Самоучитель

Владимир Дронов

Silverlight 3

Основы Microsoft Silverlight 3

Среда разработки
Microsoft Visual Web Developer 2008 Express Edition

Создание пользовательского интерфейса
на языке XAML

Создание логики приложения на языке C#

Компоненты Silverlight 3

Работа с данными

Графика, анимация и мультимедиа

Web-службы и работа с ними

Распространение готовых приложений

bhv

Владимир Дронов

Самоучитель Silverlight 3

Санкт-Петербург
«БХВ-Петербург»
2010

УДК 681.3.06
ББК 32.973.26-018.2
Д75

Дронов В. А.

Д75 Самоучитель Silverlight 3. — СПб.: БХВ-Петербург, 2010. — 464 с.: ил.

ISBN 978-5-9775-0514-7

Доступно описано создание клиентских Web-приложений на платформе Microsoft Silverlight 3. На практических примерах показано, как самостоятельно создавать приложения с богатой функциональностью и развитым интерфейсом, используя при этом исключительно бесплатные инструменты. Кратко даны основы Web-программирования, подробно рассмотрены принципы Silverlight-программирования. Рассказано о среде разработки Microsoft Visual Web Developer 2008 Express Edition, языках программирования XAML и C#, с помощью которых создаются, соответственно, интерфейс и логика Silverlight-приложения. Перечислены основные компоненты Silverlight и объяснено их использование. Дан краткий курс работы с данными, локальными и удаленными файлами и Web-службами, базами данных. Описаны графические, анимационные и мультимедийные возможности Silverlight. Приведены рекомендации по распространению готовых Silverlight-приложений.

Для Web-программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
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 30.12.09.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 37,41.
Тираж 2000 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 978-5-9775-0514-7

Оглавление

ГЛАВА 1

Что такое Silverlight

В самом деле, а что же такое Silverlight? Да, во *введении* уже говорилось, что это платформа для создания клиентских Web-приложений... Но что же тогда клиентское Web-приложение? И что такое платформа?

В двух строчках на эти вопросы не ответишь... Поэтому приготовьтесь к небольшому теоретическому курсу.

Этапы развития WWW

Начнем мы издалека — с рассмотрения процессов, происходящих с современным Интернетом. Куда шагает Интернет? Что ждет нас впереди? И как, наконец, "вскочить" на этот могучий "паровоз", не остаться прозябать на каком-нибудь богом забытом "полустанке"?

Опустим рассказ о самом Интернете (или, как его часто называют, Сети) и о самом популярном его воплощении — WWW. Все это и так знают. И уж, тем более, знают это читатели данной книги (уж раз они хотят заниматься интернет-технологиями...).

Этап первый: обычные Web-страницы

World Wide Web — Всемирная паутина — изначально создавалась для распространения по сети обычных текстовых документов. Эти документы называются *Web-страницами*, пишутся в обычных текстовых редакторах с использованием особого языка *HTML* (Hypertext Markup Language — язык гипертекстовой разметки) и отображаются в особых программах, называемых *Web-обозревателями*. Совокупность Web-страниц, имеющих общее назначение и связанных друг с другом *гиперссылками* (особыми указателями, щелк-

нув на которые можно перейти на другую страницу), называется *Web-сайтом* (или просто *сайтом*). Это все знают.

Язык HTML определяет набор особых команд, называемых *тегами HTML*. Теги задают форматирование и назначение различных фрагментов текста; например, существуют теги для создания обычного абзаца текста, заголовка, для выделения фрагментов текста полужирным и курсивным шрифтом и превращения их в гиперссылки. Таких тегов довольно много, и они позволяют форматировать фрагменты Web-страницы достаточно сложным образом.

Сама же Web-страница представляет собой обычный текстовый файл, который может быть создан в любом простейшем текстовом редакторе, например Блокноте, стандартно поставляемом в составе Microsoft Windows. Этот файл содержит *исходный код* Web-страницы — своего рода предписание Web-обозревателю на языке HTML, *что* и *как* ему следует вывести. Вот только, в отличие от хорошо нам знакомых текстовых файлов, файл Web-страницы имеет расширение не txt, а htm или html; это нужно для корректной работы Web-серверов — особых служебных программ, о которых мы поговорим чуть позже.

