение для web-среды и быть спокойным за то, что весь ваш встроенный (**embedded**) код выполняется в те моменты, когда вы этого ожидаете. Например, события, порождаемые элементом «поле ввода» (**entry**), обрабатываются по умолчанию броузером. Это означает, что любой код, определенный для события **Event:Accepted** для поля ввода не будет выполнен до тех пор, пока страница не будет отправлена по действию командной кнопки или другого управляющего элемента, отправляющего страницу. Используя переопределение для управляющего элемента, вы можете назначить частичное обновление по событию **Accepted** для поля ввода и встроенный код выполнится (как при работе под Windows).

По умолчанию у большей части управляющих элементов, позволяющих ввод данных, обработка событий производится в броузере. Поэтому большая часть встроенного кода не будет выполнена в ожидаемые моменты (например, код во вставке для события **Event:Accepted** ! для управляющего элемента не будет выполнен, пока страница не будет отправлена по кнопке **OK**). Этот раздел позволит вам переопределить обработку событий броузером.

Чтобы переопределить обработку события, выделите событие и нажмите кнопку **Properties**.

Override Default Action

Установите признак для переопределения действия по умолчанию для управляющего элемента.Установка этого признака открывает следующие пункты:

- **Action on Event** — Выберите вариант действия для выполнения при появлении события.
- **Process on Browser** — Разрешает произвести обработку события локально в броузере.
- **Partial page refresh** — Определяет, что все управляющие элементы Java и поля ввода HTML должны получить и отобразить обновленные данные.
- **Complete page refresh** — Заменить страницу целиком.

Классы

Закладка **Classes** предлагает вам определить, какие классы (объекты) использовать для решения различных задач и исходные модули, содержащие определения классов. Такой подход дает вам возможность использовать столько классов, сколько хотите из библиотеки IBC и сколько хотите — ваших собственных.

Чтобы изменить класс для пункта или переопределить класс, выделите его в списке и нажмите кнопку **Properties**.

Override default class (Переопределение классов по умолчанию)

Чтобы переопределить класс IBC, установите этот флажок и определите имя класса (**Class Name**), файл заголовков (**Header**) и файл реализации (**Implementation** (.CLW)) в полях, описанных ниже.

Class Name (Имя класса)

Определите имя класса, который будет использоваться или имя класса по умолчанию, если Вы хотели бы переопределить класс по умолчанию.

Если вы выберите другой класс из **IBC Library**, то вы не должны определять **Header** или **Implamentation** файлы.

Header file (файл заголовка)

Определите файл заголовка (файл, содержащий объявления класса) или выберите файл из **FILEDIALOG** нажатием кнопки (...).

Implementation file (файл реализации)

Определите файл реализации (файл, содержащий определения класса или исходный код) или выберите файл из **FILEDIALOG** нажатием кнопки (...).

Процедурные параметры компоненты «Окно»

Заголовок

Это область в верхней части «окна» на HTML-странице. Это часть, представляющая заголовок окна.

Override Global Settings

Установите этот флаг для переопределения установок для **Caption** в шаблоне **Application Global Extention template**. Установка этого признака открывает для определения другие пункты.

Include caption

Установите этот флаг для отображения заголовка (**Caption**). Если флаг не установлен, то заголовок не используется.

Background color

Вы можете определить цвет для использования в области заголовка.

Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из **COLORDIALOG**, нажав на кнопку (...).

По умолчанию установлен темно-синий цвет (**Navy Blue**) (эквивалент **COLOR:Navy**). Если цвет не определен и окно приложения имеет атрибут **COLOR**, то этот цвет отображается в броузере. Для определения данного свойства сгенерированный код использует метод **WebCaption.SetBackground**.

Background Image

Вы можете определить изображение для использования его в области заголовка в качестве фона. Определите имя файла с изображением или выберите его, используя диалог **FILEDIALOG**, нажав кнопку (...).

По умолчанию изображение не используется. Для определения данного свойства сгенерированный код использует метод **WebCaption.SetBackground**.

Совет: Изображение, используемое для фона, повторяется столько раз, сколько позволяет его размер внутри области HTML <**TABLE**>, представляющей область заголовка окна.

Обеспечьте небольшой размер изображения для того, чтобы сохранить трафик.

Alignment

Вы можете управлять выравниванием текста в области заголовка.

Варианты для выбора — это выравнивание влево, вправо и по центру.

По умолчанию — по центру.

Font Family name

Это позволяет вам определять шрифт для отображения на экране.

Запомните, что броузер может отображать только те шрифты, которые установлены на компьютере клиента.

Font size

При желании, вы можете определить размер шрифта, отображаемого в области заголовка. По умолчанию не установлен (выбирается тот размер, который по умолчанию использует броузер).

Font color

Вы можете определить цвет шрифта для области заголовка.

Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из **COLORDIALOG**, нажав на кнопку (...).

Меню

Это область меню в верхней или боковой части «окна» в HTML-странице.

Override Global Settings

Установите этот флаг для переопределения установок для **Menu** в **Application Global Extention template**. Установка этого признака открывает для определения другие пункты.

Background color

Вы можете определить цвет для использования в области меню.

Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из **COLORDIALOG**, нажав на кнопку (...).

Для определения данного свойства сгенерированный код использует метод **WebMenuBar.SetBackground**.

Background Image

Вы можете определить изображение для использования его в области меню в качестве фона. Определите имя файла с изображением или выберите его, используя диалог **FILEDIALOG**, нажав кнопку (...).

По умолчанию изображение не используется. Для определения данного свойства сгенерированный код использует метод **WebMenuBar.SetBackground**.

Совет: Изображение, используемое для фона повторяется столько раз, сколько позволяет его размер внутри области HTML <**TABLE**>, представляющей область заголовка окна.

Обеспечьте небольшой размер изображения для того, чтобы сохранить трафик.

Alignment

Вы можете управлять расположением меню. Возможны следующие варианты: **Above Toolbar** (по умолчанию), **Left of Window**.

Панель инструментов

Это область панели инструментов, находящаяся в верхней части «окна» в HTML-странице (под областью заголовка).

Override Global Settings

Установите этот флаг для переопределения установок для **Menu** в **Application Global Extention template**. Установка этого признака открывает для определения другие пункты.

Override Global Settings

Установите этот флаг для переопределения установок для **Toolbar** в **Application Global Extention template**. Установка этого признака открывает для определения другие пункты.

Background color

Вы можете определить цвет для использования в области панели инструментов. Определите цвет (**Color**), задав цветовой числовой эквивалент (**Color equate**) или выбрав цвет из **COLORDIALOG**, нажав на кнопку (...). Для определения данного свойства сгенерированный код использует метод **WebToolBar.SetBackground**.

Background Image

Вы можете определить изображение для использования его в области панели инструментов в качестве фона. Определите имя файла с изображением или выберите его, используя диалог **FILEDIALOG**, нажав кнопку (...). По умолчанию изображение не используется. Для определения данного свойства сгенерированный код использует метод **WebToolBar.SetBackground**.

Совет: Изображение, используемое для фона, повторяется столько раз, сколько позволяет его размер внутри области HTML <TABLE>, представляющей область заголовка окна.

Обеспечьте небольшой размер изображения для того, чтобы сохранить трафик.

Close button
Override Global Settings

Установите этот флаг для переопределения установок для кнопки **Close** в **Application Global Extention template**. Установка этого признака открывает для определения другие пункты.

Create extra close button

Определяет, когда создавать кнопку **Close** для окна.

Image for close

Определяет пиктограмму для отображения для кнопки **Close**.

Определите имя файла с пиктограммой, или выберите файл в диалоге **FILEDIALOG**, нажав кнопку (...). По умолчанию определен файл EXIT.ICO (маленькая голубая X) (поставляется с Clarion).

Клиентская область

Это область «окна» на странице HTML, представляющая клиентскую область приложения.

Override Global Settings

Установите этот флаг для переопределения установок для кнопки **Close** в **Application Global Extention template**. Установка этого признака открывает для определения другие пункты.

Background Color

Вы можете задать цвет, который будет использоваться для вашей клиентской области.

Определите цвет (**Color**), цветовой эквивалент (**Color equate**), или выберите цвет нажатием кнопки (...) в **COLORDIALOG**. Для определения данного свойства сгенерированный код использует метод **WebClientArea.SetBackground**.

Background image

Вы можете определить изображение, которое будет отображаться на экране как фон для клиентской области вашего приложения. Задайте имя файла изображения или выберите файл из **FILEDIALOG** нажатием кнопки (...). По умолчанию изображения нет. Для определения этого свой-

ства сгенерированный код использует метод **WebClient-Area.SetBackground**.

Совет: Изображение, используемое для фона, повторяется столько раз, сколько позволяет его размер внутри области HTML <**TABLE**>, представляющей область заголовка окна.

Обеспечьте небольшой размер изображения для того, чтобы сохранить трафик.

MDI-параметры процедуры Frame

Меню приложения

Override Global Settings

Установите этот флаг для переопределения установок для панели управления MDI в **Internet Application Global Extention template**.

Установка этого признака открывает для определения другие пункты.

Inclide on Child Windows

Выберите значение параметра из выпадающего списка. Возможны следующие варианты:

- **Global Setting** — Пункты меню будут отображаться в дочерних окнах так, как определено в глобальных параметрах (**Global**).
- **All Menu Items** — Все пункты меню отобразятся в дочерних окнах.
- **No Menu Items** — В дочерних окнах не будут отображаться пункты меню.
- **Selected Menu Items** — Позволяет вам выбрать определенные пункты меню из списка.

