ПАВЕЛ АГУРОВ

bhv®

ASP.NET
СБОРНИК РЕЦЕПТОВ

АРХИТЕКТУРА
ВЕБ-ПРИЛОЖЕНИЙ

ВЕБ-ФОРМЫ
И КОМПОНЕНТЫ ASP.NET

РАБОТА С БАЗАМИ ДАННЫХ

ОТЛАДКА
И ОБРАБОТКА ОШИБОК

ЗАЩИТА ПРИЛОЖЕНИЙ
ASP.NET

СОЗДАНИЕ ОТЧЕТОВ
В MS Excel

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Павел Агуров

ASP.NET
СБОРНИК РЕЦЕПТОВ

Санкт-Петербург
«БХВ-Петербург»
2010

УДК 681.3.06
ББК 32.973.26-018.2
А27

Агуров П. В.

А27 ASP.NET. Сборник рецептов. — СПб.: БХВ-Петербург, 2010. — 528 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-0521-5

В книге собраны практические советы и примеры, которые помогут при создании веб-приложений с использованием ASP.NET: разработка архитектуры веб-приложения, его отладка, профилирование, защита, конфигурирование, работа с данными и многое другое. Рассмотрены специальные инструменты и утилиты, которые позволяют ускорить и упростить разработку и отладку веб-приложений. Уделено внимание обработке исключений в веб-приложениях. Отдельная глава посвящена созданию отчетов в MS Excel. Книга будет полезна не только программистам, которые уже используют в своих разработках ASP.NET, но и тем, кто переходит на технологию ASP.NET с классической ASP или языка PHP.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.03.10.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 42,57.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

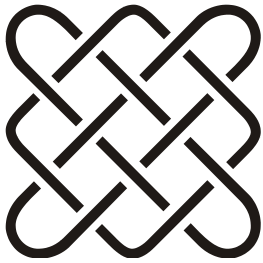
Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0521-5

Оглавление

ГЛАВА 1



Архитектура и общие вопросы

"В будущем, компьютеры будут весить не больше 1,5 тонн".
Журнал "Popular Mechanics", 1949.

1.1. Основные отличия ASP.NET 1.1 и 2.0

- Для разработки проектов не требуется IIS и виртуальная директория.
- Для веб-проектов не используются файлы .csproj, которые раньше создавались для каждого из типов проектов ASP.NET. Вместо файлов проекта Visual Studio использует структуру директорий, таким образом, для того чтобы включить в проект существующий файл, достаточно просто скопировать его в директорию проекта. Если файл или директория удаляется из дерева файлов проекта, то они физически удаляются из файловой системы.
- Директива `@Page` имеет новые атрибуты:
  - `Async` — если значение этого атрибута равно `true`, то страница будет реализовывать интерфейс `IHttpAsyncHandler`, а иначе — интерфейс `IHttpHandler`. В первом случае код страницы может выполняться асинхронно;
  - `AsyncTimeOut` — задает ограничение по времени, отведенное для выполнения асинхронных операций. По умолчанию этот параметр равен 45 секундам;
  - `Culture` — устанавливает набор региональных параметров, используемый для страницы *(см. разд. 1.24)*;
  - `UICulture` — устанавливает набор региональных параметров, используемый для пользовательского интерфейса страницы *(см. разд. 1.24)*;

- `EnableTheming` — позволяет включить или выключить поддержку тем оформления. По умолчанию включено;
- `StyleSheetTheme` — позволяет установить идентификатор темы оформления, которая будет использоваться для изменения установленной темы оформления (в атрибуте `Theme` или в файле web.config). Таким образом можно установить общую тему оформления для всего сайта, а с помощью атрибута `StyleSheetTheme` вносить некоторые изменения в общее оформление страницы и/или некоторых элементов управления, содержащихся на странице;
- `Theme` — указывает название темы оформления, которая будет использована для оформления кода данной страницы;
- `MasterPageFile` — указывает путь к шаблону, который будет использован для создания кода этой страницы. Шаблон называется "мастер-страницей" *(см. разд. 2.18)*.

