Дизайн Web - приложений: соображения

8

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. Этот раздел дает некоторые пояснения, но отнюдь не все. Он предназначен дя того, чтобы дать Вам пищу для размышлений во время разработки приложений.

Используйте частичное обновление, где возможно

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.

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.

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. Данное руководство 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 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 (т. е. 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 атрибура 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 раскраваются 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.WriteIN('<<IMG SRC=" '  &  SELF.FILES.GETAlias('mygif.gif') & ' ">')
```

найдет файл mygif.gif в любой директории, предоставленной серверному приложению.

Разработка интефейса 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 серверное 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 Clarion4 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 создаете 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
```

Примечание: **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 email 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 сервере. 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.

Примечание: **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 инсталлированый Digital Sertificate. Смотрите главу The Application Broker. 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.<</ >')
```

Примечание: **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="http://www.topspeed.com">Visit TopSpeed<</A>')
Target.WriteLn('<<A HREF="http://www.clariononline.com">Visit ClarionOnLine<</A>')
Target.WriteLn('<<A HREF="http://www.isetips.com">Visit IceTips<</A>')
Target.WriteLn('<<A HREF="http://www.finansist.com">Visit Finansist<</A>')
Target.WriteLn('<<A HREF="http://www.topspeed.com/tsnews.htm">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.

Примечание: **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 / и 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> ')
```

Примечание: **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>
</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 сооздаются 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 (строка-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. Для получения дополнительной информации смотрите Images (изображения).

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="file.gif">.

BOX

Не поддерживается—используйте встроенный HTML для изображения <IMG SRC="file.gif">.

ELLIPSE

Не поддерживается—используйте встроенный HTML для изображения <IMG SRC="file.gif">.

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 Proce dure 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 и т.д.). 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 и т.д.). Если Pre view 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. Эта глава представляет небольшую, написанную вручную программу "Hello Web", использующую IBC Library. В этой главе 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 WebClienAreaClass
     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. Для получения более полной информации смотрите The Project System в Руководстве пользователя.

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).