Ignore code in frame's ACCEPT loop

Установите этот флажок, чтобы код для пунктов меню, расположенный в цикле ACCEPT игнорировался.

Панель инструментов приложения

Этот раздел определяет стиль обработки управляющих элементов панели приложения. Он дает вам возможность определить, какие из глобальных управляющих элементов панели отображаются в «дочерних» окнах.

Override Global Settings

Установите этот флаг для переопределения установок для панели управления MDI в **Internet Application Global Extention template**.

Установка этого признака открывает для определения другие пункты.

Inclide on Child Windows

Выберите значение параметра из выпадающего списка. Возможны следующие варианты:

- **Global Setting** — Пункты панели управления будут отображаться в дочерних окнах так, как определено в глобальных параметрах (**Global**).
- **All Toolbar Items** — Все пункты панели управления отобразятся в дочерних окнах.
- **Standard Toolbar Only** — Только стандартные пункты панели управления отобразятся в дочерних окнах.
- **No Toolbar Items** — В дочерних окнах не будут отображаться пункты панели управления.
- **Selected Toolbar Items** — Позволяет вам выбрать определенные пункты панели управления из списка.

Ignore code in frame's ACCEPT loop

Установите этот флажок, чтобы код для пунктов панели управления, расположенный в цикле ACCEPT, игнорировался.

Dynamic Html Code Template

Этот код шаблона позволяет вам вставить динамический HTML-код в любую из точек встройки Internet. Этот шаблон годится только для тех точек вставки кода, которые могут «на ходу» выдавать данные на передаваемую страницу HTML.

Вы можете определить в поле ввода любое корректное выражение Clarion.

Все переменные, использованные в приложении, будут иметь текущее значение на момент времени, когда пишется HTML код.

Важно: При создании выражения для записи кода HTML Вы должны помнить о корректной обработке специальных символов, таких как <, используя в этих случаях два символа подряд.

Этот шаблон использует метод **Target.Writeln** для записи значения выражения на передающуюся страницу HTML.

Static Html Code Template

Этот шаблон позволяет вам вставить статический HTML-код в любую из точек вставки кода для Internet. Этот шаблон годится только для тех точек вставки кода, которые могут «на ходу» выдавать данные на передаваемую страницу HTML.

Вы можете задать любой значимый код HTML в поле ввода.

Этот шаблон использует метод **Target.Writeln** для записи значения выражения в передающуюся страницу HTML.

Важно: Если вы используете **Static HTML Code Template**, обработка специальных символов поддерживается автоматически.

GetCookie Code Template

Этот шаблон позволяет вам восстановить cookie (переменную, используемую для координации в межзадачном пространстве) с клиентской машины.

Cookie Name

Обеспечьте имя для cookie. Это имя используется в шаблоне **SetCookie Code** для записи cookie. Если cookie не существует, устанавливаемой переменной присваивается нулевое значение.

Variable to Set

Выберите переменную из схемы связей файлов нажатием кнопки (...).

Значение cookie присваивается переменной.

SetCookie Code Template

Этот шаблон позволяет вам установить cookie на клиентской машине для восстановления в дальнейшем.

Cookie Name

Обеспечьте имя для cookie. Это имя для использования в шаблоне **GetCookie Code** при восстановлении cookie. Если существует переменная cookie с тем же именем, то она переписывается.

New Value

Задайте значение или выберите переменную из схемы связи файлов (**file schematic**) нажатием кнопки (...). Это значение присваивается cookie.

Cookies (Persistent Client Data)

Cookies являются методом для web-серверов как для хранения, так и для восстановления информации на клиентской стороне соединения. Это позволяет серверу хранить данные на машине клиента и затем их восстанавливать.

Сервер может послать клиенту (броузеру) кусок данных, который клиент хранит локально.

Это известно как cookie (источник названия неизвестен). Cookies содержат диапазон URL, для которых это допустимо. Впоследствии, когда клиент возвращается по URL внутри этого диапазона, сервер может опросить cookie и использовать эти данные. Сервер не может восстанавливать информацию с других серверов (то есть, сервер не может опрашивать cookie, которые находятся вне его области диапазонов).

Этот механизм напоминает парадигму хранения и восстановления INI-файлов в Windows (**GETINI** и **PUTINI**) и обеспечивает метод для идентификации предпочтений пользователя и других данных. Например, приложение, которое требует от пользователя обеспечить его имя до входа может использовать cookie, чтобы избежать повторной регистрации после первого посещения.

Важно: Cookies являются машиннозависимыми, так что клиенту, который осуществляет доступ к сайту более чем с одной машины, необходимо будет обеспечить cookie для каждой машины, поэтому cookie хранится на машине. К тому же, cookies зависят от броузера: клиент, пользующийся

более чем одним броузером, должен будет устанавливать и получать cookies для каждого броузера.

Ваши web-пригодное приложения могут использовать cookies для хранения предпочтений пользователя, таких как город или штат по умолчанию для заведения новых записей. Эти установки могут быть восстановлены каждый раз, когда пользователь выполняет приложение через web.

AddServerProperty Code Template

Этот шаблон позволяет вам устанавливать значение заданного выходного элемента данных http в заголовке HTTP.

Property Name

Обеспечьте имя для cookie. Это имя, использующееся в **SetCookie Code Template** для записи cookie. Если cookie не существует, устанавливаемой переменной присваивается нулевое значение.

Property Value

Выберите переменную из схемы связи файлов (**file** schematic) нажатием кнопки (...). Значение cookie присваивается свойству.

GetServerProperty Code Template

Этот шаблон позволяет вам получить значение заданного элемента данных http в заголовке HTTP.

Property Name

Обеспечьте имя для свойства HTTP. Это имя используется в методе **SetServerProperty** для установки значения поля HTTP. Если поле HTTP не существует, устанавливаемой переменной присваивается нулевое значение.

Variable to Set

Выберите переменную из схемы связей файлов (**file schematic**) нажатием кнопки (...). Значение свойства присваивается переменной.

Дизайн Web-приложений

Большинство правил разработки приложений применимы к разработке Web-приложения. В обоих случаях (web, Windows) одинаково важно обеспечить согласующийся, понятный интерфейс для обеих платформ.

Помните, что «платформа» Web — это не Windows. Ваш интерфейс должен бы быть интуитивно понятным для пользователей на всех платформах. Управляющие элементы в библиотеке поддержки Java интуитивно понятны, но для облегчения их использования вы можете пожелать лаконичного объяснения того, как они работают в вашем приложении.

Соображения пропускной способности

Web вводит одну дополнительную программистскую проблему — сохранение трафика. Важно сохранять ваши окна простыми и использовать все методы, способные сократить сетевой трафик.

Использование частичного обновления

Использование частичного обновления, где только возможно, — лучший путь оптимизировать ваши Web-приложения.

Есть много случаев, когда подходит частичное обновление, но по умолчанию обновление — полное. Это необходимо по той причине, что шаблоны не могут предусмотреть все возможности. Например, список с вариантами сортировки, у которого нет управляющих элементов, размещенных на закладках, работает лучше, если вы используете

индивидуальные переопределения для управляющих элементов для задания частичного обновления при выборе закладки. При этом вместо полной замены страницы будут изменяться только данные в списке.

Чтобы переопределить поведение **SHEET** в предыдущем примере, произведите следующие действия:

1. Из окна **Procedure Properties** нажмите кнопку **Internet Option**.

2. Выберите закладку **Controls**.

3. Выделите управляющий элемент **Sheet** в списке **Individual Control Options** (мастер при генерации **SHEET** обычно называет его **?CurrentTab**).

4. Нажмите кнопку **Properties**, затем выберите закладку **Events**.

5. Выделите событие **Accepted**, потом нажмите кнопку **Properties**.

6. Установите флажок **Override default action** (переопределить действие по умолчанию), затем выберите из выпадающего списка **Partial page refresh** (частичное обновление страницы).

7. Нажмите кнопки **OK** во всех окнах для сохранения и выхода.

Работа с управляющими элементами

Будьте экономны с управляющими элементами. Разместите в окне необходимый минимум управляющих элементов. Это хорошая практика в разработке приложений Windows и еще более важно в реализации browser/server.

При использовании списков разместите в списке настолько мало управляющих элементов, насколько возможно для уникальной идентификации записи пользователем. Это сокращает количество данных, пересылаемых для наполнения списка. Если вы хотите отобразить больше данных для

каждой записи, Вы можете разместить «горячие поля» после списка, и они будут обновляться по мере скроллирования списка пользователем.

Использование графики

Используйте графику умеренно. Это общее правило для разработки web. Вам следовало бы ограничить количество графики на экране, чтобы ускорить загрузку страницы. В добавление, вы должны бы сократить размер файла настолько, насколько возможно, для сохранения трафика. Многие графические утилиты обладают инструментом для приспособления графических файлов к использованию в web.

Защита операции подгрузки с помощью Splash окна

Для того, чтобы запустить приложение, приспособленное для Web, броузеру клиента должна быть доступна **JSL** (Java Support Library). Начинающие пользователи должны подгрузить Clarion.CAB (для Microsoft Internet Explorer) или Clarion.ZIP (для Netscape). Для большинства броузеров JSL подгружается единожды и сохраняется в кэше (до тех пор, пока пользователь не очистит его). Хотя JSL чрезвычайно компактна для той функциональности, какую она обеспечивает, ее подгрузка займет несколько минут. Имея это ввиду, мы будем использовать всплывающее окно для предупреждения начинающего пользователя о том, что идет подгрузка. Размещая кнопки Java в этом окне, мы можем воспрепятствовать работе пользователя до тех пор, пока не загрузится JSL и кнопки Java не будут инициализированы.