Когда Web-обозреватель открывает Web-страницу, он сначала считывает ее исходный код в память, после чего просматривает его содержимое на предмет различных тегов HTML и выясняет, к каким фрагментам текста они относятся. После этого он выводит присутствующий в исходном коде текст на экран, предварительно применив к найденным ранее фрагментам соответствующие им теги. И в результате мы видим на экране абзацы, заголовки, полужирный и курсивный текст, таблицы, гиперссылки и прочее, чем богат HTML.

Но где же Web-обозреватель находит эти чудные Web-страницы? О-о-о, это весьма интересный процесс, который следует рассмотреть подробнее.

Прежде всего, если нам нужно увидеть на экране какую-либо Web-страницу, мы должны набрать в особом поле ввода окна Web-обозревателя ее *интернет-адрес*. Интернет-адрес однозначно идентифицирует компьютер в Сети, где хранится нужная нам Web-страница, и файл самой этой страницы и выглядит примерно так:

http://www.compression.ru/all_anns.htm

Знакомо, правда? Это интернет-адрес Web-страницы со списком новых поступлений сайта "Все о сжатии", посвященного принципам и программам сжатия данных. Строка **http://www.compression.ru** этого интернет-адреса указывает на компьютер, хранящий эту страницу, а строка **all_anns.htm** — на сам файл страницы (собственно, это имя файла, где она хранится). Разделяет их символ слэша (/), третий по счету, если считать с начала интернет-адреса.

Интернет-адрес может иметь и такой, несколько сокращенный, вид:

www.compression.ru/all_anns.htm

Так его, кстати, часто и пишут.

Получив от нас такой интернет-адрес, Web-обозреватель отправляет соответствующему ему компьютеру (в нашем случае — компьютеру с интернет-адресом **http://www.compression.ru**) по Сети особый запрос. Этот запрос содержит имя файла, в котором хранится нужная Web-страница (**all_anns.htm**).

Вот еще один пример интернет-адреса:

http://www.compression.ru/video/index_ru.htm

Он указывает на Web-страницу списка статей в разделе сайта "Все о сжатии", посвященном сжатию видео. Запрос Web-обозревателя будет в этом случае содержать путь к файлу данной Web-страницы — **video/index_ru.htm**.

Однако чаще всего набираемый нами интернет-адрес содержит только строку, идентифицирующую компьютер, без указания имени файла Web-страницы. Например:

http://www.compression.ru/

Понятно, что в этом случае запрос Web-обозревателя не будет содержать имени файла Web-страницы.

Хорошо, компьютер, хранящий нужную нам страницу, получил этот запрос. Что дальше?

Дело в том, что на этом компьютере работает особая программа — *Web-сервер*. Эта программа никак не взаимодействует с пользователем, а занимается только тем, что принимает запросы от Web-обозревателей, запущенных на других компьютерах, и пересылает им запрошенные файлы.

Здесь нужно сказать, что все файлы Web-страниц, составляющих Web-сайт, должны находиться в папке, путь которой указывается в настройках Web-сервера. Это так называемая *корневая папка* сайта. Именно в ней Web-сервер будет искать файлы, которые запрашивают у него Web-обозреватели.

Если запрос Web-обозревателя содержит имя файла, Web-сервер будет искать его в корневой папке. Так, в случае первого из рассмотренных ранее интернет-адресов файл all_anns.htm должен находиться именно там, иначе Web-сервер его не найдет.

Если запрос содержит путь файла, Web-сервер будет искать этот файл в папках, вложенных в корневую папку. Например, получив путь **video/index_ru.htm** (см. второй пример), Web-сервер будет искать файл index_ru.htm, вложенный в папку video, которая, в свою очередь, вложена в корневую папку.

Но что если в запросе не указаны ни имя, ни путь файла? Тогда Web-обозреватель получит *Web-страницу по умолчанию*. Она хранится в файле с именем default.htm[l] или index.htm[l] (может быть изменено в настройках Web-сервера) в корневой папке сайта.

Вот так, в общих чертах, и работает традиционный Интернет.

За все время существования языка HTML в нем появились пять значительных нововведений. Давайте вкратце рассмотрим четыре из них, а пятое отложим на потом.

Первое нововведение — это поддержка графики. Графические изображения сохраняются в отдельных файлах, после чего в нужных местах исходного кода Web-страницы ставятся особые теги, содержащие интернет-адреса этих файлов. Web-обозреватель, встретив такой тег, посылает Web-серверу еще один запрос, получает в ответ графический файл и выводит его на Web-страницу.

Второе нововведение — поддержка каскадных таблиц стилей (Cascade Style Sheet, CSS). Они добавляют к языку HTML мощные возможности форматирования текста и других элементов Web-страницы, приближающиеся к возможностям программ текстовых редакторов.

Третье нововведение — поддержка внедренных элементов Web-страниц. *Внедренными элементами* называются все элементы Web-страниц, содержимое которых не помещается в ее исходный код HTML, а хранится в отдельных файлах. Это аудио- и видеоклипы, документы Adobe PDF, Microsoft Word, Excel и PowerPoint, графика и анимация Adobe Flash и пр. В исходный код Web-страницы помещаются определенные теги, содержащие интернет-адреса соответствующих файлов; встретив такой тег, Web-обозреватель запрашивает нужный файл у Web-сервера и выводит его содержимое на Web-страницу.

На заметку

Собственно, обычные графические изображения также являются внедренными элементами, поскольку хранятся в отдельных файлах.

Четвертое нововведение — поддержка Web-сценариев. *Web-сценариями* называются небольшие программы, помещаемые прямо в исходный код HTML Web-страницы и выполняющие какие-либо действия над ней в ответ на манипуляции пользователя (щелчки мышью, перемещение мыши, нажатие кнопок на клавиатуре и пр.). Такие программы пишутся на особом *языке программирования* JavaScript и могут быть сколь угодно сложными.

Благодаря этим нововведениям мы имеем сейчас совершенно умопомрачительные Web-сайты наподобие YouTube (**http://www.youtube.com/**). В основе лежит старый добрый HTML, обильно используются графические изображе-

ния, для форматирования содержимого применяются каскадные таблицы стилей CSS, видеоклипы представляют собой внедренные элементы, а для управления ими и реализации всяческих дополнительных "красивостей" создаются Web-сценарии.

Этап второй: серверные Web-приложения

Среди пользователей Интернета традиционно много представителей различных компьютерных и околокомпьютерных специальностей, в том числе и программистов. И вот смотрят эти программисты на шикарные Web-странички и думают, к чему бы их можно еще приспособить, кроме публикации в Сети текстов и картинок. Думали они, думали и придумали...

Они размышляли так. Вот написали мы, скажем, программу по складскому учету. Ее нужно установить на каждое рабочее место, настроить, чтобы она "нашла" рабочие данные, научить пользователей с ней работать, а при выходе новой версии — обновить, опять же, на всех рабочих местах. Морока!..

А что если установить эту программу на одном-единственном компьютере, где находятся и ее рабочие данные? А саму ее переписать так, чтобы она работала как Web-сервер, но в ответ на запросы Web-обозревателей не искала готовые Web-страницы на жестком диске, а *сама создавала их* на основе рабочих данных? Да тут такого можно наворотить — все коллеги обзавидуются!

Скажем, набирает пользователь в Web-обозревателе интернет-адрес этой программы (она ведь работает еще и как Web-сервер) и получает сгенерированную ей Web-страницу со списком всех категорий позиций, что есть на складе. Потом он выбирает нужную категорию, и программа генерирует и пересылает ему другую Web-страницу — со списком позиций, относящихся к этой категории. Просто замечательно!

Теперь пользователю нужно списать какую-то позицию. Он выбирает ее из списка, на очередной сгенерированной складской программой Web-странице задает данные, необходимые для списания, и подтверждает их. Программа получает эти данные, обрабатывает и выдает Web-страницу со списком позиций, в которой уже отражены все сделанные изменения.

Что ж, цель ясна! Назовем подобные программы *серверными Web-приложениями* (или просто *серверными приложениями*) — и за работу!

На заметку

На самом деле большинство современных серверных Web-приложений не имеют функциональности Web-сервера, а работают совместно с уже имеющимся Web-сервером, который пересылает им данные, принятые от пользователей, и перенаправляет пользователям сгенерированные этими приложениями Web-страницы. Но это уже детали.

Достоинство такого подхода — серверное приложение не нужно устанавливать на каждый компьютер каждого пользователя, который должен с ним работать, — достаточно иметь там Web-обозреватель (который там наверняка уже есть). Недостаток — писать серверные приложения несколько сложнее, чем обычные, "настольные". (Хотя эта проблема решается подбором хорошей среды разработки.)

Осталась мелочь — внести в язык HTML пятое по счету нововведение. Это *Web-формы* — особые элементы Web-страницы, предназначенные для ввода данных. Они служат вместилищем для *элементов управления*: полей ввода, флажков, переключателей, списков, кнопок и пр., а также занимаются формированием данных для пересылки их серверному приложению.

Сначала серверные приложения были уделом корпоративных сетей, а потом вышли на "широкие просторы" Интернета. Почтовые Web-сервисы, интернет-магазины, поисковые системы — вот далеко не полный перечень областей их применения. И, разумеется, широко распространились Web-формы (пример — на рис. 1.1).

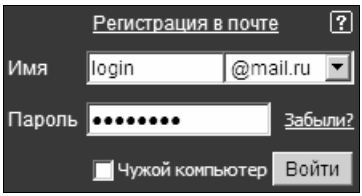

Рис. 1.1. Web-форма на главной Web-странице почтового Web-сервиса Mail.ru

Этап третий: клиентские Web-приложения

Программисты — люди деятельные, и их редко что-либо удовлетворяет полностью. Вот и после создания серверных приложений они начали думать, что бы еще такое сделать, чтобы облегчить жизнь пользователям (и заодно себе).

А что если поместить прямо на Web-страницу для ввода сведений о списании позиции (применительно к складской программе, о которой речь шла ранее) небольшую программу, которая бы проверяла введенные данные на правильность перед тем, как отправить их серверному приложению. Тогда неправильные данные не отправлялись бы по Сети, не отнимали бы время у серверного приложения, не занимали бы системные ресурсы компьютера, на котором она работает, а пользователь сразу бы получил сообщение об ошибке ввода и смог бы ее быстро исправить.

Можно пойти дальше. Когда пользователь требует список позиций, относящихся к определенной категории, серверное приложение генерирует Web-страницу с этим списком и отправляет ему. А ведь эта Web-страница очень

велика! А у пользователя может быть медленный канал, по которому он подключается к сети организации! Результат — Web-страница грузится слишком долго, и пользователь недоволен.

Теперь пользователь выполняет списание позиции. Серверное приложение снова генерирует немалую Web-страницу с обновленным списком позиций, Web-обозреватель неспешно вытягивает ее из сети, и пользователь снова ждет и нервничает.

А мы поместим на Web-страницу другую программу, которая будет принимать от серверного Web-приложения *только данные*, которые значительно компактнее Web-страницы, сохранять их в памяти на будущее и выводить на экран. Если же пользователь выполнит списание, эта программка отправит данные серверному приложению, получит от него сигнал, что списание успешно выполнено, просто-напросто исправит те данные, что получило ранее и хранит теперь в памяти, и выведет их повторно на экран. Никакой долгой загрузки — все произойдет моментально!

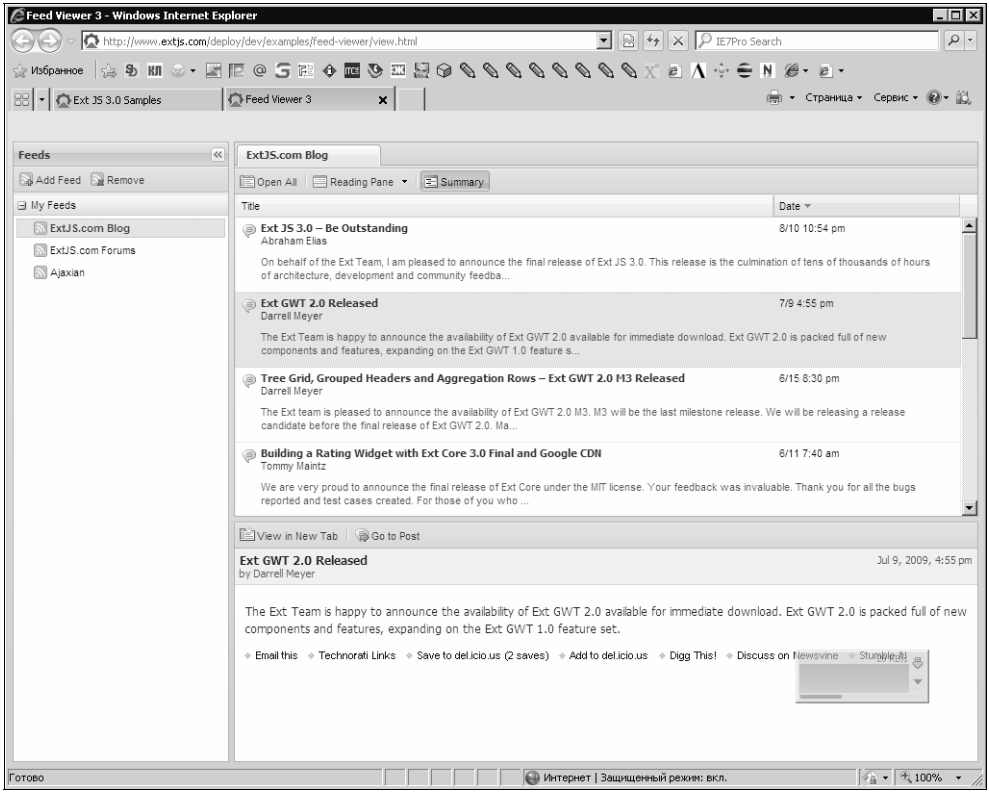

Рис. 1.2. Клиентское Web-приложение — просмотрщик каналов RSS

Да и выглядят эти Web-формы как-то непрезентабельно... Конечно, мы можем применить CSS, чтобы их немного разукрасить, но все равно не то...

А в многомудрую голову (которая рукам покоя не дает) лезут совсем уж крамольные мысли — Web-страница с программкой, которая вообще не будет работать ни с одним серверным приложением. Скажем, игра "15", чтобы сотрудникам было чем заняться, пока начальство не следит за ними...

Такие программы, вставляемые прямо в Web-страницу, программисты назвали *клиентскими Web-приложениями*. И тотчас засели за их написание. Пример такого приложения вы можете увидеть на рис. 1.2.

А чтобы облегчить себе жизнь, попутно создали несколько программных платформ для их создания и приспособили под эти нужды уже существующие. Рассмотрением этих платформ мы сейчас и займемся.

Программные платформы для создания клиентских Web-приложений

Программной платформой, или просто *платформой*, в мире Web-программирования называется совокупность:

- языка программирования;
- набора правил написания на нем программ;
- *библиотек* (дополнительных модулей, расширяющих функциональность данного языка);
- дополнительных программ, необходимых для создания программ на этом языке;
- программы, с помощью которой выполняются написанные на этом языке программы (так называемой *среды исполнения*). Впрочем, среда исполнения присутствует не во всех платформах.

Платформ для создания клиентских Web-приложений довольно много. Сейчас мы рассмотрим три самые популярные и поговорим об их достоинствах и недостатках.

HTML+CSS+JavaScript

Самый очевидный подход при создании клиентских Web-приложений — использовать традиционные интернет-технологии: язык HTML, каскадные таблицы стилей CSS и Web-сценарии, написанные на языке JavaScript. То есть все то, что применяется для создания обычных Web-страниц.