МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ

ТЕХНОЛОГИЯ ASP

Учебно-методическое пособие
по дисциплине
«ВЕБ-ПРОГРАММИРОВАНИЕ»

Набережные Челны
2018

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

Рассматривается реализация современных web-приложений на базе технологии ASP. Представлены объекты и компоненты, файл GLOBAL.ASA, компоненты ActiveX. Приведены контрольные вопросы. Для студентов направлений подготовки «Информатика и вычислительная техника», «Программная инженерия».

Введение

Dynamic HTML представляет собой основное средство программирования клиента для Microsoft Internet Explorer 4.0 и выше, но такие программы просмотра Web, как Netscape Navigator, не поддерживают Dynamic HTML. На самом деле очень малая часть функциональности клиентской части, поддерживаемой различными программами просмотра, может рассматриваться как действительно кросс-платформенная.

Если Вы хотите разработать Интернет-узел, открытый для доступа самым различным программам просмотра, то должны перенести программирование с клиента на сервер. Такую возможность предоставляют Microsoft ASP (Active Server Pages — активные серверные страницы). По сути ASP не что иное, как сценарий на VBScript, который исполняется на сервере. Когда запрашивается страница, этот сценарий порождает HTML-текст. Это ключевая особенность ASP — клиент никогда не видит вашего кода, а только результирующий HTML, который воспринимает любая программа просмотра.

Объекты и компоненты

На самом простом уровне создание ASP-страницы — это не что иное, как написание серверного кода для получения ожидаемого результата. Но VBScript не является полнофункциональным языком и, как только Вы приступаете к построению более сложных страниц, его выразительных средств начинает не хватать. Так, в VBScript нет встроенных функций доступа к данным; не умеет он и открывать текстовые файлы. Собственно говоря, в VBScript отсутствуют какие-либо встроенные средства доступа к каким бы то ни было внешним источникам данных. Так как же в таком случае при помощи ASP выполняются такие сложные действия, как доступ к данным? Ответ будет таким: нужно дополнить VBScript *объектами* и *компонентами* ASP.

ASP-объекты и компоненты — это не что иное, как компоненты ActiveX, подобные обычным DLL ActiveX, с

которыми Вы наверняка работали в Microsoft Visual Basic. Различие между объектами и компонентами ASP состоит в том, каким образом они появляются в программе. ASP-объекты — это элементы управления ActiveX, которые в коде на VBScript доступны всегда: их не нужно создавать явно. В число объектов ASP входят Application, Session, Request, Response и Server. А вот ASP-компоненты представляют собой DLL, существующие вне структуры ASP. Эти компоненты могут быть написаны на любом языке, а некоторые полезные ASP-компоненты просто поставляются в комплекте с Visual InterDev. ASP-компоненты нужно явно создавать в коде. ASP поддерживает компоненты Database Access, File Access, Browser Capabilities, Ad Rotator и Content Linking.

Файл GLOBAL.ASA

Одна из главных трудностей разработчика для Интернета, независимо от того, какую технологию он использует, состоит в том, как сложно создать в Интернете настоящее *приложение.* Взаимодействие программы просмотра и Web-сервера представляет собой по сути лишенную состояния транзакцию, в ходе которой сервер посылает клиенту Web-страницу и затем забывает о его существовании. Когда клиент запрашивает другую Web-страницу, сервер ничего не помнит о предыдущем запросе. Коренная проблема для всех Web-приложений такова: как показать, что это именно приложение?

Определить приложение в среде Microsoft Windows довольно просто. Приложение запускается двойным щелчком значка и завершается, когда в меню File выбран пункт Exit. В промежутке между двумя этими событиями данные хранятся в переменных. Но для Интернет-приложений это не так. Как определить, когда приложение начинается, а когда заканчивается? Можно сказать, что приложение начало работу, если пользователь зашел на узел и просматривает одну из его страниц. Но что если он переходит к другому узлу, а через пять минут возвращается? Приложение все еще активно? А если пользователь отсутствовал час или два?

