МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ

КЛИЕНТ-СЕРВЕРНЫЕ ПРИЛОЖЕНИЯ

Учебно-методическое пособие
по дисциплине
«ВЕБ-ПРОГРАММИРОВАНИЕ»

Набережные Челны
2018

Галиуллин Л.А. Клиент-серверные приложения: учебно-методическое пособие по дисциплине «Веб-программирование» [Электронный ресурс] / Казанский федеральный университет, Электронный архив, 2018.

Рассматривается проблемы проектирования и разработки клиент-серверных приложений. Приведены контрольные вопросы. Для студентов направлений подготовки «Информатика и вычислительная техника», «Программная инженерия».

Введение

Веб-программирование – раздел программирования, ориентированный на разработку веб-приложений (программ, обеспечивающих функционирование динамических сайтов Всемирной паутины).

Языки веб-программирования – это языки, которые в основном предназначены для работы с веб-технологиями. Языки веб-программирования можно условно разделить на две пересекающиеся группы: клиентские и серверные.

Как следует из названия, программы на клиентских языках обрабатываются на стороне пользователя, как правило, их выполняет браузер. Это и создает главную проблему клиентских языков – результат выполнения программы (скрипта) зависит от браузера пользователя. То есть, если пользователь запретил выполнять клиентские программы, то они исполняться не будут, как бы ни желал этого программист. Кроме того, может произойти такое, что в разных браузерах или в разных версиях одного и того же браузера один и тот же скрипт будет выполняться по-разному. С другой стороны, если программист возлагает надежды на серверные программы, то он может упростить их работу и снизить нагрузку на сервер за счет программ, исполняемых на стороне клиента, поскольку они не всегда требуют перезагрузку (генерацию) страницы. Самыми распространенными клиентскими языками программирования являются:

- HTML
- CSS
- JavaScript
- VBScript
- ActionScript
- Java
- CoffeeScript

Когда пользователь дает запрос на какую-либо страницу (переходит на неё по ссылке или вводит адрес в адресной строке своего браузера), то вызванная страница сначала обрабатывается на сервере, то есть выполняются все программы, связанные со

страницей, и только потом возвращается к посетителю по сети в виде файла. Этот файл может иметь расширения: HTML, PHP, ASP, ASPX, Perl, SSI, XML, DHTML, XHTML.

Работа программ уже полностью зависима от сервера, на котором расположен сайт, и от того, какая версия того или иного языка поддерживается. К серверным языкам программирования можно отнести: PHP, Perl, Python, Ruby, любой .NET язык программирования (технология ASP.NET), Java, Groovy.

Важной стороной работы серверных языков является возможность организации непосредственного взаимодействия с системой управления базами данных (или СУБД) – сервером, на котором упорядоченно хранится информация, которая может быть вызвана в любой момент. Популярными среди систем управления базами данных являются:

- Firebird
- IBM DB2
- IBM DB2 Express-C
- Microsoft SQL Server
- Microsoft SQL Server Express
- mSQL
- MySQL
- Oracle
- PostgreSQL
- SQLite
- Sybase Adaptive Server Enterprise
- ЛИНТЕР
- MongoDB

Язык HTML

За последние годы разработки для Интернета эволюционировали от статических страниц до динамических информационных систем. Некоторое время назад создание современных Web-страниц не требовало практически ничего, кроме совершенного владения языком разметки гипертекста (Hypertext Markup Language, HTML). HTML представляет собой простой язык обработки текстов; на этом языке при помощи набора *тегов* (tags) создается документ, который можно

просматривать специальной программой просмотра Web (browser). Так, HTML-код из листинга 1.1 создает простую Web-страницу.

Листинг 1.1. *Исходный код простой Web-страницы.*

```
<HTML>
<HEAD><TITLE>My First Web Page</TITLE></HEAD>
<BODY BGCOLOR="WHITE">
<H2><CENTER>Добро пожаловать на мою первую Web-страничку! </CENTER></H2>
</BODY></HTML>
```

HTML – не язык программирования в том смысле, как C++ или Visual Basic; он больше напоминает средства форматирования документов с использованием управляющих последовательностей. Кодирование на HTML часто сравнивают с созданием документа в формате Microsoft Word путем набивки кодов форматирования прямо в Notepad. Очевидно, что функциональность этого крайне мала.