❒ Класс `Page` имеет новые свойства и методы. Вот некоторые из них:

- `ClientScript` — объект типа `ClientScriptManager` для работы с клиентскими скриптами;
- `RegisterRequiresControlState` — регистрирует элемент, требующий сохранения состояния;
- `EnableTheming` — если значение равно `false`, то поддержка тем на странице отключена;
- `GetValidators` — метод, возвращающий коллекцию валидаторов данной страницы *(см. главу 4)*;
- `Header` — ссылка на объект `HtmlHead`, позволяющий контролировать содержимое раздела `<head>` страницы, при условии, что он отмечен как серверный (`runat="server"`);
- `IsAsync` — если `true`, то страница будет работать в асинхронном режиме;
- `IsCrossPagePostBack` — свойство, позволяющее определить, была ли данная страница запрошена в ответ на отправку данных с другой страницы;
- `Master` — ссылка на экземпляр мастер-страницы *(см. разд. 2.18)*;
- `MasterPageFile` — свойство, содержащее имя файла мастер-страницы *(см. разд. 2.18)*;
- `MaxPageStateFieldLength` — *см. разд. 3.17*;
- `PreviousPage` — ссылка на экземпляр объекта страницы, с которой была осуществлена отправка формы;

- `SetFocus` — *см. разд. 2.7*;
- `Title` — свойство, позволяющее получить и изменить заголовок страницы.

- Сменился цикл жизни страницы. В ASP.NET 1.1 последовательность событий была следующей: инициализация (`Init`), загрузка (`Load`), создание (`PreRender`) и завершение (`Unload`). После этого страница возвращалась клиенту. В ASP.NET 2.0 добавилось еще несколько событий:
  - `PreInit` — вызывается до начала инициализации страницы;
  - `InitComplete` — вызывается после завершения инициализации страницы;
  - `LoadComplete` — вызывается после завершения загрузки страницы;
  - `PreRenderComplete` — вызывается после завершения создания всех элементов страницы.
- В ASP.NET 2.0 применена новая модель компиляции страниц "на лету". Код страниц представляет собой частичный класс (partial class) без описания серверных элементов управления. Эти описания вынесены в другую часть класса страницы, которая генерируется автоматически во время выполнения. Это позволяет разделить автоматически генерируемый код и код пользователя.

1.2. В ASP было...

Несколько ответов на вопрос "вот в ASP было так, а где это в ASP.NET":

- файл global.asa теперь называется global.asax и имеет значительно больше функциональности *(см. разд. 1.20)*;
- эквивалент функции `date()` выглядит как:

```
System.DateTime.Now.ToShortDateString()
```

- эквивалент функции `time()` выглядит как:

```
System.DateTime.Now.ToShortTimeString()
```

- созданные объекты не нужно уничтожать самостоятельно, это делает сборщик мусора (однако файлы и другие ресурсы нужно освобождать сразу после завершения их использования).

Приведу несколько статей, описывающих миграцию с ASP на ASP.NET:

- **http://www.asp.net/downloads/archived/migration-assistants/asp-to-aspnet/**
- **http://www.asp101.com/articles/john/asptoaspnet/default.asp**
- **http://www.asp101.com/articles/paolo/asp2aspnet/default.asp**

1.3. Можно ли запустить приложение ASP.NET под Apache

Да, можно. См. **http://httpd.apache.org/cli/mod_aspdotnet**.

1.4. Где найти исходный код Framework

См. ссылку **http://referencesource.microsoft.com/netframework.aspx**.

1.5. Использование SSI *include* в ASP.NET

Common\SSI[1]

Оператор `include` из языка SSI (Server Side Include, включение на стороне сервера) можно использовать, например, для создания блока скриптов, которые повторяются на всех страницах или регистрации элементов управления. В общем, для создания блоков текстовой информации, которая повторяется на многих страницах.