Проблема определения моментов запуска и завершения приложения оказывает серьезное влияние на правильное

управление переменными и последовательностью выполнения. К счастью, ASP предлагает решение. Оно состоит в том, что для определения начала и завершения — как всего приложения, так и отдельных пользовательских сессий — используется специальный файл под названием GLOBAL.ASA. На этот файл возложено реагирование на четыре ключевых события узла: Application_OnStart (запуск приложения), Application_OnEnd (завершение приложения), Session_OnStart (начало сессии) и Session_OnEnd (завершение сессии). В листинге 1 приведен типичный файл GLOBAL.ASA.

Листинг 1. *Файл GLOBAL.ASA.*

```
<SCRIPT LANGUAGE="VBSCRIPT" RUNAT="Server">
' В этот файл можно добавить обработчики событий ASP.
' Для создания обработчиков внесите в файл подпрограмму с именем,
' соответствующим событию, на которое Вы бы хотели среагировать.
'Название события Описание
'SessionOnStart Происходит, когда пользователь в первый раз вы-
'     полняет любую страницу Вашего приложения
SessionOnEnd Происходит, когда превышен лимит времени,
'     в течение которого пользователь не обращается
'     к страницам Вашего приложения, или если имел
'     место явный выход
'Application_OnStart Происходит один раз, когда любой
'     пользователь впервые выполняет первую страницу
'     Вашего приложения
'Application_OnEnd Происходит один раз при остановке Web-сервера
Sub SessionOnStart
End Sub
Sub Session_OnEnd
End Sub
Sub ApplicationOnStart
End Sub
Sub Application On_End
```

End Sub
</SCRIPT>

Для обозначения разделов сценария GLOBAL.ASA содержит теги <SCRIPT>. Эти теги имеют особый атрибут RUNAT=Server, который означает, что содержащийся в теге код на VBScript должен исполняться на сервере, а не на клиенте. Функционально RUNAT=Server означает то же, что и сочетания угловых скобок и знака процента, используемые для обозначения серверного сценария на Web-страницах. Обработка стандартных событий на сервере записывается в GLOBAL.ASA в стандартном синтаксисе. Например, обработку запуска приложения выполняет следующий фрагмент кода:

<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Application_OnStart
' Конкретный код приложения
End Sub </SCRIPT>

Хотя GLOBAL.ASA отмечает начало и завершение приложения при помощи событий, остается неясным, что же все-таки составляет собственно приложение. Одна из рабочих формулировок, предложенная Microsoft, определяет Интернет-приложение как виртуальный каталог со всеми его файлами. Если пользователь запрашивает Web-страницу из виртуального каталога под названием Bookstore, то тем самым он запускает приложение Bookstore, и в GLOBAL.ASA возбуждаются события Application_OnStart и Session_OnStart.

Согласно этому определению с приложением одновременно могут работать несколько программ просмотра. Но событие Application_OnStart происходит только один раз: когда первый пользователь запрашивает Web-страницу из виртуального каталога. Когда затем страницы из этого ката-лога запрашивают другие пользователи, возбуждается только событие Session_OnStart.

В то время как *приложение* может относиться к нескольким программам просмотра, обращающимся к одному и тому же множеству Web-страниц, *сессия* касается какой-то одной программы просмотра, обращающейся к тем же Web-страницам. Для конкретной программы просмотра сессия длится, пока программа продолжает запрашивать страницы виртуального

каталога. Если же пользователь не запрашивает Web-страницы (из данного виртуального каталога) на протяжении 20 минут (по умолчанию), сессия завершается, и возбуждается событие Session_OnEnd. Когда в данном виртуальном каталоге завершаются все сессии, возбуждается событие Application_OnEnd.