По многим причинам HTML – бедный язык с точки зрения программирования. Во-первых, возьмем *гиперссылки* (hyperlinks) – эти подчеркнутые и выделенные голубым цветом слова, которые Вы щелкаете, чтобы перейти к другой странице. Гиперссылка – это, по сути, облагороженный оператор перехода GOTO, обеспечивающий переход к жестко указанному месту приложения. Об операторе GOTO и его недостатках написана масса статей. Жестко закодированные ссылки порождают код, который очень трудно сопровождать, и если вы когда-либо писали HTML-код, то знаете, как трудно его повторно использовать и модифицировать.

Во-вторых, HTML не предоставляет никакой реальной возможности сохранять данные в процессе работы приложения. Да и вообще, в этом случае трудно даже говорить о приложении в Web. Когда каждая страница представляет собой лишенную состояния транзакцию с сервером, как вообще можно определить, где приложение начинается и где заканчивается? Сравните это с типичным клиент-серверным приложением, о начале работы которого сигнализирует двойной щелчок значка на рабочем столе, а о конце — выбор пункта **Exit** в меню **File.**

В-третьих, у HTML очень ограниченные возможности для

взаимодействия. Стандартный HTML довольствуется статическими Web-страницами с текстом, рисунками и ссылками на другие страницы. Подобные узлы называют *Желтыми Страницами* WWW (World Wide Yellow Pages), так как их формат очень напоминает страницы телефонной книги.

Разумеется, HTML обеспечивает некоторую интерактивность при помощи *встроенных элементов управления* (intrinsic controls) — тех самых полей ввода, которые обычно присутствуют в HTML-формах. Простые формы можно создать, например, при помощи тегов <INPUT>. Тег <INPUT> допускает применение *текстовых полей* (text boxes), *флажков* (check boxes), *переключателей* (radio buttons) и *кнопок* (buttons). Листинг 1.2 определяет HTML-форму, которая содержит текстовые поля ввода для имени, номера телефона и адреса электронной почты.

Листинг 1.2. *Код для HTML-формы.*

```
<HTML><HEAD><TITLE>Simple HTML Form</TITLE></HEAD>
<B0DY BGCOLOR="WHITE">
<FORM>
<INPUT TYPE="TEXT" NAME="txtName">Имя<P>
<INPUT TYPE="TEXT" NAME="txtPhone">Телефон<P>
<INPUT TYPE="TEXT" NAME="txtEMail">Адрес электронной почты<P>
</FORM></BODY></HTML>
```

Формы представляют собой простейшее средство взаимодействия с HTML. Пользователь заполняет ряд форм, которые затем отсылаются серверу. В процессе пересылки данные преобразуются в некий заранее определенный формат и отсылаются в текстовом виде исполняемому файлу сервера. После этого процесс на сервере может использовать полученные данные, например, для доступа к базам данных, посылки почтового сообщения или выполнения иных функций.

HTML представляет собой обычный текст, поэтому первоначально большинство разработчиков писали свои программы непосредственно в текстовых редакторах, таких как Notepad. Со временем ряд фирм предложили графические средства разработки, например, Microsoft FrontPage, дающие

возможность создавать Web-страницы, не зная в явном виде HTML. Эти графические редакторы позволяют непосредственно макетировать Web-страницы без трудоемкой возни с тегами. К сожалению, мощь подобных графических редакторов оборачивается и серьезным их недостатком: у разработчиков создается впечатление, что им ни к чему изучать синтаксис и теги HTML, — а между тем трудно придумать что-нибудь более далекое от истины, чем это утверждение. Если вы даже ничего больше не вынесете из этого краткого введения в HTML, то запомните хотя бы это: чтобы быть настоящим Web-разработчиком, вы ***должны*** знать HTML. Навыки редактирования страницы непосредственно в виде исходного текста позволят вам добиться желаемого эффекта независимо от того, поддерживает ли его ваш любимый графический редактор.