Создание окна и переименование BUTTON в Java Button

Создайте процедуру, используя шаблон **WindowProcedure**. Это позволит вам именовать вашу процедуру — **Splash**. Окно **Splash** должно содержать некоторый текст и шаблон

управления **Close Button**. Текст на кнопке **BUTTON** вы можете изменить на **Continue**. Так как кнопка создана как HTML-кнопка по умолчанию, вы должны будете уточнить, что вы хотите, чтобы это была Java кнопка (она не будет доступна конечному пользователю до тех пор, пока JSL будет подгружаться).

1. В **Application Tree** подсветите новую процедуру, нажмите кнопку **Properties**.

2. Нажмите кнопку **Internet Options**.

3. Выберите закладку **Controls**.

4. Подсветите **?Close** в списке **Individual Control Overrides**, нажмите кнопку **Properties**.

5. Выберите закладку **Classes**.

6. Пометьте признак **Override default Class**, выберите **WebJavaButtenClass** из выпадающего списка **ClassName**.

7. Нажмите **OK**.

Вызов процедуры перед открытием Application Frame

1. В **Application Tree** подсветите **Main** процедуру и нажмите клавишу **Proreties**.

Откроется окно **Procedure Properties**.

2. Нажмите клавишу **Embeds**.

Откроется окно **Embedded Source**.

3. Подсветите **WindowManagerExecutable Code Section — Init-(),BYTE**.

4. Нажмите клавишу **Insert**.

Откроется окно **Select Embed Type**.

5. Выделите подсветкой **Source**, нажмите клавишу **Select**.

6. В редакторе **Embedded Source** введите следующее:

```
IF WebServer.Active THEN Splash
```

Это позволит процедуре **Splash** быть вызванной, когда приложение исполняется в Web.

7. Установите значение **Priority** в 1.

8. Убедитесь, что вставка внесена в список перед вызовом любой другой процедуры, использующей кнопки «вверх» и «вниз».

Это гарантирует, что процедура **Splash** вызывается перед открытием любого другого окна.

9. Нажмите кнопку **Close** в окне **Embedded Source** и кнопку **OK** в окне **Procredure Properties**.

10. Нажмите кнопку **Procedures**.

Откроется окно **Procedure**.

11. Выделите **Splash**, затем нажмите **OK**.

Это «подцепит» процедуру **Splash** к **Main** процедуре в дереве приложений **Application Tree**. Сделать это необходимо, если ваше приложение используется в **Local Map's**.

Соображения о косметическом дизайне

Использование групп

Когда вы размещаете **GROUP** или **WINDOW**, предложения, относящиеся к описанию управляющего элемента, не обязательно оканчиваются внутри структуры **GROUP**. Это может вызвать ситуацию, когда HTML-представление окна отличается от оригинала. Убедитесь, что все управляющие элементы, которые вы желаете разместить внутри **GROUP** действительно находятся внутри **GROUP**-структуры.

В первом примере из описанных ниже (**BadWind**) предложения, описывающие управляющие элементы, все

11 Зак. 3215

находятся вне **GROUP**-структуры. Это окно прекрасно отображается в Windows, так как значения атрибута **AT** управляют положением и размером GROUP box. При запуске через Web, **GROUP box** является клеткой HTML **<TABLE>** и управляется ее содержимым.

```
BadWind WINDOW('Caption'), AT(,,260, 120), GRAY
GROUP('Customer Info'), AT(5, 9, 205, 102),
USE(?Group1), BOXED
END
PROMPT('Customer:'), AT(11, 28),
USE(?CUST:Name:Prompt)
ENTRY(@s30), AT(61, 26), USE(CUST:Name), LEFT, REQ
PROMPT('Address:'), AT(15, 47),
USE(?CUST:Address:Prompt)
ENTRY(@s30), AT(61, 65), USE(CUST:Address), LEFT
PROMPT('City:'), AT(29, 69), USE(?CUST:City:Prompt)
ENTRY(@s30), AT(61, 65), USE(CUST:City), INS
PROMPT('State:'), AT(25, 88), USE(?CUST:State:Prompt)
ENTRY(@s2), AT(61, 86), USE(CUST:State), LEFT, UPR
END
```

Во втором примере (**GoodWind**) описания управляющих элементов находятся внутри **GROUP**-структуры (то есть, между предложениями **GROUP** и **END**) и будут отображаться так, как предполагалось при выполнении через Web.

```
BadWind WINDOW('Caption'), AT(,, 260, 120), GRAY
GROUP('Customer Info'), AT(5, 9, 205, 102),
USE(?Group1), BOXED
PROMPT('Customer:'), AT(11, 28),
USE(?CUST:Name:Prompt)
ENTRY(@s30), AT(61, 26), USE(CUST:Name), LEFT, REQ
PROMPT('Address:'), AT(15, 47),
USE(?CUST:Address:Prompt)
ENTRY(@s30), AT(61, 65), USE(CUST:Address), LEFT
PROMPT('City:'), AT(29, 69), USE(?CUST:City:Prompt)
```

```
ENTRY(@s30), AT(61, 65), USE(CUST:City), INS
PROMPT('State:'), AT(25, 88), USE(?CUST:State:Prompt)
ENTRY(@s2), AT(61, 86), USE(CUST:State), LEFT, UPR
END
END
```

Использование изображений

Управляющие элементы **IMAGE** с переменной в качестве атрибута предложения **USE** автоматически становятся управляющими элементами Java. Эти управляющие элементы автоматически обновляются, когда значение переменной изменяется (т.е. производится частичное обновление страницы). Если Вы хотите использовать эту возможность для статического **IMAGE**, который изменяется через присвоение значения свойству (например, **?Image{PROP:TEXT}='New.gif'**), используйте **Individual Controls Overrides** (индивидуальные переопределения для управляющего элемента) и установите признак для динамического обновления.

Графические файлы, используемые управляющими элементами **IMAGE**, раскрываются во временную директорию для сеанса соединения в случае, если они не найдены в директории /**PUBLIC**. Динамическая библиотека (**runtime**) будет раскрывать файлы различных типов, но большинство броузеров поддерживают только форматы GIF и JPG. Поэтому при работе в Web вам следует ограничиться только двумя этими форматами при выборе типа графического файла для управляющего элемента **IMAGE**. Вы также должны устанавливать режим **hide** (спрятать) для **IMAGE**, формат которого не поддерживается броузером, используя **Individual Controls overrides**. Если **IMAGE** использует файл, который не включен в исполняемый модуль (**not linked**), вы должны разместить этот файл в директорию приложения.

Вы должны обеспечить альтернативный текст для изображения (в **Individual Control Overrides**). Он добавляется к признаку HTML <ING ALT=>.

Альтернативный текст отображается, пока графический файл передается в броузер (до отображения изображения), или вместо изображения, если пользователь отключил в настройках броузера отображение изображений.

Пиктограммы, используемые в управляющих элементах **LIST** или на **BUTTONs**, автоматически не раскрываются и должны быть размещены в директории /**PUBLIC**.

Если вы ссылаетесь на изображение в HTML-коде, вы должны указать расположение графического файла. Если вы производите размещение для EXE-версии Application Broker, вы можете поставить перед именем файла знак / и разместить изображение в директории /**PUBLIC**. Например,

```
<IMG SRC="/ LOGO.GIF">
```

Если вы используете **ISAPI DLL** версию Application Broker, вы должны использовать метод **SELF.FILES.GETAlias** для определения виртуального пути к файлу.

Например:

```
Target.WriteLN('<<IMG SRC="' &
SELF.FILES.GETAlias('mygif.gif') & '">')
```

найдет файл mygif.gif в любой директории, предоставленной серверному приложению.

Разработка интерфейса пользователя

MDI window access

Windows-приложения часто используют Multiple Document Interface (MDI). Это позволяет открыть несколько экземпляров «дочерних» MDI-окон. Каждое из этих окон доступно и в него можно перейти, используя несколько навигационных методов (например, меню Window). Это очень удобно, но вносит некоторую путаницу при переносе приложения на web-платформу. Web-страница в броузере представляет собой один документ, однако основное сервер-

ное приложение может быть MDI-приложением и содержать множество окон. В приложении на сервере может быть открыто много окон, но броузер отображает только текущее. Вы должны помнить это при разработке приложения.

В Web-пригодном приложении вы можете разрешить видимость в «дочерних» окнах всех пунктов меню и панели инструментов. Это может быть полезно, когда мы хотим позволить пользователю входить в разные части приложения без закрытия «дочернего» окна, пользуясь пунктами главного меню или панели инструментов. В этом сокрыта потенциальная ловушка, заключающаяся в возможности вызова пользователем нескольких экземпляров процедуры. Хотя только одно окно может быть видимо в определенный момент времени, открыто при этом может быть несколько. Если открыто два или более одинаковых окна, пользователю может показаться, что окно не закрывается при нажатии кнопки **Close**. По этой причине вы должны запретить доступ к пунктам **Global menu/Toolbar** или определить ограничение на единственность экземпляра для каждой MDI-процедуры, используя код для потокового ограничения (**Thread limiting code**). Один из подходов к ограничению нитей продемонстрирован в одном из стандартных примеров Clarion — EventMgr.APP.

Ограничение редактирования по месту (Edit-In—Place))

ABC шаблоны Clarion позволяют использовать редактирование по месту с элементами выбора. Однако, это средство не поддерживается во время исполнения в Web, в этом случае вы можете использовать **Form** для обновлений. Этот простой метод — альтернатива между редактированием по месту во время локального исполнения под Windows и вызовом формы во время исполнения в Web.