В качестве примера рассмотрим следующий сценарий. Два пользователя намереваются посетить на Web-узле приложение Magazine. Пользователь 1 оказывается проворнее и быстренько запрашивает Web-страницу DEFAULT.ASP. Тут же возбуждаются события Application_OnStart и Session_OnStart. Буквально пятью минутами позже к приложению обращается пользователь 2. Поскольку пользователь 1 как-то проявлял себя в течение последних 20 минут, приложение Magazine активно. Следовательно, возбуждается только событие Session_OnStart, сигнализируя о начале новой сессии. Кроме того, теперь для завершения приложения необходимо, чтобы завершились обе сессии.

В течение следующих 15 минут пользователь 1 не запрашивает никаких страниц приложения Magazine. Поскольку он не проявлял активности на протяжении 20 минут, ASP приходит к выводу, что пользователь 1 закончил свою работу с приложением, и возбуждает событие Session_OnEnd. Но приложение все еще активно, поскольку в течение последних 20 минут к нему обращался пользователь 2.

Пользователь 2 работает с приложением еще час, то и дело запрашивая новые Web-страницы. Но в конце концов он отключается, а через 20 минут после того, как он покинул узел (точнее, в последний раз запросил Web-страницу приложения), возбуждается событие Session_OnEnd. Поскольку пользователь 2 был последним пользователем данного приложения, оно завершается, и возбуждается событие Application_OnEnd.

Объекты

В ASP есть несколько встроенных объектов, которые доступны разработчику. Эти объекты помогают управлять многими вещами: от переменных, до передачи форм. Работать с

ними легко, они вызываются из кода напрямую без какого-то особого синтаксиса.

Объект Application

Объект Application (приложение) позволяет создавать *переменные приложения* (application variables) - переменные, доступные всем пользователям данного приложения. Все, кто обращается к Web-страницам данного виртуального каталога, могут совместно использовать любую переменную приложения определенную для этого каталога.

В листинге 2 приведен пример программы, которая использует Объект Application. В нем переменная приложения служит для отслеживания времени последнего обращения к страницам приложения.

Листинг 2. *Объект Application.*

```
<%@ LANGUAGE ="VBScript"%>
<html><head><title>Application Variables</TITLE>
</HEAD><BODY><CENTER>
Эта страница последний раз посещалась:
<%=Application("Time")%>
<% Application.Lock
Application("Time")=Now
Application.UnLock %>
</BODY></HTML>
```

Создание переменной приложения сводится к адресации объекта Application именем новой переменной, которую вы хотите создать. Например, следующий код создает новую переменную приложения с именем Company и присваивает ей значение NewTech.

```
Application("Company")="NewTech"
```

Имя может быть произвольным, а переменная может содержать любую информацию, будь то число или текст.

Поскольку такие переменные доступны нескольким пользователям одновременно, вы не сможете гарантировать, что два пользователя не попытаются одновременно присвоить одной и той же переменной разные значения. Для разрешения подобных коллизий объект Application поддерживает методы

Lock и UnLock. Метод Lock блокирует весь объект Application, а не только переменную, вы хотите изменить, поэтому сразу же после изменения значения переменной разблокируйте объект:

```
Application.Lock
Application("Company")="NewTech"
Application.UnLock
```

Несмотря на то, что переменные приложения пригодны для временного хранения данных, их нельзя использовать для долговременного хранения. Все данные в переменных приложения уничтожаются, когда происходит событие Application_OnEnd. Так что, если вы хотите, чтобы они сохранились после завершения приложения, позаботьтесь о их переносе в файл или базу данных.

Объект Session

Зачастую разработчиков меньше интересуют данные, совместно используемые несколькими пользователями, зато гораздо больше - данные, связанные с конкретным пользователем. ASP поддерживает переменные для индивидуального пользователя при помощи объекта Session (сессия), который позволяет создавать *переменные сессии* (session variables).

Листинг 3 демонстрирует, как определить несколько переменных сессии в файле GLOBAL.ASA. Само по себе их определение так же просто, как и в случае переменных приложения. Все, что нужно сделать - это адресовать объект Session именем переменной, которую вы хотите создать. Основное различие между переменными этих объектов - их области видимости. Переменные сессии предназначаются для одного пользователя и живут, пока пользователь поддерживает сессию. Как только в течение 20 минут (по умолчанию) пользователь не обращается к страницам данного виртуального каталога, данные пропадают.

Листинг 3. *Создание переменных сессии.*

```
<SCRIPT LANGUAGE="VBSCRIPT" RUNAT="Server">
Sub Session_OnStart
Session("Company")="NewTech"
```

```
Session("Email")="info@newtech.com"
End Sub
</SCRIPT>
```

Переменные сессии можно создавать на любой Web-странице или в файле GLOBAL.ASA, а доступны они на любой Web-странице приложения, в котором эти переменные были первоначально созданы. Получить значения переменных сессии можно, считывая их из объекта Session. Следующий фрагмент кода считывает переменные сессии, созданные в листинге 4.4, и выводит их в полях ввода:

```
<FORM>
<P><INPUT TYPE="TEXT" VALUE=<%=Session("Company")%>Компания</P>
<P><INPUT TYPE="TEXT" VALUE=<%=Session("Email")%>Эл. Почта</P>
</FORM>
```

Ранее мы определили Интернет-приложение как лишенные статуса транзакции между Web-сервером и программой просмотра. Как же тогда ASP запоминает переменные сессии для каждого пользователя приложения? Ответ будет таким: эти переменные сохраняются на сервере для каждого клиента. Программа просмотра получает от сервера уникальный идентификатор, позволяющий определить, какой набор переменных кому принадлежит. Клиент этот идентификатор (Globally Unique Identifier, GUID) сохраняет, а впоследствии посылает серверу и получает именно ему предназначенные данные. Таким образом каждый клиент может иметь свой набор данных в каждом Интернет-приложении.

Осталось сказать, что для установки или считывания впемени жизни сессии (в минутах) применяется свойство Timeout объекта Session:

```
Session.Timeout=30
```

Объект Request

Для передачи данные клиенту создается Web-страница, а для передачи данных в обратном направлении программа просмотра использует отправку формы (form submission). В форме

содержатся текстовые поля, переключатели и т.п. Клиент размещает введенные данные в этих полях и пересылает пакет серверу. Процессом передачи формы управляют два атрибута тега <FORM>: METHOD и ACTION. Первый атрибут - METHOD - определяет, каким именно образом данные пересылаются серверу. Атрибут может иметь два значения: POST и GET. POST диктует программе просмотра, что данные нужно поместить внутрь формы, а GET пересылает данные как составную часть URL целевой страницы. Второй атрибут - ACTION - задает целевую страницу для обработки отправленных данных. Следующий код посылает все данные формы сценарию DATA.ASP методом POST:

```
<FORM METHOD="POST" ACTION="/l5/data.asp">
<P><INPUT TYPE="TEXT" NAME="Name"></P>
<P><INPUT TYPE="TEXT" NAME="EMail"></P>
<P><INPUT TYPE="SUBMIT"></P>
</FORM>
```

Элемент формы с типом SUBMIT - это кнопка, нажатие которой пользователем заставляет программу просмотра упаковать данные формы и отправить их. Формат пересылки данных определен строго и сервер знает, чего ожидать от клиента. Данные имеют вид пар *Поле=Значение*, отсылаемых серверу в формате открытого текста. Если в предыдущем примере ввести в поле Name *NewTech* и info@newtech.com в поле Email, то сценарию DATA.ASP будет послан следующий текст:

```
Name=NewTech&Email=info@newtech.com
```

На сервере эти данные можно вновь разобрать по полям и использовать в любых целях. Вот тут-то и нужен объект Request. Он используется в ASP для разбора полученных от клиента данных. Для работы с объектом Request просто сообщите ему имя поля, значение которого хотите получить, и объект вернет вам это значение. Например, следующий код вернет вам значение поля Name:

```
<%=Request.Form("Name")%>
```

Request.Form применяется, когда данные были отправлены методом POST и именно этому сценарию. Если для отправки данных используется метод GET или сценарий вызывается с

передачей параметров прямо в гиперссылке

<A HREF=/l5/data.asp?Name=NewTech&Email=info@newtech.com> Чтобы отправить данные, щелкните здесь!</A>

то для разбота данных применяют свойство Request.QueryString, который работает так же, как Request.Form. Следующий фрагмент кода вернет значение поля Name из гиперссылки:

<%=Request.QueryString("Name")%>

Другое свойство - Request.Cookies используются для извлечения информации из кукисов (cookies), отосланных вместе с запросом строке пользовательского агента программы просмотра. А листинг 4 демонстрирует применение свойства ServerVariables для определения имени компьютера, с которого клиент вызвал сценарий.

Листинг 4. *Определение компьютера пользователя.*

```
<%@ LANGUAGE="VBSCRIPT"%>
<html><head><title>Server Variables</TITLE>
</HEAD><BODY><CENTER>Вы вошли с компьютера
<%=Request.ServerVariables("Remote_Host")%>
</BODY></HTML>
```

Переменные сервера представляют широкий круг информации о клиенте и Web-сервере. Доступ к каждой конкретной переменной сводится к чтению соответствующей переменной.

Объект Response

Этот объект управляет содержимым страницы, которую ASP возвращает программе просмотра. Фактически в комбинации <%=переменная%> знак равенства представляет собой сокращенное обозначение метода Write объекта Response. Так что следующие две строки кода эквивалентны:

```
<%="NewTech"%>
<%Response.Write "NewTech"%>
```

Поскольку объект Response используется очень часто, такое сокращение оправдано.

Полезное свойство объекта Response - Expires. Оно задает

время (в минутах) за которое страница устаревает. Если установить его в нуль, то страница будет устаревать в момент загрузки и Internet Explorer не будет ее кэшировать. Кэширование сильно влияет на разработку и может привести к тому, что приложение будет функционировать неправильно. IE кэширует страницы двумя способами: на диске и в памяти. Рассмотрим такой фрагмент кода, показывающий текущее время и дату:

```
<CENTER>Сейчас <%=Now%>
```

Когда IE запрашивают страницу с этим кодом, на сервере выполняется сценарий, и на странице появляется текущее время. Однако, если программа просмотра переходит к другой странице, а затем возвращается к этой, со временем, то время не изменится, поскольку IE не запрашивает ее повторно. В листинге 5 приведена исправленная версия примера, устаревающая уже в момент загрузки.

Листинг 5. *Страница, устаревающая уже в момент загрузки.*

```
<%@ LANGUAGE="VBSCRIPT"%>
<%Response.Expires=-1%>
<HTML><HEAD><TITLE>Forcing a Page to Expire</TITLE></HEAD><BODY>
<H1>Сейчас <%Response.Write Now%>
</BODY> </HTML>
```

Еще один полезный метод объекта Response - Redirect, перенаправляющий программу просмотра на указанный URL:

```
<% Response.Redirect "enter.asp"%>
```

Объект Server

Объект Server (сервер) представляет собой в некотором роде свалку — в том смысле, что предоставляемые им функции никак не связаны между собой, за тем исключением, что все они полезны разработчику для Интернета. Пожалуй, самая важная из всех функций объекта Server — это метод CreateObject, который создает экземпляр компонента ActiveX. Причем это может быть как встроенный компонент, входящий в комплект поставки, так и тот, который написали Вы сами на любом языке. В любом

случае использование компонента ActiveX на сервере требует вызова метода CreateObject.

Контрольные вопросы

1. Что Вы знаете о технологии ASP?
2. Что Вы знаете об объектах и компонентах?
3. Что Вы знаете о файле GLOBAL.ASA?
4. Что Вы знаете об объекте Application?
5. Что Вы знаете о объекте Session?
6. Что Вы знаете об объекте Response?
7. Что Вы знаете об объекте Server?
8. Что Вы знаете о динамических страницах?
9. Что Вы знаете об IIS ?
10. Что Вы знаете о средах программирования?

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, 2015. - 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, 2017. - 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, 2016. - 496 с. [Электронный ресурс]. http://znanium.com/bookread.php?book=139428.