Программирование на стороне клиента

В своих первых попытках повысить интерактивность HTML Web-страниц разработчики обратились к *сценариям* (scripting), добавляя функциональность путем комбинирования языка программирования с HTML. В результате зачастую получается странный гибрид кода и тегов, что вынуждает разработчиков вернуться к текстовым редакторам. Был введен специальный тег <SCRIPT>, который определяет раздел кода на Web-странице. Код из листинга 1.3 при помощи VBScript создает пример, выводящий приветствие «Hello, World!».

Листинг 1.3. *Код на VBScript, выводящий «Hello, World!».*

```
<HTML><HEAD>
<META HTTP-EQUIV="Content-Type content=text/html, charset=windows-1251">
<TITLE>Yet Another Hello, World! Example</TITLE>
<SCRIPT LANGUAGE="VBScript”> <!--
Sub cmdClickMe_OnClick()
MsgBox "Hello, World!"
End Sub
-->
</SCRIPT>
</HEAD>
```

<BODY BGCOLOR= WHITE >
<FORM>
<INPUT TYPE= BUTTON NAME= cmdClickMe VALUE="Click Me!”>
</FORM></BODY></HTML>

VBScript представляет собой язык описания сценариев, в основе которого лежит Visual Basic for Applications (VBA), популярный язык, применяемый, например, в Microsoft Office 97. VBScript — это не полная версия VBA, а скорее его подмножество, которое сохраняет многие ключевые возможности VBA, но в то же время не реализует те, которые сделали бы его чересчур громоздким и небезопасным. Так, VBScript не поддерживает типы данных: все переменные объявляются как Variant.

Как и его старший брат, VBA, язык VBScript управляется событиями. Это означает, что написанный Вами код выполняется в ответ на *событие* (event), возникшее в результате взаимодействия пользователя с графическим интерфейсом (graphical user interface, GUI). В нашем случае GUI представляет собой Web-страницу. Так, в приведенном выше примере, когда пользователь взаимодействует с GUI, нажимая кнопку «Click Me!», это действие вызывает событие OnClick. Это событие, в свою очередь, обрабатывается кодом на VBScript, организованным в виде процедуры обработки события. Имена таких процедур имеют вид *ИмяЭлементаУправления_ИмяСобытия,* представляя собой произвольные комбинации из имен элементов управления и событий.

Хотя сценарии и представляют собой шаг вперед в развитии интерактивности, у них есть и определенные ограничения. Например, не все программы просмотра распознают и обрабатывают сценарии, а те, которые это делают, используют разные языки. Главным образом это касается Netscape Navigator, который не распознает VBScript, однако работает с JavaScript — языком описания сценариев, первоначально разработанным для Netscape Navigator. По функциональности JavaScript очень похож на VBScript, но по синтаксису эти языки сильно различаются. В отличие от VBScript, JavaScript не поддерживает

концепцию процедур обработки событий. Все процедуры в JavaScript — это функции, вызываемые при помощи атрибутов событий, расположенных в HTML-теге. JavaScript-версия предыдущего примера на VBScript представлена в листинге 1.4.

Листинг 1.4. *Код на JavaScript, выводящий «Hello, World!».*

```
<HTML><HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html;charset=windows-1251">
<TITLE>JavaScript Hello, World! Example</TITLE>
<SCRIPT LANGUAGE="JavaScript"> <!--
function clickme() {
alert("Hello, World!");
return true; }
-->
</SCRIPT>
</HEAD><BODY BGCOLOR="WHITE">
<FORM>
<INPUT TYPE="BUTTON" NAME="cmdClickMe"
VALUE="ClickMe!" OnClick="var rtn=clickme();">
</FORM> </BODY> </HTML>
```

Плохо не только то, что поддержка сценариев различается в разных программах просмотра, но и то, что сценарии не обеспечивают всей развитой функциональности, которой ожидают от языка программисты. Сценарии предоставляют подмножество тех языковых возможностей, которые обычно используют разработчики: базовые структуры и операторы — циклы и ветвления. По сути, сценарии годятся только для проверки корректности введенных данных перед отсылкой формы на сервер.