Например, каждая страница сайта должна содержать набор скриптов, описанный в файле `scripts.inc`:

```
<script type="text/javascript" src="Scripts/common.js"></script>
... другие файлы скриптов ...
```

Для включения содержимого этого файла в страницу используется конструкция `include`, заключенная в знаки HTML-комментариев:

```
<head runat="server">
 <title></title>
 <!-- #include virtual="~/Include/scripts.inc" -->
</head>
```

Теперь содержимое файла scripts.inc будет подставлено в указанное место страницы, и обновлять необходимый список скриптов будет легко и просто. Аналогично можно вставлять любые блоки текста. Только учтите, что реализовывать с помощью SSI единый дизайн страниц (как это делалось в классическом ASP) необходимости нет. Для этого есть более мощный механизм мастер-страниц *(см. разд. 2.18)*. И, кроме того, учтите, что вставка происходит *после* компиляции страницы, поэтому использовать при этом серверный код не получится.

---

[1] Таким образом указывается название папки на прилагаемом к книге компакт-диске, содержащем полный и компилируемый исходный код.

И еще одно замечание. Включение файла можно сделать, например, так:

```
<% Response.WriteFile("~/Include/scripts.inc"); %>
```

Данная запись более близка к ASP.NET, но, как мне кажется, менее красива и проста.

1.6. Как узнать версию ASP.NET, под которой работает сайт

Вызов `System.Environment.Version.ToString()` возвращает версию ASP.NET.

1.7. Как узнать браузер и версию клиента, запустившего сайт

Свойство `Request.Browser.Browser` возвращает название браузера клиента, а `Request.Browser.Version` — полную строку версии браузера (свойства `MajorVersion` и `MinorVersion` возвращают соответственно старшую и младшую части строки версии браузера). Название браузера может быть, например, IE, Netscape, Opera и т. д.

1.8. Как узнать параметры компьютера, на котором работает сайт

С помощью класса `System.Environment` можно узнать информацию о компьютере, на котором работает сайт:

- `System.Environment.MachineName` — имя машины;
- `System.Environment.OSVersion` — версия операционной системы;
- `System.Environment.WorkingSet` — объем памяти;
- `HttpContext.Current.Server.MachineName` — имя машины;
- `HttpContext.Current.Request.ServerVariables["LOCAL_ADDR"]` — локальный IP-адрес машины.

1.9. Где расположен временный каталог

Определить расположение временного каталога можно с помощью вызова

```
Environment.GetEnvironmentVariable("TEMP"))
```

1.10. Как изменить временный каталог ASP.NET

Временный каталог ASP.NET можно изменить с помощью настройки в глобальном файле web.config, который расположен в папке %windows% \Microsoft.NET\Framework\v2.0.50727\CONFIG.

Эта настройка задается параметром `tempDirectory`, например:

```
<compilation tempDirectory="C:\MyTemp">
```

1.11. Информация о соединении

Переменная `Request.IsLocal` возвращает `true`, если запрос локальный. Эта информация полезна, если нужно отобразить некоторые данные только для локальных пользователей (например, отладочную информацию, *см. разд. 14.8*).

Переменная `Request.IsSecureConnection` возвращает `true`, если запрос сделан через защищенное соединение. Проверка этой переменной — более правильный путь, чем сравнение начала URL-адреса со строкой `https`.

1.12. Зачем создается пользователь *ASPNET*

После установки ASP.NET вы можете увидеть созданную учетную запись пользователя `ASPNET`. Он используется в IIS5 процессом aspnet_wp.exe. Если вы не используете IIS5 (или IIS6 в режиме совместимости с IIS5), то можете просто удалить или запретить этот аккаунт. Процессом w3wp.exe это имя не используется.

1.13. Где сохранить данные при переходе между страницами

Для сохранения данных есть следующие варианты:

- адресная строка (URL);
- куки (cookies);
- скрытые поля (hidden fields);
- состояние страницы (ViewState);
- сессия (Session);
- приложение (Application).