Если вы адаптировали **Edit-In-Place** и уточнили процедуру обновления с шаблоном управления BrowseBox, то

вы уже сделали две трети всей работы. Код, сгенерированный шаблоном, вызывает отдельную процедуру обновления или исполняет редактирование по месту, в зависимости от значения **BRWn.AskProcedure**. Установите значение **BRWn.AskProcedure** в ноль, и вы получите редактирование по месту; установите в единицу, и вы вызовите процедуру обновления.

Для того, чтобы использовать **Edit-In-Place** для локального Windows и **Form** для исполнения в Web:

1. Выберите процедуру **Browse**, затем нажмите клавишу **Properties**.

2. В разделе **UpdateButton** окна **Procedure Properties** установите флаг **Use Edit in Place** (использовать редактирование по месту).

Отметьте, что процедура замены всегда уточняется.

Далее, задайте код, чтобы установить обновление при редактировании по месту, когда исполняется в Windows и вызовите form при исполнении через Web.

3. Нажмите кнопку **Button**.

Откроется окно **Embedded Source**.

4. Выделите точку вставки кода **Window Manager Method-Init-BYTE()**, затем нажмите кнопку **Insert**.

5. Выделите **Source**, нажмите кнопку **Select**.

6. В редакторе **Embedded Source** введите следующий код:

```
IF WebServer.Active
BRW1:AskProcedure=1
END
```

Неподдерживаемые стандартные Windows-диалоги

Есть определенные стандартные Windows-диалоги, которые непригодны для приложений, работающих в Web.

Вызов любого из них приводит к появлению на экране сообщения **Not Supported**:

```
HALT
COLORDIALOG
FILEDIALOG
FONTDIALOG
PRINTERDIALOG
STOP
```

Если вы вызываете любой из этих диалогов нажатием кнопки, то используйте **Individual Control Options** для кнопки для того, чтобы спрятать ее при работе через броузер («**Hide if launched from Browser**») (**Internet Options Controls**).

Если вы вызываете функцию из исходного кода, заключите вызов функции в условную структуру.

Например:

```
IF NOT WebServer.Active
!Работаем через web?
retval=COLORDIALOG()
!Если нет, вызываем COLORDIALOG
END
```

Использование параметров командной строки

Если ваше приложение должно получать параметры из командной строки, вы можете передать их, используя командную строку броузера, или гиперссылку.

В поле ввода, содержащем адрес расположения броузера (URL), определите URL, за ним следует имя исполняемого модуля, затем точка и нуль (.0) и завершает строку параметр с лидирующим вопросительным знаком.

Например,

```
HTTP://mydomain.com/myapp.exe.0?MyParametr
```

Для обработки параметра в приложении, вы должны опросить свойство **WebServer.CommandLine**. Если вы созда-

ете гибридное приложение и хотите получать параметры командной строки в Windows и Web, используйте код, похожий на приведенный в примере:

```
IF WebServer.Active ! Работаем через web?
PRE:MyField = WebServer.CommandLine
ELSE
PRE:MyField = COMMAND(,')
END
```

Важно: Если вы передаете несколько параметров, то необходимо произвести разбор строки для доступа к индивидуальным параметрам.

Изменение класса для индивидуального параметра

В некоторых случаях у нас может появиться желание заменить класс, заданный по умолчанию для управляющего элемента. Например, управляющий элемент **STRING**, являющийся переменной, трансформируется в управляющий элемент Java String, а вам хотелось бы, чтобы он стал обычным текстом HTML. Вы можете изменять элемент за элементом, редактируя закладку **Individual Control Overrides Classes Tab**. В этом примере вы не переопределяете класс, а просто определяете другой класс для использования при обработке данного управляющего элемента.

1. В окне **Procedure Properties** нажмите кнопку **Internet Options**.

2. Выберите закладку **Controls**.

3. Выделите управляющий элемент в списке **Individual Control Overrides**, затем нажмите кнопку **Properties**.

4. Установите флаг **Override Default Class**.

5. Выберите класс из выпадающего списка (в этом примере это класс **WebHtmlStringClass**). Вы не должны обеспечивать файл заголовка (**Header File**) и реализации (**Implementation**).

Вы можете использовать тот же подход для замены **JavaImageControl** на элемент HTML **<IMG>**.

Вызовы API

Вызовы API в Windows связаны с машиной, на которой выполняется приложение. В действительности, Web-пригодные приложения выполняются на сервере, а представление посылается клиенту в форме страниц HTML. Таким образом, любые вызовы API в вашем приложении выполняются на сервере.

Во многих случаях это не годится. Например, издающий звук файл на сервере — вообще не лучшая идея, и пользователь, работающий в приложении, не будет его слышать. В других случаях, вам следует запретить вызов, когда приложение работает через web.

Если вы производите вызов с помощью управляющего элемента **BUTTON**, используйте **Individual Control Options**, чтобы **«Спрятать, если запущено из Броузера»** («Hide if launched from Browser») (управляющие элементы параметров Internet).

Если вы делаете вызов в исходном тексте, поместите функцию вызова внутрь условной структуры. Например:

```
IF NOT WebServer.Active ! Работаем в Web?
SoundFile='fanfare.wav'
sndPlaySound(SoundFile,1)
END
```

В других случаях будет вполне подходяще сделать вызов на сервере. Например, процедура, которая использует MAPI, чтобы послать e-mail с сервера, основана на событии. В других случаях вам следует удостовериться, что вызов работает правильно на сервере. Стоит придерживаться того же правила при выполнении через web.

Похожим образом отчеты без разрешенного просмотра перед печатью (**Print Preview**) будут печататься на серве-

ре. В некоторых случаях это допустимо, но важно при этом понимать, что происходит.

Соображения секретности

Есть несколько способов обеспечения секретности в Ваших web-приложениях.

- Обеспечение секретности внутри самого приложения.
- Ограничение доступа (парольная защита) к процедуре, когда она стартует через Web.
- Пересылка через секретное соединение.

Первый метод — обеспечение безопасности внутри самого приложения — не требует дополнительного рассмотрения в вашем Web-приложении. Усиление секретности в версии Windows должно так же работать в вашем Web-приложении.

Второй метод — ограниченный доступ при работе в Web — использует встроенное в Броузер распознавание.

Третий метод — пересылка через секретное соединение — преследует другую цель. Он не предназначен для ограничения доступа пользователю. Он предполагает запрещение перехвата данных во время передачи. Эта мера безопасности может быть использована отдельно или вместе с любой из двух других.

Использование паролей

Парольная защита шаблона расширения процедур для Internet использует встроенную в HTTP поддержку аутентификации. При вызове защищенной паролем процедуры отображается окно аутентификации броузера. Вам не нужно создавать окно для сбора информации для регистрации в системе. Парольная защита основана на области, имени

пользователя и пароле. Областью является защищенная процедура.

Когда броузер обращается к защищенной паролем области, он получает обратно ответ, содержащий имя пользователя и пароль для доступа к области. По умолчанию имя области создается из заголовка окна и имени процедуры. Все это сохраняется в свойстве **WebWindow.AuthorizeArea**. Броузер предлагает пользователю ввести имя и пароль. Это отправляется затем для проверки в программу. Если программа воспринимает пароль (т.е. она возвращает TRUE из метода **WebWindow.ValidatePassword**), отображается новая страница, иначе броузер предлагает ввести все сначала. После трех попыток броузер отображает сообщение, информирующее пользователя, что доступ запрещен. Эта страница автоматически возвращает пользователя в последнее активное место в программе.

Важно: Если страница уже посещалась во время текущего соединения, броузер подставляет пользовательское имя и пароль без необходимости ввода. Эта возможность встроена в большинство броузеров.

В процедурный шаблон (procedure **template**) встроены два уровня парольной защиты. Простейшим методом является выбор ограниченного доступа и определение единственного или переменного пароля. В этом случае происходит автоматическая проверка шаблоном и игнорирование имени пользователя. Если вы используете переменную, шаблон сверяет введенный пароль с текущим значением переменной.

Более мощным методом является переопределение метода **WebWindow.ValidatePassword** вставкой кода в точку вставки кода **Internet-Password Validation Code Section**. Эта точка вставки находится внутри метода с двумя параметрами: **UserName** и **Password**, получаемыми из броузера. Вы должны возвращать TRUE, если пароль верен, и FALSE —

если нет. Это позволяет вам выбирать информацию из файла или использовать другой метод для проверки пароля.

Пример:

```
USE:UserID=UserName
Get(Userlist,USE:UserIDKEY)
IF ERRORCODE() THEN RETURN(False).

IF USE:UserPassword=Password
RETURN(True)
Else
RETURN(False)
END
```

При желании, вы можете изменить сообщение, отображаемое в диалоге пароля броузера, присваиванием значения **WebWindow.AuthorizeArea** в точке встройки **Internet-After Initializing the window object**.

Использование SSL (Secure Socket Layer)

Эта мера секретности требует, чтобы вы выполняли ISAPI-версию брокера приложения под ISAPI-согласованным Web-сервером, и имели инсталлированный Digital Sertificate.

SSL-поддержка шаблона Internet-расширения процедур использует ISAPI SSL-шифрование во время исполнения процедуры. Когда вызвана процедура с включенным SSL, брокер приложения переключается в режим SSL. Когда процедура заканчивается, восстанавливается нормальный доступ. Это дает возможность секретных транзакций на уровне процедуры. Помните, что шифрование заметно сказывается на производительности. Вам нужно обеспечивать секретность только для процедур, пересылающих важные данные. Разрешение шифровать только такие процедуры улучшает выполнение как на стороне клиента, так и на стороне сервера.