Как только к возможностям программ просмотра добавляются сценарии, возрастает сложность клиентской платформы. Очевидно также, что раз отсутствует универсальный язык описания сценариев, то теряются все разрекламированные преимущества платформенной независимости Web. Для многих Web-мастеров и разработчиков постоянная война между программами просмотра за преобладание на рынке создает необходимость поддерживать две версии Web-узла: для Microsoft Internet Explorer и для

Netscape Navigator.

Компоненты ActiveX

По мере совершенствования технологии программ просмотра зависимость от платформы усилилась — она была привнесена компонентами ActiveX, технологией, основанной на COM — модели многокомпонентных объектов Microsoft (Component Object Model). Компоненты ActiveX варьируются от причудливых элементов управления, таких как движки (spinners), до невизуальных компонентов, обеспечивающих доступ к базам данных или электронной почте. Подобные компоненты делают страницы в Internet Explorer более функциональными и привлекательными, но практически бесполезными в среде, не поддерживающей ActiveX, например, в Netscape Navigator.

Компонент ActiveX добавляется в Web-страницу при помощи тега <OBJECT>, однозначно определяющего компонент для программы просмотра. Приведенный ниже код, используя тег <OBJECT>, помещает на Web-страницу элемент управления ActiveX — метку (label).

```
<OBJECT ID="Label1" WIDTH=291 HEIGHT=41
CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-
00AA003F40D0"
CODEBASE="http://www.microsoft.com/activex/controls/FM20.
DLL">
<PARAM NAME="ForeColor" VALUE="65408">
<PARAM NAME="VariousPropertyBits"
VALUE="276824091">
<PARAM NAME="Caption" VALUE="Щелкни меня!">
<PARAM NAME="Size" VALUE="7691;1094">
<PARAM NAME="SpecialEffect" VALUE="1">
<PARAM NAME="FontEffects" VALUE="1073741827">
<PARAM NAME="FontHeight" VALUE="480">
<PARAM NAME="FontCharSet" VALUE="204">
<PARAM NAME="ParagraphAlign" VALUE="3">
<PARAM NAME="FontWeight" VALUE="700">
</OBJECT>
```

Значения GUID в любой операционной системе хранятся в *реестре* — централизованной базе данных, которая отвечает за поддержание информации о программных объектах, используемых приложениями. Когда Internet Explorer обнаруживает тег <OBJECT>, он обращается к реестру и ищет там GUID, совпадающий со значением атрибута CLASSID. Когда такой GUID найден, из реестра выбирается дополнительная информация, позволяющая отыскать файл, который соответствует данному элементу управления ActiveX.

В теге <OBJECT> можно выделить несколько ключевых составных частей, которые определяют, как именно компонент ActiveX будет размещен на странице. Атрибут ID задает имя элемента управления, посредством которого ко всем его свойствам, методам и событиям можно будет получить доступ из текста сценария.

CLASSID представляет собой буквенно-цифровой код, который однозначно идентифицирует данный компонент ActiveX среди всех остальных. Этот код, известный как глобально уникальный идентификатор (Globally Unique Identifier, GUID), не использует больше ни один компонент ActiveX. При помощи GUID Internet Explorer однозначно определяет требуемый компонент и создает его на странице. Если нужный элемент управления ActiveX на клиентской машине отсутствует, Internet Explorer обращается к атрибуту CODEBASE за информацией о том, где находится этот элемент на сервере. Следуя этой информации, файлы данного элемента управления загружаются с сервера, и элемент устанавливается на клиентской машине. Теперь Internet Explorer может свободно работать с ним.

Доступ к компонентам ActiveX посредством тега <OBJECT> не ограничивается элементами управления. Этот тег может активизировать *произвольный* компонент ActiveX, в том числе и те компоненты, которые можно написать на языках Visual Basic, C++ и Microsoft FoxPro. В сущности, вы легко можете расширить функциональность, доступную клиенту, написав свои собственные компоненты ActiveX и загрузив их в программу просмотра. Следует, правда, помнить, что Internet Explorer по умолчанию не загружает и не выполняет компоненты без

цифровой подписи разработчика.

Окончательное обеспечение компонента данными происходит через тег <PARAM>, имеющий атрибуты NAME и VALUE, при помощи которых задаются начальные значения свойств данного компонента, когда он впервые создается на Web-странице. После того, как начальные значения установлены, значения свойств легко изменить во время выполнения из текста сценария.

```
<SCRIPT LANGUAGE="VBScript"><!--
Sub Label1_DblClick(Cancel)
Label1.Font.Weight=24
Label1.Caption="Щелкни снова!"
end sub
Sub Label1_Click()
Label1.AutoSize = false
Label1.Font.Weight = 30
Label1.Caption="Еще два раза!!!!!"
Label1.SpecialEffect=1
end sub
-->
</SCRIPT>
```

Документы ActiveX

Visual Basic, начиная с версии 5.0, позволяет, помимо элементов управления ActiveX, создавать документы ActiveX. Документы ActiveX представляют собой программные объекты, которые могут загружаться и работать внутри ActiveX-контейнера, такого как Internet Explorer. Документы ActiveX позволяют разработчикам на Visual Basic немедленно применить свой опыт работы на Visual Basic для создания приложений для Интернета. Что самое существенное, документы ActiveX предоставляют доступ к большей части ключевых возможностей Visual Basic в загружаемом формате.

Java

Не будем забывать и о Java! Этот язык очень быстро приобрел популярность, и его поддерживают как Internet Explorer, так и Netscape Navigator. Апплеты, разработанные на Java при помощи таких средств, как Microsoft J++, во многом напоминают компоненты ActiveX: это самодостаточные, загружаемые фрагменты Web-страницы. Так же, как и у компонентов ActiveX, у апплетов имеется свой особый тег — <APPLET>, который дает программе просмотра указание загрузить код на Java и выполнить его. Нижеследующий код исполняет апплет на Web-странице:

<APPLET CODE="DBLBULB.CLASS" HEIGHT=35 WIDTH=26> </APPLET>

Атрибут CODE тега <APPLET> идентифицирует исходный код апплета Java практически так же, как атрибут CODEBASE определяет источник для компонента ActiveX. У апплетов могут также быть теги <PARAM>, задающие начальные значения. Во многих случаях апплеты представляют собой функциональные эквиваленты элементов управления ActiveX. Во всяком случае, языки описания сценариев могут обращаться к открытым функциям апплетов точно так же, как они обращаются к методам компонентов ActiveX.

Контрольные вопросы

1. Какие серверные языки веб-программирования Вы знаете?
2. Какие СУБД Вы знаете?
3. Как на HTML создать объект «кнопка»?
4. Как на HTML создать объект «текстовое поле»?
5. Как на HTML создать объект «флажок»?
6. Как на HTML создать таблицу 3x3?
7. Что делает SQL-команда «SELECT»?
8. Что делает SQL-команда «INSERT»?
9. Что делает SQL-команда «DELETE»?
10. Что делает SQL-команда «UPDATE»?

Рекомендуемые источники

1. Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие / В.Д. Колдаев; Под ред. Л.Г. Гагариной. - М.: ИД ФОРУМ: ИНФРА-М, 2017. - 416 с. [Электронный ресурс]. http://znanium.com/bookread.php?book=336649.
2. Гагарина Л.Г. Технология разработки программного обеспечения: Учеб. пос. / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул; Под ред. проф. Л.Г.Гагариной - М.: ИД ФОРУМ: НИЦ Инфра-М, 2016. - 400 с. [Электронный ресурс]. http://znanium.com/bookread.php?book=389963.
3. Голицына О. Л. Программирование на языках высокого уровня: Учебное пособие / О.Л. Голицына, И.И. Попов. - М.: Форум, 2017. - 496 с. [Электронный ресурс]. http://znanium.com/bookread.php?book=139428

. …