Я имею в виду текущие данные пользователя, а не данные вообще, поэтому просто хранение данных в базе данных я здесь не рассматриваю (вопрос хранения данных в БД будет разбираться в *главе 12*).

Каждый из перечисленных вариантов имеет свои сильные и слабые стороны и при выборе нужно четко понимать их различия и в каких случаях какой из вариантов лучше.

1.13.1. Адресная строка

Некоторые данные можно передавать между страницами в *адресной строке*. Точнее сказать, эти данные будут являться параметрами вызывающей страницы, но это вполне можно считать передачей данных. При этом следует учитывать, что, во-первых, эти данные открыты для пользователя (некоторые соображения по безопасности я расскажу в *разд. 14.14*), а во-вторых, длина адреса не может превышать 1024 символа.

Считать данные из параметров страницы можно с помощью свойств `QueryString` и `PathInfo` *(см. разд. 8.13.1)*.

1.13.2. Куки

Куки хранятся на клиентской стороне. В этом их преимущество — если работа с данными идет на клиентской стороне, то не нужно передавать данные на сервер. В этом и недостаток — клиент может легко найти сохраненные данные и подменить их. Не стоит хранить в куках пароли и другую конфиденциальную информацию.

Для работы с куками предусмотрены методы `Add` (добавить), `Get` (получить), `Clear` (очистить), `Remove` (удалить), `Set` (задать) класса `Request.Cookies`.

Например:

```
HttpCookie langCook = new HttpCookie("Language");
langCook.Value = newLang;
Response.Cookies.Add(langCook);
  ..............
if (Request.Cookies["Language"] != null)
{
if (Request.Cookies["Language"].Value != null)
 lang = string.Format(Request.Cookies["Language"].Value);
}
```

Такой пример я привел только ради простоты. В реальном коде обращение к кукам по имени лучше оборачивать в свойства, иначе одна опечатка в строке имени будет стоить не мало времени отладки (см. совет в *разд. 1.15.6*).

1.13.3. Скрытые поля

Скрытые поля использовались в классическом ASP для хранения данных пользователя. Точно так же можно использовать их и в ASP.NET. Нужно понимать, что скрытые поля хранятся в самой странице, а значит, во-первых, легко доступны пользователю, а во-вторых, большой объем данных утяжеляет страницу и увеличивает объем данных, передаваемых между клиентом и сервером. Не стоит класть внутрь страницы большие объемы данных. Не стоит хранить в скрытых полях пароли и секретную информацию.

Для доступа к скрытому полю из CS-кода нужно установить скрытому полю атрибут `runat="server"` и обращаться к такому полю по имени.

1.13.4. Состояние страницы

Состояние страницы (`ViewState`) по сути является тем же скрытым полем, поэтому к нему относится все то, что я перечислил ранее. Преимуществ состояния страницы несколько.

- Все серверные компоненты умеют работать с ним автоматически.
- Доступ к состоянию страницы очень простой — через индексатор `ViewState[имя]`.
- Состояние страницы сжимается, что значительно уменьшает объем передаваемых данных.
- ASP.NET 2.0 может производить контроль изменений состояния страницы. Если данные изменились будет сгенерировано исключение защиты.

К минусам относится увеличивающийся размер страницы (см. также *разд. 3.9.6* и *главу 13*).

1.13.5. Сессия

Сессия (`Session`) позволяет сохранять данные пользователя на то время, пока пользователь работает с сервером. Доступ к данным производится с помощью индексатора `Session[имя]`.

Приведу несколько полезных фактов "из жизни сессии".

- Данные в сессии живут ровно столько, сколько живет сама сессия пользователя. По умолчанию это 20 минут. Если в течение 20 минут пользователь не общается с сервером, сервер закрывает сессию и все данные удаляются. Очень сильно увеличивать это время не хорошо из соображений безопасности *(см. разд. 14.13)*. Как одно из решений — на страницах требующих длительных операций можно добавить таймер, периодически "дергающий" сервер *(см. разд. 13.14)*.