Для включения SSL для процедуры:

1. В окне **Procedure Properties** нажмите кнопку **Internet Options**.

2. Выберите закладку **Advanced**.

3. Установите флаг **Transfer over a secure connection**.

Использование встроенного HTML

Одной из наиболее мощных возможностей IBC-шаблонов (IBC Templates) является способность встраивания HTML-кода в HTML-страницы, являющиеся выходными для web-пригодного приложения, работающего через Application Broker. Когда вы встраиваете HTML-код (используя специальные точки встраивания, добавляемые шаблонами), он вставляется в определенные места в HTML-файле, возвращаемом в броузер, который выполняет приложение.

Существуют два метода встраивания HTML:

1. **Individual Control Overrides** (индивидуальные переопределения для управляющего элемента) в **Internet Procedure Extention Template** (шаблон Интернет-расширение для процедур). Этот пункт предоставляет два текстовых поля для ввода HTML-кода.

2. Использование **Dynamic HTML Code Template** (шаблон для динамического HTML-кода), или **Static HTML Code Template** (шаблон для статического HTML-кода) в одной из точек вставки для Интернет. Эти шаблоны для вывода «на ходу» в передаваемые HTML-файлы используют виртуальный метод **Target.Writeln**.

Static HTML Code Template позволяет вам встраивать код в том виде, как он написан. **Dynamic HTML Code Template** позволяет вам соединять HMTL-код с переменными из вашего приложения.

При желании, вы сами можете использовать метод **Target.Writeln** в любом пригодном для вставки этого кода месте.

Эти пункты вставки кода в начале описания содержат слово **INTERNET**.

Использование метода **Target.Writeln** в одном из таких пунктов позволит вам добавлять HTML-код в различные места HTML-документа в момент передачи пользователю. Этот код не влияет на работу приложения при выполнении его под Windows.

Например, если вы хотите, чтобы в конце страницы, передаваемой Брокером (Application Broker) в процедуру приложения, появился блок текста, вставьте **Static HTML Code Template** в пункте **Internet, before the closing </BODY> tag** в генераторе приложения (**Application Generator**) и определите HTML-код. Этот HTML-код добавляется к результирующей HTML-странице, пересылаемой клиенту.

Вы можете использовать виртуальный метод **Target. Writeln** в любых пунктах вставки кода, где применимы **Dynamic HTML Code Template** и **Static HTML Code Template**.

Пример:

Добавьте этот код в пункт вставки кода **Internet, before the closing </BODY> tag:**

```
Target.WriteLn('<<h>Copyright 1997, TopSpeed&trade;
Corporation, All Rigths Reserved.<</ >')
```

Важно: При написании исходного текста на Clarion для записи HTML-кода помните об особой обработке спецсимволов, таких как <, используя их повторение. При использовании **Static HTML Code Template** эта ситуация отрабатывается автоматически.

Одним из преимуществ использования Clarion-кода в этих точках вставки является возможность управлять

HTML-кодом. Пример, приведенный ниже, показывает простой способ случайного отображения гиперссылки:

```
EXECUTE  RANDOM(1,5)
Target.WriteLn
('<<A  HREF=i.http://www.topspeed.coml.>
Visit  TopSpeed<</A>')
Target.WriteLn
('<<A  HREF=i.http://www.clarionnonline.comli>
Visit  ClarionOnLine<</A>')
Target.WriteLn
('<<A  HREF=i.http://www.isetips.comle>
Visit  IceTips<</A>')
Target.WriteLn
(/<<A  HREF=i.http://www.finansist.comlV>
Visit  Finansist<</A>')
Target.WriteLn
('<<A  HREF=i.http://www.topspeed.com/tsnews.html.>
TopSpeed  News<</A>')
```

Использование ссылок на файлы во встроенном HTML-коде

При использовании ссылок на файлы во встроенном HTML-коде помните, что каждое соединение имеет свою временную директорию. Поэтому, **/PUBLIC** никогда не является текущей директорией для передаваемой web-страницы. Это означает, что вы должны указывать расположение файлов. Это можно сделать двумя способами.

Если вы ссылаетесь из HTML на изображение, необходимо указать расположение файла с изображением. Если вы разворачиваете приложение, используя EXE-версию Application Broker, то должны поставить перед именем файла символ / и разместить файл в директории **/PUBLIC**. Например:

```
<IMG  SRC="/LOGO.GIF">
```

При использовании версии ISAPI DLL брокера приложений, вы должны использовать метод **SELF.FILES.GetAlias()** для определения виртуального пути к файлу.

Например:

```
Target.WriteLn
('<<IMG SRC="'& SELF.Files.GetAlias('mygif.gif')
& '">'
```

найдет файл mygif.gif в любой директории, открытой серверному приложению.

Важно: Предпочтительным является использование метода **SELF.Files.GetAlias()**, потому что он работает в обеих версиях Application Broker (EXE и ISAPI DLL).

Для использования собственных файлов с классами Java Applet, используйте **CODEBASE=признак**, как показано ниже.

Если вы размещаетесь под EXE-версией брокера приложения, вы можете вместо лидирующего символа / разместить файл .CLASS в директорию **/PUBLIC**. Если вы используете версию ISAPI DLL брокера приложения, вы должны опросить свойство **WebServer.JavaClassPath** для определения виртуального пути **<CODEBASE>**.

Пример вставки HTML:

```
<IMG SRC="/mypic.gif">
<applet codebase="/" code="Ticker Tape.class"
width="500" heiht="32">
</applet>
Embedded Source Examples (in any Internet Embed
Point)
Target.WriteLN('<<IMG SRC="'&
SELF.FILES.GETAlias('mygif.gif')&'">')
Target.Writeln('<<applet')
Target.Writeln('Codebase="'& WebServer.JavaClassPath &
'"')
```

```
Target.Writeln('code="TickerTape.class">')
Target.Writeln('<</applet>')
```

Важно: В APPLET HTML tag атрибут **CODEBASE** должен предшествовать атрибуту кода. Это распечатано в ошибочном порядке в некоторых ссылках HTML. Код HTML с атрибутами, стоящими в неправильном порядке, может вызвать ошибку апплета (с сообщением «**Not found**»).

Реализация помощи в ваших Web-приложениях

Ссылки на HTML-странице производятся, основываясь на **Help ID** текущего окна. Это строится одним из двух способов: используя базовый документ с внутристраничными ссылками, или используя индивидуальные документы помощи.

Это определено в **Global Application Extension Template** или в **Procedure Extension template's Internet Options**.

Использование базового документа с внутристраничными ссылками

Этот метод использует единичную Web-страницу с внутристраничными закладками или ссылками. Обращение к странице конструируется добавлением **Help ID** к имени базовой страницы с символом # между ними (то есть **HELP.HTM#IDNAME**). Нажатие на кнопку **Help** вызывает открытие страницы и пролистывание до подходящей ссылки. В примере ниже первое окно имеет **Help ID ~FirstWindowID**. Это значит, что кнопка **Help** вызовет **HelpFile.HTM#FirstWindowID**.

Пример:

```
<html>
<head>
<title> Example Help Document</title>
```


12 Зак. 3215

```
</head>
<body background="bgrnd.gif" bgcolor="#FFFFFF">
<h1 align="center">Program Help </h1>
Вводный текст

Вводный текст

Вводный текст

Вводный текст

<h2 align="center"><a name="FirstWindowID">Help For
First Window</a></h2>
Объяснение как процедура работает
Объяснение как процедура работает
Объяснение как процедура работает
Объяснение как процедура работает
<h2 align="center"><a name="SecondWindowID">Help For
Second Window</a></h2>
Объяснение как процедура работает
Объяснение как процедура работает
Объяснение как процедура работает
Объяснение как процедура работает
</body>
</html>
```

Использование индивидуальных документов помощи

Этот метод использует единственную web-страницу для каждого окна. Запрос к странице строится на добавлении к **Help ID** расширения .HTM. Нажатие на кнопку **Help** вызывает открытие страницы.

Оба метода открывают страницу в новом окне броузера под названием «**_HELP**».

Если вы открываете ваше приложение внутри множества фреймов, один из которых называется «_HELP», страница помощи открывается в этом фрейме.

Web-пригодное приложение, выполняемое брокером приложения, создает HTML-файлы в директории /**Public**. Эти страницы посылаются броузеру, который запускает приложение, и обновляются и перепосылаются, когда клиент взаимодействует с приложением.

Управляющие элементы Windows и их HTML-эквиваленты

Web-пригодное приложение, выполняемое Application Broker, передает HTML броузеру, который запускает приложение и обновляет и перепосылает, когда клиент взаимодействует с Web-страницей, представляющей приложение.

Некоторые управляющие элементы легко транслируются в HTML, другие же создаются как JAVA-классы, используя **Clarion Java Support Library**. Некоторые оконные управляющие элементы еще полностью не воплощены в этой версии.

В списке, приведенном ниже, перечислены стандартные оконные управляющие элементы, поддерживаемые Clarion и их эквиваленты, создаваемые web-пригодным приложением Internet Connect.

SSTRING (переменная строка)

Отображает Java String Control (строка-управляющий элемент JAVA), обновляемый динамически.

STRING

По умолчанию, отображается как текст. Через индивидуальные настройки (**Individual Control Overrides**) можно определить, чтобы отображался как Java String Control (стро-

ка-управляющий элемент JAVA), обновляемый динамически.

Если вы меняете **STRING** через присвоение свойств, необходимо указать, что строка обновляется динамически.

IMAGE

Статическое изображение отображается как HTML-изображение **<IMG>**, где в качестве источника определяется графический файл.

Через индивидуальные настройки (**Individual Control Overrides**) можно определить, чтобы отображался как Java Image Control (изображение — управляющий элемент JAVA), обновляемый динамически.

REGION

Частичная поддержка. **REGION**, который включает элемент **IMAGE** и функциональность которого реализована в его **EVENT:Accepted**, создает HTML-изображение как таблицу изображения (**image map**) (**USEMAP=**) с функциональностью области, ассоциируемой с этой частью изображения.

LINE

Не поддерживается — используйте встроенный HTML для изображения **<HR>** или **<IMG SRC=i.file.gifle>**.

BOX

Не поддерживается — используйте встроенный HTML для изображения **<IMG SRC=i.file.gifle>**.

ELLIPSE

Не поддерживается — используйте встроенный HTML для изображения **<IMG SRC=i.file.gifle>**.

ENTRY

Создается как HTML-поле ввода <INPUT TYPE= TEXT VALUE=value in field>. Шаблоны ввода не поддерживаются.

BUTTON

Создается как <INPUT TYPE=SUBMIT>, если только не имеет **ICON**, затем создается кнопка **JAVA**, на которой отображается пиктограмма. Пиктограммы, отображаемые на кнопках, должны быть размещены в директории /PUBLIC.

PROMPT

Отображается как текст.

OPTION

Создается как HTML <OPTION>. Если **OPTION** имеет атрибут **BOXED**, он реализуется в HTML как <TABLE> с границей, определенной в параметрах **Global** или **Procedure** для **OPTIONs**.

CHECK

Создается как HTML **checkbox** <INPUT TYPE=CHECKBOX VALUE=value in field>.

GROUP

Если **GROUP** имеет атрибут **BOXED**, он реализуется в HTML как <TABLE> с границей, определенной в параметрах **Global** или **Procedure** для **GROUPs**.

Tree

Создает **Java Tree List Box**. Поддерживает все атрибуты, включая условные цвета и пиктограммы. Пиктограммы должны быть размещены в директории /PUBLIC.

LIST

Создает **Java List Box**, который поддерживает большинство атрибутов **LIST**, включая условные цвета и пиктограммы.

Пиктограммы должны быть размещены в директории **/PUBLIC**.

Когда **Java List Box** является активным элементов в броузере, поддерживаются навигационные нажатия (стрелка вверх, страница вверх). Если у **LIST** есть локатор, **Java List Box** поддерживает его, когда он является активным. Обработка двойного нажатия мыши тоже поддерживается. «Взять и перетащить», «редактирование по месту» и контекстное меню по нажатию правой кнопки мыши не поддерживаются.

COMBO

Создается как HTML-поле ввода **<INPUT TYPE= TEXT VALUE=value in field>**.

SPIN

Создается как HTML-поле ввода **<INPUT TYPE= TEXT VALUE=value in field>**.

TEXT

Создается поле HTML **Text <TEXTAREA>**.

CUSTOM(.VBX)

Не поддерживается.

MENU

Создает список гиперссылок, которые отображаются в верхней части HTML-страницы или в левой части окна, что определяется в **Global Internet Options**.

ITEM

Смотри меню.

RADIO

Создает кнопку HTML **Radio**.

APPLICATION

HTML <**TABLE**> внутри HTML-страницы.

WINDOW

HTML <**TABLE**> внутри HTML-страницы.

REPORT

Если доступен **Print Preview**, это создает серию HTML-страниц с кнопками навигации Java (**Next page**, **Previous page**).

Если **Preview** не включен, то отчет будет печататься на сервере.

HEADER, FOOTER, BREAK, FORM, DETAIL

Смотри **REPORT**.

OLE

Не поддерживается (кроме возможности вставки ActiveX в встроенный HTML).

FileDropCombo

Создается как HTML **drop-down** (структура <**SELECT**>) со значениями из файла, указанного в **Options**. Здесь не поддерживаются множественные колонки. При желании вы можете создать список Java **Non-drop**, который поддерживает множественные колонки.

droplist

Создается как HTML **drop-down** (структура <**SELECT**>). Здесь не поддерживаются множественные колонки.

При желании вы можете создать список Java **Non-drop**, который поддерживает множественные колонки.

PROGRESS

Не поддерживается.

SHEET

Создается как управляющий элемент JAVA **Tab**.

TAB

Создается как управляющий элемент JAVA **Tab**.

PANEL

Не поддерживается. Вы можете использовать **GROUP** с подходящей толщиной границы для обеспечения схожего внешнего вида.

TOOLBAR

Создается как ряд в HTML <TABLE>. Управляющие элементы на панели инструментов размещаются, как это определено в глобальных или процедурных Internet-параметрах.

Программирование приложений вручную

Шаблоны Internet Connect порождают код, необходимый для подготовки к работе в Web Clarion-приложений. Однако вы не должны использовать шаблоны Internet Connect для подготовки к Web ваших программ.

Это означает, что вы можете использовать для подготовки к Web ваших вручную написанных программ **IBC Library**. Здесь представляется небольшая, написанная вручную программу «Hello Web», использующая IBC Library и обсуждаются требования проектной системы для IBC Library.

Наиболее легкий путь для изучения использования IBC Library в программах, написанных вручную, — это подготовка приложения к Web с использованием **Internet Connect Templates** с последующим изучением сгенерированного кода.

HelloWeb — пример программы

Следующая гибридная Web/Windows-программа отображает одно окно или Web-страницу с сообщением «Hello Web» и кнопкой «Goodbye Web» для завершения программы.

```
HelloWeb PROGRAM
LinkBaseClasses EQUATE(1)
!Разрешить LINK в описаниях CLASS
!так компоновщик (linker) может найти файлы
!реализации (.clw)
BaseClassDllMode EQUATE(0)
!Активизировать DLL в описаниях
! CLASS
! для требующегося 32-разр.
! разименования (dereference)
INCLUDE('ICBROKER.INC')
! Описать BrokerClass
INCLUDE('ICWINDOW.INC')
! Описать WebWindowClass
INCLUDE('ICSTD.EQU')
! Описать IC стандартные EQUATEs
MAP
Hello
! Прототип процедуры Hello
WebControlFactory(SIGNED),*WebControlClass
! Прототип WebControlFactory
MODULE('  ')
SetWebActiveFrame(<*WebFrameClass>)
! Прототип SetWebActiveFrame
```

```
END
END
Broker BrokerClass
! Описать объект Broker
HtmlManager HtmlClass
! Описать объект HtmlManager
JavaEvents JslEventsClass
! Описать объект JavaEvents
WebServer WebServerClass
! Описать объект WebServer
WebFilesManager WebFilesClass
! Описать объект WebFilesManager
ShutDownManager CLASS(ShutDownClass)
Close PROCEDURE,VIRTUAL
END
ICServerWin Window,AT(-1,-1,0,0)
! Описать «невидимое» окно на сервере
END
CODE
SetWebActiveFrame()
! Сообщить объектам IBC
! (WebWindow), что нет активного
! фрейма приложения
WebFilesManager.Init(1,'')
!Инициализировать WebFilesManager
JavaEvents.Init
!Инициализировать JavaEvents
Broker.Init('HelloWeb', WebFilesManager)
!Инициализировать Broker
HtmlManager.Init(WebFilesManager)
!Инициализировать HtmlManager
WebServer.Init(Broker,'',600,'',WebFilesManager)
!Инициализировать WebServer
IF (WebServer.GetInternetEnabled())
! Если запущено брокером приложения
```

```
OPEN(ICServerWin)
! открыть «невидимое» окно на сервере
ACCEPT
IF (EVENT() = EVENT: OpenWindow)
WebServer.Connect
! Установить канал для брокера приложения
Hello
BREAK
END
END
ELSE
! Если запущено не брокером приложения
Hello
! Вызов Hello (режим Web)
END
WebServer.Kill
! Закрыть объект WebServer
HtmlManager.Kill
! Закрыть объект HtmlManager
Broker.Kill()
! Закрыть объект Broker
JavaEvents.Kill
! Закрыть объект JavaEvents
WebFilesManager.Kill
! Закрыть объект WebFilesManager
Hello PROCEDURE
Window WINDOW,AT(,,139,59),GRAY,DOUBLE
! описать окно
STRING('Hello Web!'),AT(51,14),USE(?Hello)
! со строкой Hello Web
BUTTON('Goodbye Web!'),AT(39,31),USE(?Bye)
! и кнопкой Goodbye Web
END
WebWindow WebWindowClass
! Описать объект WebWindow
```

```
CODE
OPEN(window)
! Открыть окно
WebWindow.Init(WebServer,HtmlManager)
! Инициализировать объект сбором информации об
! окне и его управляющих элементах
ACCEPT
IF WebWindow.TakeEvent/() THEN BREAK
! Обработка события Web: обрабатывает все
! события, необходимые для ответов на
! запрос клиента, например, создает новую
! страницу HTML
IF EVENT() = EVENT:Accepted
! Обычная обработка события Windows
POST(Event:CloseWindow)
! Закрыть окно по кнопке ?Bye
END
END
CLOSE(window)
! Закрыть окно
WebWindow.Kill
! Закрыть объект WebWindow
RETURN
WebControl Factory PROCEDURE(SIGNED Type)
!Создать экземпляры объектов WebControl
NewControl & WebControlClass
!запрошенные объектом WebWindow
CODE
CASE (Type)
OF CREATE:ClientArea
NewControl &= NEW WebClienAreaClabs
OF CREATE:String
NewControl &= NEW WebHtmlStringClass
OF CREATE:TextButton
NewControl &= NEW WebHtmlButtonClass
```

```
END
IF (~NewControl &= NULL)
NewControl.IsDynamic = TRUE
END
RETURN NewControl
ShutDownManager.Close PROCEDURE
CODE
WebServer.Kill
HtmlManager.Kill
Broker.Kill()
JavaEvents.Kill
WebFilesManager.Kill
```

Обсуждение проекта, закодированного вручную

Библиотека IBC требует несколько компонент для успешной компиляции и связи объектных модулей. Укажите следующие компоненты в диалоге **Project Editor**.

ICSTD.CLW

ICSTD.CLW содержит множество процедур, разделяемых несколькими различными объектами IBC. Прототипы для этих процедур находятся в ICSTD.INC.

Эти процедуры не являются классами или методами и, следовательно, не могут быть определены для загрузчика через атрибут **LINK** как методы IBC. Для того, чтобы указать на них загрузчику, необходимо добавить файл ICSTD.CLW в дерево проекта, в пункт с названием **External source files**. Файл ICSTD.CLW по умолчанию инсталлируется в Clarion-директорию **LIBSRC**.

Драйвер DOS-файлов

Объекты библиотеки IBC используют драйвер DOS-файлов для записи HTML-кода и данных JSL, запрашиваемых броузером клиента. Вы должны добавить драйвер DOS-файлов CLW в пункт дерева проекта с названием **Database**

drive libraries для разрешения ссылок IBC на процедуры драйвера DOS-файлов.

Драйвер ASCII-файлов

Объекты библиотеки IBC используют драйвер ASCII-файлов для выполнения отчета. Вы должны добавить драйвер ASCII-файлов CLW в пункт дерева проекта с названием **Database drive libraries** для разрешения ссылок IBC на процедуры драйвера ASCII-файлов.

CW2HTM32.LIB

CW2HTM32.LIB содержит множество откомпилированных объектов, разделяемых несколькими несколькими различными объектами IBC. Прототипы для этих исполняемых объектов находятся в ICSTD.INC. Для того, чтобы указать на них загрузчику, необходимо добавить файл CW2HTM32.LIB в дерево проекта, в пункт с названием **Library, object, and resourse files.**

Web-пригодные программы должны быть 32-х разрядными. По всем практическим соображениям, Web-пригодные программы должны быть 32-х разрядными. В Интернет-среде несколько клиентов могут одновременно обращаться к приложению, следовательно, программа должна поддерживать существование нескольких экземпляров на Web-сервере. 32-х разрядные программы позволяют это (в противоположность 16-ти разрядным).

Краткое руководство по IBC Library

Для обеспечения работы гибридных Web/Windows приложений, шаблоны Internet Connect Templates основываются на библиотеке Internet Builder Class (IBC) Library.

Классы и их шаблонно-сгенерированные объекты

Шаблоны Internet Connect создают экземпляры объектов из IBC Library. Имена объектов обычно похожи на соответствующие имена классов, но не совпадают полностью. В результате, сгенерированный код вашего Web-приложения выглядит примерно так:

```
Broker.Init
MainFrame.TakeEvent
IC:CurFrame.CopyControlsToWindow
WebWindow.OptionBorderWidth = 2
IC:CurControl.Init
IC:CurControl.DisabledAction = DISABLE:Show
WebMenuBar.SetBackground(16711680, '')
HtmlPreview.Init(WebServer, HtmlManager, PrintPre-
viewQueue)
```

Различные классы IBC и их экземпляры в шаблонах перечислены ниже таким образом, чтобы вы легче могли определить объекты IBC в сгенерированном коде приложения:

Internet Builder Class	Template Generated Object
BrokerClass	Broker
HtmlClass	HtmlManager
JslEventsClass	JavaEvents
TextOutputClass	Target
HttpClass	BrokerHttp
WebFilesClass	WebFilesManager, Broker.FILEs, HtmlManager.Files, Broker.Http.Files, JavaEvents.Files, WebServer.Files, WebWindow.Files, Target.Files
WebServerClass	WebServer
WebClientManagerClass	Broker.CurClient

WebFrameClass	MainFrame
	IC:CurFrame
WebWindowClass	WebWindow
WebControlClass	IC:CurControl
WebCaptionClass	WebCaption
WebClientAreaClass	WebClientArea
WebMenuBarClass	WebMenuBar
WebToolbarClass	WebToolBar
WebReportClass	HtmlPreview

Краткое руководство

BrokerClass(Broker)

Init (инициализирует объект BrokerClass)

Kill (закрывает объект BrokerClass)

ServerName (имя сервера)

WebClientManagerClass(Broker.CurClient)

IP (IP-адрес клиента)

HtmlClass (HTMLManager)

Init (инициализирует объект HTMLClass)

Kill (закрывает объект HTMLClass)

JslEventsClass (JavaEvents)

Init (инициализирует объект JslEventsClass)

Kill (закрывает объект JslEventsClass)

TextOutputClass (HtnlManager или Target)

WriteLine (пишет одну строчку текста)

HttpClass(Broker.Http)

GetCookie (получить cookie от клиента)

SetCookie (установить cookie для клиента)

SetProcName (установить имя защищенной области)

SetProgName (установить имя сервера)

WebFilesClass(WebFilesManager или Files)

GetAlias (возвращает HTML alias для файла)

Init (инициализирует объект .WebServerClass)

Kill (закрывает объект WebFilesClass)

SelectTarget (устанавливает общедоступный или секретный канал)

WebServerClass (WebServer)

Active (Web или Windows-режим)

CommandLine (параметры командной строки)

Connect (открывает канал соединения с Брокером)

Init (инициирует объект WebServerClass)

PageToReturn (возвращает URL)

ProgramName (путь на сервере)

Quit (завершение серверной программы)

SetSendWholePage (вызов полного обновления страницы)

SetNewPageDisable (подавление выходящих web-страниц)

TimeOut (период отсутствия активности, после которого происходит завершение задачи)

WebFrameClass(MainFrame или IC:CurFrame)

CopyControlsToWindow (помещение глобальных управляющих элементов в локальное окно)

FrameWindow (ссылка на приложение)

TakeEvent (обработать события Броузера и цикла ACCEPT)

13 Зак. 3215

WebWindowBaseClass(WebWindow)

AllowJava (генерировать или подавлять генерацию JavaScript)

BorderWidth (толщина обрамления Web-страницы)

CloseImage (изображение на кнопке Close)

CreateCaption (включает заголовок на web-страницу)

CreateClose (включает кнопку Close на Web-страницу)

DisabledAction (HTML по умолчанию для отключенных управляющих элементов)

FormatBorderWidth (толщина обрамления клетки HTML таблицы)

GroupBorderWidth (толщина обрамления группы элементов (group box))

Menubar (расположение меню)

OptionBorderWidth (толщина обрамления выбора из вариантов (option box))

SheetBorderWidth (толщина обрамления листа (sheet))

WebWindowClass(WebWindow)

AuthorizeArea (имя защищенной паролем Web-страницы)

HelpDocument (документ помощи (HTML))

HelpEnabled (флаг разрешения помощи(HTML))

HelpRelative (удаленный или локальный документ помощи)

IsSecure (общедоступный или секретный канал)

AddControl (добавление управляющего элемента)

CreateHtmlPage (генерация HTML для окна)

GetControlInfo (возвращает ссылку на управляющий элемент)

GetToolbarMode (возвращает Toolbar entity)

Init (инициализирует объект WebWindowClass)

Kill (закрывает объект WebWindowClass)

MenubarType (расположение меню)

SetBackground (установить фон Web-страницы)

SetPassword (требовать пароль)

SetSplash (сделать это окно splash-окном)

SetTimer (установить для Web-страницы таймер и действие)

SupressControl (не включать управляющий элемент на Web-страницу)

TakeEvent (обработать события Броузера и цикла ACCEPT)

ValidatePassword (проверить пароль)

WebControlClass(IC:CurControl)

DisableAction (HTML для отключенных управляющих элементов)

CreateHtml (записать HTML для управляющего элемента и его атрибутов)

Feq (номер управляющего элемента)

ParentFeq (номер родительского управляющего элемента)

Init (инициирует объект WebWindowClass)

SetBorderWidth (установить толщину обрамления)

WebJavaStringClass (IC:CurControl)

SetAutoSpokLink (установить «живые» гипертекстовые ссылки)

WebHtmlImageClass (IC:CurControl)

SetDescription (установить альтернативный текст для Web-изображения)

WebJavaListClass (IC:CurControl)

ResetFromQueue (записать изменения в очередь для LIST на сервере)

SetAutoSpotLink (установить «живые» гипертекстовые ссылки)

SetEventAcion (установить очередь — источник данных)

WebCaptionClass (WebCaption)

Alignment (выравнивание текста)

SetBackground (установить фон заголовка Web-страницы)

SetFont (установить шрифт заголовка Web-страницы)

WebClientAreaClass (WebClientArea)

SetBackground (установить фон области клиента Web-страницы)

WebMenubarClass (WebMenubar)

SetBackground (установить фон области меню Web-страницы)

WebToolbarClass (WebToolbar)

SetBackground (установить фон области панели инструментов Web-страницы)

WebReportClass (HtmlPreview)

Init (инициировать объект WebReportClass)

Kill (закрыть объект WebReportClass)

Preview (генерировать HTML для представления отчета)

Приложения

Глоссарий

Все определения являются общими понятиями, кроме специально помеченных.

Понятия, помеченные (Clarion) являются специфичными для языка Clarion и для Среды разработки Clarion.

applet

Небольшое, целевое приложение. Необязательно представляют собой отдельные исполняемые модули. Маленькие программы, написанные на Java, обычно называют апплетами. В HTML, <APPLET> означает Java applet.

Application Broker (Clarion)

Application Broker необходим для исполнения гибридных Web/Windows приложений Clarion. Application Broker запускает гибридное Web/Windows приложение на Internet — сервере и обновляет Clarion Java Support Library (JSL) в Броузере. Затем Application Broker организует трафик сообщений в сессии удаленного соединения, направляя события, порождаемые Java Support Library в гибридное Web/Windows приложение, а HTML — скрипты, порождаемые приложением — в Броузер.

Broker (Clarion)

Смотри Application Broker.

Client (Clarion)

Броузер клиента, запускающий посредством Application Broker гибридное Web/Windows приложение.

cookie

Информация, хранимая на машине клиента по запросу сервера.

default button

Командная кнопка, которая по умолчанию активизируется, когда пользователь нажимает кнопку ввода.

disabled

Окно, меню или управляющий элемент, видимый, но не способный получить управление.

enciption

Представление данных в зашифрованном виде, чтобы неавторизованный пользователь не смог получить доступ к данным в понятном представлении.

font

Основная часть названия файлов, связанных одним типом начертания. Например, «Times New Roman» — это имя шрифта, а «Times New Roman plain», «Times New Roman Italic», «Times New Roman Bold» — это стили, содержащиеся в отдельных файлах.

font style

Форматирование символа, применяемое к начертанию шрифта, например bold, italic или bold italic.

Gif image

Формат Graphics Interchange File (GIF). Формат изображений, популяризуемый Compuserve. В основном, при-

знанный за способность эффективного сжатия изображения, содержащего 256 или менее цветов. Внимание: везде, где вы используете слово «GIF», вы должны добавлять упоминание торговой марки: «GIF (Graphics Interchange File) является торговой маркой CompuServe Information Services».

global toolbar

Горизонтально или вертикально упорядоченная группа командных кнопок и/или других управляющих элементов, в основном предполагающая доступ к ней в продолжении всего времени выполнения программы.

hide

Предотвращает вывод на экран управляющего элемента или окна.

Управляющий элемент существует, но не виден на экране конечному пользователю.

HTML

Hyper-Text Markup Language-язык, используемый Интернет-броузерами для форматирования и отображения Web-страниц.

Hybrid Web/Windows Application

Гибридные Web/Windows приложения выглядят как стандартные Windows приложения при работе в Windows, но работают как Интернет сервера, когда запускаются из Application Broker.

Гибридные Web/Windows приложения могут управляться из любого Java-совместимого броузера (например, Microsoft Internet Explorer или Netscape Navigator).

icon (пиктограмма)

Графическое представление определенного физического объекта, присутствующего в системе (например, прин-

тера). Пиктограмма также может обозначать действие, понятие программы в случаях, когда появляется на командных кнопках. Обычно, файл с пиктограммой имеет расширение .ICO. Одним из основных свойств пиктограммы является встроенная поддержка прозрачности. Это позволяет отображать маленькие изображения без уничтожения фона.

include file

Внешний файл с исходным кодом, считываемый и препроцессируемый во время компиляции. В Clarion, Equates и других файлах в директории LIBSRC по умолчанию являются include файлами.

Internet Developer's Kit (Clarion)

Internet Developer's Kit является дополнительным продуктом, который может использоваться с Clarion Standard, Professional или Enterprise Edition для разработки новых гибридных Web/Windows приложений, или для приспособления к Web уже существующих приложений.

Версия Application Broker, поддерживающая одно соединение, включена в Internet Developer's Kit.

Java Support Library (Clarion)

Java Support Library (JSL) — это небольшое множество Java-классов (меньше 200k), которое обеспечивает в Интернет-броузере широкое множество Windows-подобных управляющих элементов. JSL порождает события из Интернет-броузера и обрабатывает сообщения от Интернет-сервера.

JPG image

Графический формат, поддерживающий 24-х разрядные цвета.

Обычно обеспечивает большое сжатие при потере разрешающей способности.

JSL data

Протокол и данные, которые гибридное Web/Windows приложение посылает в Интернет-броузер для обработки библиотекой Java Support Library (JSL). Гибридное Web/Windows приложение посылает JSL-данные в Интернет-броузер для достижения очень быстрого частичного обновления Web-страницы.

Remote Computing Session (Clarion)

Clarion Application Broker организует события, порождаемые Java Support Library (JSL) и HTML-страницы, порождаемые гибридными Web/Windows приложениями, в удаленную сессию, поддерживая состояние диалога между броузером и сервером.

Reusable Client (Clarion)

Java Support Library (JSL) — это небольшое множество Java-классов (менее 200 K), генерирующее события из Интернет-броузера и обрабатывающее сообщения от Интернет-сервера. Этот «тонкий» клиент может использоваться повторно любым гибридным Web/Windows приложением, минимизируя, таким образом, время соединения и требования к ресурсам локального броузера (RAM и дисковое пространство).

Server (Clarion)

Гибридное Web/Windows приложение, запускаемое Application Broker-ом по запросу Интернет-броузера.

Session Router (Clarion)

Session Router обеспечивает выполнение удаленной сессии в Интернет на нескольких Application Broker-ах в случаях, когда это требуется (например, в случаях высокой популярности приложения или других случаях необходимости развертывания дополнительных серверов Интернет).

Вы можете отдельно приобрести Session Router.

timer

Ресурс Windows, который может автоматически посылать сообщение приложению через предопределенные интервалы.

Ultra-thin Reusable Client (Clarion)

Java Support Library (JSL) — это небольшое множество Java-классов (менее 200 K), генерирующее события из Интернет-броузера и обрабатывающее сообщения от Интернет-сервера. Этот «тонкий» клиент может использоваться повторно любым гибридным Web/Windows приложением, минимизируя, таким образом, время соединения и требования к ресурсам локального броузера (RAM и дисковое пространство).

Содержание

Язык программирования Clarion

Универсальный язык 3
Формат программы и операторов 4
Формат предложений 5
Формат операторов 6
Резервируемые (служебные) слова 7
Специальные символы 9
Локальные параметры 18
Директивы компилятора 20
Объявление переменных 22
Обозначения шаблона 23
Оператор EXTERNAL 30
Структура GROUP 32
Неявные переменные 32
Массивы как параметры процедур и функций 35
Директивы компилятора 36
Выражения 38
Функции 38
Числовые константы 41
Числовые операторы 42
Числовые выражения 42

Символьные константы 43
Вычисление числовых и символьных выражений 44
Операторы конкатенации 45
Символьные выражения 45
Логические операторы 46
Логические выражения 47
Постоянные выражения 47
Функции в выражениях 48
Операторы присваивания 49
Оператор CLEAR 50
Правила преобразования данных 51

Clarion Internet Connect

Создаем приложения для Web 54
Программа установки 57
Брокер приложений 59
Удаленный доступ к брокеру приложений 72
Как создать Web-приложение 77
Оптимизация приложения 82
Добавление графики 86
Создание, размещение и запуск приложения 88
Преобразование готовых Clarion-приложений в Web-приложения 88
Дополнительная техника программирования для Web 93
Встроенный HTML 97

Добавление динамического HTML, используя переменную 98
Добавление статического HTML 99
Добавление условного HTML в исходный код Clarion 101
Защита операции подгрузки с помощью Splash-окна 103
Переименование BUTTON в Java Button 104
Центрирование поля Image в окне Splash 105
Применение частичного обновления к управляющим элементам Update 107
Ограничение доступа к процедуре 109
Ограничение редактирования по месту 112
Шаблоны Internet Builder Class 115
Global Internet Extention Template 115
Многодокументный интерфейс 121
Глобальные параметры компоненты окна 125
Распределенный шаблон процедуры для Internet 131
Индивидуальные переопределения настроек для управляющего элемента 139
Процедурные параметры компоненты «Окно» 145
MDI-параметры процедуры Frame 151
Dynamic Html Code Template 153
Static Html Code Template 153
GetCookie Code Template 154
SetCookie Code Template 154
Cookies (Persistent Client Data) 155

AddServerProperty Code Template 156
GetServerProperty Code Template 156
Дизайн Web-приложений 157
Соображения о косметическом дизайне 161
Разработка интерфейса пользователя 164
Соображения секретности 170
Использование встроенного HTML 173
Реализация помощи в ваших Web-приложениях 177
Управляющие элементы Windows и их HTML-эквиваленты 179
Программирование приложений вручную 184
Краткое руководство по IBC Library 190

Приложения

Глоссарий 197

Научно-популярное издание

Серия книг «Мой компьютер»

Дегтярев Иван Андреевич

Язык программирования Clarion 5.0: Неофициальное руководство пользователя по созданию приложений для Internet

Изд. лиц. ИД № 01844 от 22.05.2000 г. Подписано в печать 21.02.02. Формат 60x84/16. Бумага газетная. Печать офсетная. Усл. печ. л. 13. Заказ 3215—02.

Издатель Осипенко Александр Иванович / «МАЙОР»
111395, Москва, Молдагуловой, 16-2-215, тел. (095) 373-04-20.
E-mail: majorpub@mtu-net.ru