- ASP.NET позволяет гибко настраивать хранилище данных сессии. Сессию можно сохранить в куках, в памяти сервера и в БД *(см. разд. 13.12)*.
- Данные, хранящиеся в сессии, доступны только "владельцу" этой сессии, если, конечно, данные не хранятся в БД, а приложение забирает их в обход всех правил.
- К плюсам хранения данных в сессии можно отнести возможность хранения существенных объемов информации (если, не хранить сессию в пользовательских куках). К минусам — часто довольно затруднительно определить момент, когда данные из сессии можно удалить.
- При хранении сессии в памяти сервера возможны проблемы с кластером серверов. Ведь у каждого сервера в кластере будет своя сессия пользователя. Решений этой проблемы два. Либо не хранить сессию в памяти сервера (или вообще не использовать сессию), либо настроить кластер таким образом, чтобы клиент, подключившись к конкретному серверу кластера один раз, всю сессию работал только с этим сервером.
- В ASP.NET 4.0 добавлен новый параметр `compressionEnabled`, который позволяет сжимать данные в сессии с помощью ZIP-алгоритма *(см. разд. 13.3)*.

1.13.6. Память приложения

Приложение (Application) позволяет сохранять данные, общие для всех сессий, например, счетчик посетителей сайта. Доступ к данным приложения осуществляется с помощью индексатора `Application[имя]`.

Кроме того, любая переменная, описанная как `static`, будет храниться в области данных приложения, а значит, будет общей для всех пользователей сайта. Учитывайте это при разработке приложений *(см. разд. 1.15.13)*!

Аналогично сессиям, в использовании памяти приложения есть проблема, связанная с работой на кластере серверов. Данные приложения хранятся в памяти IIS, поэтому для каждого сервера кластера это будет *своя* область памяти. Нет ничего страшного, если это используется для кэширования — все сведется к тому, что на каждом сервере кластера будет загружен свой кэш. Но если данные приложения хранят общую для всех пользователей (точнее — для всех сессий) информацию, например, счетчик пользователей, то это окажется проблемой — каждый сервер кластера будет иметь свой собственный счетчик. В таких случаях придется обойтись без использования памяти приложения и хранить такие счетчики (и вообще общие данные) где-то в БД.

1.13.7. Что же выбрать

Выбор конкретного хранилища зависит от нескольких факторов:

1. Должны ли данные быть доступны всем пользователям сервера?
2. Каков объем данных?
3. Есть ли в данных конфиденциальная информация?
4. Должны ли данные храниться дольше, чем время жизни страницы?
5. Будет ли приложение работать на кластере?

Если данные должны быть доступны всем пользователям сервера, то единственный вариант, предоставляемый ASP.NET, — это память приложения (`Application`). Нужно только учитывать особенность работы приложения на кластере. Если приложение планируется масштабировать, то лучше сразу хранить данные в БД и не использовать другие варианты.

Если объем данных небольшой, эти данные используются только одной страницей и не являются секретом для пользователя (например, номер выбранной страницы в таблице данных), то лучше воспользоваться адресной строкой, данными страницы или куками.

Куки удобно использовать для неконфиденциальных данных, работа с которыми ведется только на стороне клиента.

Если предполагается хранить существенный объем данных или конфиденциальные данные, то лучше хранить их в сессии (конечно, если сессия не хранится в куках).

1.14. Не используйте подчеркивание в имени серверов

Согласно стандарту RFC 952 (**http://www.ietf.org/rfc/rfc952.txt**) подчеркивание не является разрешенным символом в имени серверов. При его использовании в ASP.NET будут проблемы с сессией и куками — сессия будет работать не предсказуемо.

1.15. Общие правила создания страниц

1.15.1. Не путайте разметку и код

Архитектура веб-страницы в ASP.NET предполагает разделение разметки и кода, но очень часто это правило нарушается. Например, приветствие выводится так: