Базовые элементы Web-технологий

Содержание

Web-приложения и технологии их разработки

Программное обеспечение в современном мире становится все сложнее, поскольку на него возлагается решение все более сложных задач. Коммерческие компании и государственные организации стремятся автоматизировать все больше своих процессов, как внутренних, так и тех, что связаны с общением с внешним миром. Необходимые при этом программные системы обычно должны поддерживать одновременную работу многих географически распределенных пользователей с централизованными и интегрированными хранилищами данных организаций. При этом должны обеспечиваться эффективность обработки запросов, высокая надежность работы, предоставление результатов в удобном пользователям виде, разграничение областей доступа и разрешенных операций для разных групп пользователей и т.д. Разработка таких приложений (в тех случаях, когда подходящей системы нет на рынке готовых продуктов), их внедрение и поддержка становятся все дороже из-за возрастающего количества предоставляемых ими функций, ограничений, которым они должны удовлетворять, составляющих их компонентов, возможных связей и взаимодействий между ними.

Есть, тем не менее, факторы, которые помогают значительно снизить расходы на создание подобных систем, один из них — широчайшее распространение Интернет. Если программное обеспечение использует для связи между своими элементами базовые протоколы Интернет (TCP/IP и HTTP) и предоставляет пользовательский интерфейс в виде страничек HTML, которые можно просматривать в любом браузере, то практически каждый его потенциальный пользователь не имеет технических препятствий для обращения к этому ПО. Не нужно прокладывать сети, тратить усилия на настройку связи с серверами, разрабатывать различные клиентские компоненты для разных операционных систем, устанавливать клиентам специальное оборудование и программные компоненты, и т.д. Интернет предоставляет готовую инфраструктуру для создания крупномасштабных

программных систем, в рамках которых десятки тысяч компонентов работают совместно и миллионы людей пользуются предоставляемыми ими услугами.

Таким образом, возник особый класс программных систем — ***Web-приложения***. Это программные системы, использующие для связи протоколы Интернет, а в качестве пользовательского интерфейса — HTML страницы. Сейчас они стали одним из самых востребованных видов ПО. Однако, чтобы сделать потенциальные выгоды от использования Интернет реальными, необходимы технологии разработки Web-приложений, которые позволяли бы строить их на компонентной основе, минимизируя затраты на интеграцию отдельных компонентов, их развертывание и поддержку в рабочем состоянии.

Архитектура Web-приложений

Общая схема архитектуры Web-приложений представлена на Рис. 1. Она включает ряд характерных элементов Web-приложений, но далеко не каждое Web-приложение содержит их все.

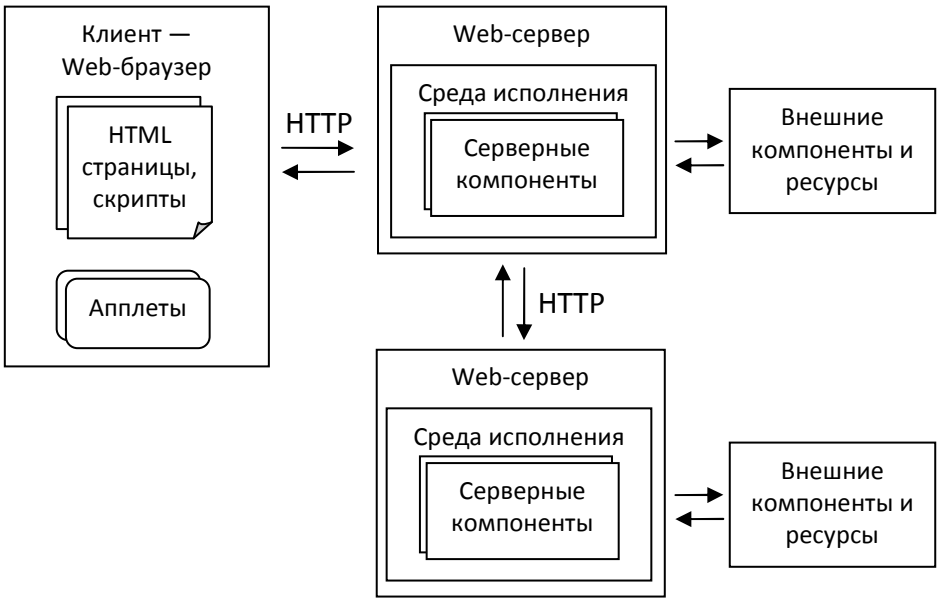

Рисунок 1. Общая архитектура Web-приложений.

Основные элементы Web-приложений следующие.

- Web-приложения состоят из клиентской и серверной частей.
  Клиентская часть предназначена для взаимодействия приложения с пользователем и реализуется в рамках Web-браузера в виде набора HTML-страниц (HTML — Hypertext Markup Language, язык разметки гипертекста).
  Серверная часть предназначена для хранения нужной для работы приложения информации и обработки запросов пользователей. Серверная часть реализуется в виде набора компонентов, работающих в рамках Web-сервера, сервера, обрабатывающего запросы по протоколу HTTP (Hypertext Transfer Protocol, протокол передачи гипертекста), или тесно взаимодействующих с ним. Запрос пользователя, упакованный в HTTP-запрос, поступает Web-серверу на обработку, поскольку он направляется браузером по определенному URI (Uniform Resource Identifier, унифицированному идентификатору ресурса) — адресу, показываемому в адресной строке браузера, для обращения к любым ресурсам в Web.
  Для поиска в глобальной сети машины, на которой размещен данный ресурс, используется протокол DNS (Domain Name Service, служба доменных имен). С помощью DNS адрес домена, т.е. базовая часть адреса ресурса, чаще всего удобная

для восприятия человека (примеры: www.google.com, www.msu.ru и пр.), транслируется в IP-адрес машины, к которой нужно обращаться со всеми запросами на соответствующие адреса.

- Пользовательский интерфейс на стороне клиента реализуется с помощью Web-страниц, оформленных на языке HTML, часто с использованием таблиц стилей и скриптовых языков для динамической модификации представления страниц в браузере в зависимости от действий пользователя или конфигурации браузера (включающей его тип, набор подключенных дополнительных модулей, некоторые характеристики платформы, на которой он работает). Все подобные техники динамической модификации HTML-страниц часто объединяют под названием *динамического HTML* (DHTML) или *клиентских сценариев* (client-side scripting). Иногда, когда возможностей динамического HTML для реализации нужных свойств интерфейса не хватает, используются *апплеты* — приложения, работающие в рамках виртуальной машины, входящей в браузер в виде подключаемого модуля. По сравнению с обычными приложениями апплеты по соображениям безопасности обычно ограничены в доступе к локальным ресурсам (устройствам, файлам, и пр.), находящимся на машине, где работает браузер, но имеют возможность использовать внешние, доступные в Интернет ресурсы.

- На стороне сервера также имеются компоненты, выполняющие функции пользовательского интерфейса. Они принимают и обрабатывают HTTP-запрос, выделяя из него информацию, существенную для данного приложения, а также компонуют или генерируют HTML-страницы (вместе с встроенными в них скриптами) для передачи их клиенту, на основе результатов обработки запроса, пришедшего от пользователя, или конфигурации браузера (насколько ее характеристики присутствуют в HTTP-запросе). Разнообразные техники построения HTML-страниц на стороне сервера называются *серверными сценариями* (server-side scripting)

- Кроме того, на стороне сервера в работе Web-приложения часто участвуют компоненты, задачей которых является собственно обработка запросов — выполнение поиска на сайте, регистрация заказа в магазине, регистрация перевода денег с кредитной карты и пр. В зависимости от используемых технологий такие компоненты могут работать как в рамках процесса Web-сервера, так и в отдельных процессах.

- Очень часто обработка запросов связана с обращением к внешним по отношению к Web-приложению ресурсам — базам данных, службам каталогов и т.д.

- Иногда внешние ресурсы представляют собой *Web-службы*. Для программиста обращение к Web-службам часто ничем не отличается от работы с другими видами ресурсов, а реализация обращений к ним выполнена в виде пересылки сообщений в HTTP-запросах к серверам, на которых находятся компоненты, реализующие такие службы. Современные корпоративные системы могут использовать десятки и сотни различных Web-служб, реализуя на их основе огромное количество функций.

Технологии разработки Web-приложений

Поскольку Web-приложения приобрели огромную популярность, при их разработке используется настолько много разнообразных технологий и языков, что одно их перечисление занимает много места.

- Языки описания содержимого Web-страниц.
    - ***Язык разметки гипертекста*** (Hypertext Markup Language, HTML) служит основой для построения пользовательского интерфейса Web-приложений.

Это базовый язык описания содержимого Web-страниц. Основные элементы HTML — теги и их атрибуты. Тег может соответствовать структурному элементу документа (заголовку, абзацу, таблице), элементу оформления или ссылке на другой документ, картинку, анимацию и пр. Атрибуты тега описывают свойства соответствующих элементов.

- ***Расширяемый язык разметки*** (Extensible Markup Language, XML) так же, как HTML, использует теги и атрибуты, но набор и значение этих тегов и атрибутов определяются типом документа, описываемым его схемой. XML по сравнению с HTML более строг в синтаксических правилах — всегда требует закрытия тегов и указания значений атрибутов. Он предоставляет стандартную лексическую форму для представления текстовой информации различной структуры и стандартные же способы описания этой структуры. XML часто используется для описания разнообразных данных, которыми обмениваются компоненты Web-приложений: конфигурационных файлов, отдельных элементов, из которых компонуются Web-страницы. Аналог HTML на базе XML — ***расширяемый язык разметки гипертекста*** (Extensible Hypertext Markup Language, XHTML).
- ***Каскадные таблицы стилей*** (Cascading Style Sheets, CSS) являются языком для описания представления документов, написанных на одном из языков разметки. В таблице стилей определяются правила, по которым элементам такого документа сопоставляется стиль его представления (задающий цвет текста и фона, шрифт, форматирование абзаца и пр.). Таблицы стилей позволяют разделить описание структуры и представления документа, давая тем самым возможность отображать один и тот же документ по-разному в зависимости от конфигурации браузера или пользовательских настроек.

- Технологии создания клиентских сценариев.
  - ***Объектная модель документа*** (Document Object Model, DOM) представляет собой кросс-платформенный и независящий от используемых языков интерфейс для доступа и обработки структурных элементов документов на HTML и XML (включая XHTML). Этот интерфейс описывается в терминах абстрактных типов объектов, соответствующих различным элементам XML-документа, и возможных операций над ними. Кроме того, указывается техника привязки этого абстрактного интерфейса к реализующим его конкретным в различных языках (сейчас — в Java и ECMAScript). DOM является международным стандартом, поддерживаемым группой W3C.
  - ***Язык ECMAScript*** (с многочисленными диалектами, включающими JavaScript, JScript, ActionScript) — язык, чаще всего используемый для обработки элементов XML- и HTML-документов в браузерах с помощью интерфейса DOM. Соответственно, все браузеры имеют встроенный интерпретатор того или иного диалекта этого языка. Очень часто сценарии на диалектах ECMAScript используются для быстрой, без передачи на сервер, проверки целостности и корректности данных, вводимых пользователем в формах на Web-странице, или для быстрой реакции на те или иные действия пользователя.
  - Разнообразные языки для описания нетекстовых данных, включаемых в представление Web-страницы, и сценариев работы с ними — такие, как Adobe Flash (для мультимедиа–данных, чаще всего, для представления анимационных роликов), VRML (для трехмерной графики и анимации) и пр.

- Технологии создания серверных сценариев.
    - ***Общий интерфейс шлюза*** (Common Gateway Interface, CGI) является наиболее простой, самой старой и уже редко используемой технологией для динамической генерации содержимого Web-страниц на стороне сервера. Это стандарт на запросы, обработку которых Web-сервер передает внешним приложениям, работающими со стандартным вводом/выводом для генерации Web-страниц, выдаваемых в ответ на запросы.
    - Более гибкими и более надежными с точки зрения обеспечения безопасности являются технологии серверных сценариев, выполняемых в рамках Web-сервера или подключаемых к нему модулей. Такие технологии включают ***сервлеты*** и ***серверные страницы Java*** (Java Server Pages, JSP), ***активные серверные страницы*** (Active Server Pages, ASP и ASP.NET) от Microsoft, технологии создания серверных сценариев на языках PHP, ColdFusion, Perl, Python, Ruby и т.д.
- Технологии создания «богатого Web-интерфейса» (Rich Internet Applications).
    - Термином Ajax (исходно Asynchronous JavaScript and XML) обозначается любая технология, использующая клиентские сценарии и передачу XML-сообщений по протоколу HTTP для более гибкой и быстрой обработки запросов пользователей. При этом запросы и ответы отправляются в асинхронном режиме (пользователь может продолжать работать с Web-страницей пока его запрос обрабатывается на сервере) и содержат меньше данных (за счет упаковки только необходимой информации в XML, без передачи всей HTML страницы туда и обратно). За счет этого достигается большая интерактивность Web-страниц — поддерживается больше разных видов действий пользователя, вызывающих реакцию приложения, и гарантируется меньшее время реакции на эти действия — при более эффективном обмене данными между клиентом и сервером.
    - Более сложные технологии, такие как Adobe Flex, Microsoft Silverlight и JavaFX, используются при создании сложных пользовательских интерфейсов на основе как клиентских, так и серверных компонентов. Для описания элементов пользовательского интерфейса и обработки действий пользователя они используют собственные языки ActionScript, XAML и JavaFX Script. Разрабатываемая в настоящий момент следующая версия HTML, HTML 5, должна будет унифицировать такие языки.
- Протоколы, на базе которых происходит обмен сообщениями в Web-приложениях.
    - ***Протокол передачи гипертекста*** (Hypertext Transfer Protocol, HTTP) является основным протоколом, используемым для передачи сообщений и данных в Web-приложениях. Он предназначен для передачи прежде всего текстовых сообщений между слабо связанными сервером и клиентом. HTTP удобен тем, что большинство сетей заранее настроено на прием сообщений по этому протоколу, поскольку браузеры используются повсеместно; для многих других протоколов нужна специальная настройка, соответствующие порты по умолчанию закрыты во многих сетях. Поэтому использование Web-приложений не требует специальной настройки сетей.
    - Протоколы взаимодействия Web-служб. Базовым протоколом передачи обращений Web-служб друг к другу является ***простой протокол доступа к объектам*** (Simple Object Access Protocol, SOAP). Он работает поверх HTTP (реже используются протоколы удаленных вызовов RPC) и определяет на базе XML формат текстовых сообщений, в которых передается информация

о вызовах операций или их результаты. В дополнение к SOAP разрабатываются многочисленные протоколы с названиями, начинающимися на WS (WS-Security, WS-Addressing, WS-Discovery, WS-Coordination и т.п.), которые предназначены для обеспечения дополнительных характеристик взаимодействия Web-служб — поддержки шифрования, аутентификации участников, интроспекции, транзакций, координации нескольких Web-служб в рамках сценария взаимодействия, и т.д.

В рамках данного курса наибольшее внимание уделяется технологиям разработки Web-приложений на основе языка программирования Java и элементам платформы J2EE. Впоследствии планируется расширить набор рассматриваемых технологий.

В рамках следующих нескольких разделов даются краткие сведения о базовых языках для создания Web-страниц и проколе HTTP.

	API	Протоколы	Языки	Диалекты, смеси	Технологии
Описание содержимого страниц и их представления		HTTP	XML HTML CSS	XHTML	
Описание клиентских сценариев	DOM		ECMAScript	ActionScript JScript JavaScript	Flex Silverlight JavaFX
Серверные сценарии	ISAPI Servlet	CGI	Java C# Perl PHP Ruby Python	JSP JSTL JSF ASP.NET - Web Forms	J2EE .NET

Адреса в Web

Унифицированные идентификаторы ресурсов (Uniform Resource Identifier, URI) служат для идентификации различных ресурсов в Web, глобальной информационной сети. Есть несколько видов URI, часто используемым не только для идентификации ресурсов, но и указания способа доступа к ним является URL (Uniform Resource Locator, единый указатель ресурсов). Такой адрес-идентификатор может быть глобальным или локальным.

Глобальные идентификаторы состоят из следующих частей.

- Идентификатор протокола или используемой схемы именования.
- Имя машины-сервера, на которой размещен ресурс, или куда нужно обращаться, чтобы его получить.
- Путь к самому ресурсу, обычно, документу определенного формата, на данной машине.
- (Не обязательно) Идентификатор части документа, если данный адрес указывает не на весь документ.

Например, адрес http://www.ispras.ru/~kuliamin/sdt-course.html раскладывается так:

- http:// — указание того, что для доступа к данному ресурсу нужно использовать протокол HTTP и соответствующую схему именования.
- www.ispras.ru — имя доменного сервера Института системного программирования РАН. Запросы к ресурсам этого домена отправляются на машину с IP-адресом 83.149.199.48. Этот адрес может измениться, тогда потребуется обновить записи в таблицах DNS-серверов, связывающих имя www.ispras.ru с конкретным IP-адресом. Благодаря этому, несмотря на изменения IP-адреса машины, люди могут продолжать пользоваться мнемоническим именем www.ispras.ru.
- ~kuliamin/sdt-course.html — путь к нужному документу на сервере, в данном случае, страничке курса «Технологии программирования».

Если же используется адрес http://www.ispras.ru/~kuliamin/sdt-course.html#test_results, последняя его часть, #test_results, указывает на раздел документа, в котором находятся сведения о прошедших экзаменах по этому курсу.

Локальные идентификаторы URI строятся относительно какого-то фиксированного документа и устроены почти как пути в UNIX-системах. Например, адрес lectures-sdt/Lecture01.pdf относительно указанного выше документа соответствует глобальному адресу http://www.ispras.ru/~kuliamin/lectures-sdt/Lecture01.pdf. В конце локального адреса также может использоваться идентификаторы частей документов с разделителем ‘#’. Можно использовать в начале локального адреса конструкцию «../» для указания перехода в директорию верхнего уровня, однако это делает его корректность зависящей от расположения файлов во внешних директориях.

Унифицированные идентификаторы используются в XML и HTML-документах как ссылки на другие документы, ресурсы или их составляющие.

В идентификаторах ресурсов можно использовать символы различных языков, но те, что не входят узкое множество букв латинского алфавита, цифр и знаков препинания, кодируются. Сначала такой символ представляется в кодировке UTF-8 (символы из первой половины ASCII кодируются одним байтом, а для прочих используется различное число байт, от 2-х до 4-х), а затем каждый байт преобразуется в последовательность из символа ‘%’ со следующими за ним двумя шестнадцатеричными цифрами.

Расширяемый язык разметки XML

Данный раздел содержит краткий обзор основных конструкций XML, для более глубоко изучения этого языка и связанных с ним технологий стоит обратиться к [2-7].

XML [3-5] является *языком разметки*: различные элементы данных в рамках XML-документов выделяются *тегами* — каждый элемент начинается с открывающего тега `<tag>` и заканчивается закрывающим `</tag>`. Здесь `tag` — идентификатор тега, который обычно является английским словом или набором слов, разделяемых знаками ‘-’, которое(-ые) описывают назначение этого элемента данных. Элементы данных могут быть вложены друг в друга, образуя дерево документа. Кроме того, каждый элемент может иметь набор значений атрибутов, которые представляют собой строки, числа или логические значения. Значения атрибутов для данного элемента помещаются внутри его открывающего тега. Элемент данных, не имеющий вложенных подэлементов, может быть оформлен в виде конструкции `<tag … />`, т.е. не иметь отдельного закрывающего тега.

Ниже приведен пример описания информации о книге в виде XML.

```
<book
    title = "Pattern-Oriented Software Architecture, Volume 1: A System of
Patterns"
    ISBN = "047195869"
    year = 1996
    hardcover = true
```

```
    pages = 476
    language = "English">
  <author>Frank Buschmann</author>
  <author>Regine Meunier</author>
  <author>Hans Rohnert</author>
  <author>Peter Sommerlad</author>
  <author>Michael Stal</author>
  <publisher
      title = "John Wiley & Sons"
      address = "605 Third Avenue, New York, NY 10158-0012, USA" />
</book>
```

В этом примере тег `<book>`, представляющий описание книги, имеет вложенные теги `<author>` и `<publisher>`, представляющие ее авторов (таких тегов может быть несколько) и издателя. Он также имеет атрибуты `title`, `ISBN`, `year`, `hardcover`, `pages` и `language` (название книги, ее международный стандартный номер, т.е. International Standard Book Number или ISBN, плюс год издания, наличие твердой обложки, число страниц и язык). Тег `<publisher>`, в свою очередь, имеет атрибуты `title` и `address` (название и юридический адрес издательской организации).

Элементы XML-документа, называемые также *сущностями*, являются в некотором смысле аналогами значений структурных типов значений, а значения их атрибутов — аналогами соответствующих значений полей. При этом теги играют роль самих типов, а атрибуты и вложенные теги — роль их полей, имеющих, соответственно, примитивные и структурные типы. Расширяемым XML назван потому, что можно задать специальную структуру тегов и их атрибутов для некоторого вида документов. Эта структура описывается в отдельном документе, называемом *схемой*, который сам написан на специальном подмножестве XML, *DTD (Document Type Declaration, декларация типа документа)* [3-5] или *XMLSchema* [6].

XML-документ всегда начинается заголовком, описывающим версию XML, которой соответствует документ, и используемую кодировку. По умолчанию используется кодировка UTF-8.

Затем чаще всего идет описание типа документа, указывающее схему, которой он соответствует, и тег корневого элемента, содержащего все остальные элементы данного документа. Схема может задаваться в формате DTD или XMLSchema. Второй, хотя и является более новым, пока еще используется реже, потому что достаточно много документов определяется с помощью DTD и очень многие инструменты для обработки XML могут пользоваться этим форматом. Используемая схема определяется сразу двумя способами — при помощи строки, которая может служить ключом для поиска схемы на данной машине, и при помощи унифицированного идентификатора документа (Unified Resource Identifier, URI), содержащего ее описание и используемого в том случае, если ее не удалось найти локально.

Ниже приводится пример заголовка и описания типа документа для дескриптора развертывания EJB компонентов.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Application
Server 8.1 EJB 2.1//EN" "http://www.sun.com/software/appserver/dtds/sun-ejb-
jar_2_1-1.dtd">
<sun-ejb-jar>
…
</sun-ejb-jar>
```

Другой пример показывает заголовок документа DocBook — основанного на XML формата для технической документации, которая может быть автоматически

преобразована в HTML, PDF и другие документы с определенными для них правилами верстки.

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.3//EN"
"http://www.oasis-open.org/docbook/xml/4.3/docbookx.dtd">
<article>
…
</article>
```

Помимо элементов данных и заголовка с описанием типа документа, XML-документ может содержать комментарии, помещаемые в теги `<!-- … -->`, *инструкции обработки* вида `<? processor-name … ?>` (здесь `processor-name` — идентификатор обработчика, которому предназначена инструкция) и *секции символьных данных* CDATA, которые начинаются набором символов `<![CDATA[`, а заканчиваются с помощью `]]>`. Внутри секций символьных данных могут быть любые символы, за исключением закрывающей комбинации. В остальных местах некоторые специальные символы должны быть представлены комбинациями символов в соответствии с Таблицей 1.

Символ	Представление в XML
<	<
>	>
&	&
"	"
'	'

Таблица 1. Представления специальных символов в XML.

XML содержит много других конструкций, помимо уже перечисленных, но их рассмотрение выходит за рамки данного курса. Читатель, желающий узнать больше об этом языке и связанных с ним технологиях, может обратиться к [1-7].

Протокол передачи гипертекста HTTP

Поскольку основная функциональность компонентов интерфейса Web-приложений связана с созданием и обработкой сообщений HTTP, знание элементов этого протокола необходимо для понимания технологий разработки приложений такого рода.

HTTP (Hypertext Transfer Protocol, протокол передачи гипертекста) представляет собой протокол прикладного уровня, использующий для пересылки данных протокол транспортного уровня. Достаточно подробное описание его можно найти в [8,9].

Сообщения HTTP бывают двух видов: ***запросы клиента*** и ***ответы сервера***.

Запрос HTTP состоит из идентификации метода запроса, идентификатора запрашиваемого ресурса (URI), указания версии протокола и, возможно, набора заголовков с дополнительной информацией, а также поля данных общего вида.

<Request> ::= GET <URI> CrLf
| <Method> <URI> <HTTP-Version> CrLf <Req-Header>* (<Data>)?
<Req-Header> ::= <Field> : <Value> CrLf
CrLf ::= '\r''\n'

Основные методы протокола HTTP следующие.

- GET
  Служит для получения любой информации по URI запроса, обычно — документа, хранящегося по указанному адресу или генерируемого по запросу с таким URI.

Может иметь заголовок If-Modified-Since, который предписывает не посылать тело запрашиваемого ресурса, если он не изменялся с указанной даты.

- POST
  Служит для создания нового ресурса, связанного с указанным по URI. Чаще всего используется для аннотации ресурсов, добавления сообщений в группы новостей, дистанционной работы с базами данных. Реальная обработка такого запроса зависит от содержащегося в нем URI.

Остальные методы — HEAD, PUT, DELETE, LINK, UNLINK — используются гораздо реже.

Заголовки запроса служат для передачи дополнительной информации об этом запросе или о клиенте. Заголовок состоит из идентификатора поля и его значения, разделенных двоеточием, и бывает одного из следующих типов.

- From
  Содержит e-mail адрес пользователя, под чьим именем работает клиент.
  Пример: From: webmaster@yandex.ru
- Accept, Accept-Encoding, Accept-Charset и Accept-Language
  В таком заголовке через запятую перечисляются возможные форматы (соответственно, кодировки, используемые таблицы символов и языки) ответов на данный запрос.
  Пример: Accept: text/plain, text/html, text/x-dvi; q=.8; mxb=100000; mxt=5.0
- User-Agent
  Содержит название используемой клиентской программы.
- Referer
  Используется для указания адреса ресурса, с которого был получен данный запрос.
- If-Modified-Since
  Используется для отмены ответной пересылки документов, модифицированных не позднее указанной даты, с целью снижения нагрузки на сеть.
- Authorization
  Содержит авторизационную информацию, специфичную для используемых сервером протоколов авторизации.
- ChargeTo
  Содержит информацию о том, куда выставить счет за обработку запроса.
- Pragma
  Содержит дополнительные директивы для промежуточных серверов, например, прокси-серверов.

Пример запроса.

```
GET /locate?keywords=HTTP+description HTTP/1.1
Date: Mon, 15 Dec 2004 12:18:15 GMT
Accept: image/gif, image/jpg, */*
Accept-Charset: iso-8859-1, *, utf-8
Accept-Language: en
Connection: keep-Alive
User-Agent: Mozilla/4.7 [en] (Win98; u)
```

Группа кодов	Код	Фраза-объяснение, следующая за кодом	Значение кода
1xx			Информационное сообщение
2xx			Успешная обработка
	200	OK	Все нормально
	201	Created	Документ создан
3xx			Перенаправление запроса
	301	Moved Permanently	Ресурс перемещен
	302	Moved Temporarily	Ресурс перемещен временно
4xx			Ошибка клиента
	400	Bad Request	Некорректно составленный запрос
	401	Unauthorized	Нужна аутентификация клиента
	403	Forbidden	Доступ к ресурсу запрещен
	404	Not Found	Запрашиваемый ресурс отсутствует
5xx			Ошибка сервера
	500	Internal Server Error	Внутренняя ошибка сервера

Таблица 2. Некоторые коды статуса ответа HTTP и их объяснение.

Ответ сервера на HTTP-запрос состоит либо только из запрашиваемого клиентом документа, либо в дополнение к нему содержит код статуса ответа и, возможно, несколько заголовков ответа.

<Response> ::= (<Content>)?
| <HTTP-Version> <Code> <Explanation> CrLf <Resp-Header>* (<Content>)?
<Resp-Header> ::= <Field> : <Value> CrLf
CrLf ::= '\r''\n'

Некоторые коды статуса ответа поясняются в Таблице 12.

Возможны следующие заголовки ответа.

- Allowed
  Перечисляются через пробел доступные для пользователя методы запросов по данному URI.
- Public
  Перечисляет доступные всем методы запросов.
- Content-Length, Content-Type, Content-Encoding и Content-Language
  Задают размер содержимого в байтах (подразумевается, что содержимое имеет бинарный формат и не должно разбиваться на строки при чтении), его формат или MIME-тип, кодировку и язык.
- Date
  Дата создания содержащегося документа или объекта.
- Last-Modified
  Дата последнего изменения содержащегося объекта.
- Expires
  Дата, послед которой содержащийся объект считается устаревшим.

- URI
  URI содержащегося объекта.
- Title
  Заголовок содержащегося документа.
- Server
  Описывает серверную программу.
- Retry-After
  Определяет промежуток времени на обработку запроса, до прохождения которого не надо направлять запрос повторно, если ответа на него еще нет.

Пример HTTP-ответа.

HTTP/1.0 200 OK
Content-Length: 2109
Content-Type: text/html
Expires: 0
Last-Modified: Thu, 08 Feb 2001 09:23:17 GMT
Server: Apache/1.3.12
<HTML> <HEAD> <TITLE> … </TITLE> </HEAD>
<BODY> …</BODY>
</HTML>

Язык разметки гипертекста HTML

HTML (Hypertext Markup Language, язык разметки гипертекста) — язык разметки, т.е., как и XML, использующий для выделения различных элементов теги, предназначенный для описания гипертекстовых документов, содержащих, помимо обычного форматированного текста, ссылки на картинки, видео, звуковые файлы, а также другие документы или файлы. В отличие от XML правила оформления тегов менее жесткие — некоторые элементы могут не иметь закрывающих тегов, а некоторые — содержания.

Здесь дается поверхностный обзор основных элементов HTML, для более детального знакомства с этим языком см. [10-12].

Структура документов HTML

HTML-документ обычно построен по следующему шаблону. В начале идет декларация типа документа.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Затем — основной элемент <html> с заголовком и содержимым.

```
<html>
  <head>
    <title>Заголовок документа</title>
  </head>
<body>
Содержимое документа
</body>
</html>
```

Тип документа — специфический формат, определяющий набор используемых тегов и правила их интерпретации браузером, указывается в самом первом теге. В приведенном примере — это HTML версии 4.01. Можно встретить такие объявления типа документа.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

или

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

В последнем случае указывается, что документ составлен на XHTML. XHTML отличается от HTML жестким следованием правилам XML при оформлении элементов — это, фактически, HTML, вписанный в синтаксис XML.

В качестве атрибута тега типа документа в первом и последнем случаях указан также URI схемы документа.

За типом документа следует корневой элемент HTML-документов <html/>. Весь остальной текст помещается внутрь этого элемента.

В элемент <head/> помещается имя документа, показываемое в виде заголовка окна браузера (внутри тега <title/>), а также дополнительная информация о документе в целом.

Вслед за элемент <head/> идет элемент <body/>, в который помещается содержимое документа — все то, что отображается в окне браузера при просмотре этой странички.

Основные виды элементов

Все конструкции языка HTML являются элементами с точки зрения обобщенного языка разметки (SGML, Standard Generalized Markup Language, стандартный обобщенный язык разметки). Такой элемент может состоять из открывающего тега, его содержимого и закрывающего тега. В некоторых случаях может отсутствовать закрывающий тег (например, тег <p> открывает абзац, который обрывается там, где начинается следующий структурный элемент), иногда может отсутствовать и открывающий тег (можно не писать теги <head> или <body>, но лучше так не делать), иногда — и содержимое (например, тег
 обозначает просто разрыв строки). Далее при описании элементов их обозначение указывается в виде <tag/>, если такой элемент должен или может иметь закрывающий тег, и в виде <tag>, если закрывающего тега не должно быть.

Элементы могут иметь атрибуты, которые, как и в XML, указываются в виде пар `имя = значение` внутри открывающего тега. Значения атрибутов указываются в двойных или одинарных кавычках, в тех случаях, когда значение описывается строкой из цифр и букв латинского алфавита (или включает также подчеркивания, запятые, дефисы и двоеточия), можно не использовать кавычек.

Прописные и строчные буквы в именах тегов и атрибутов не различаются.

В HTML документах можно использовать все символы Unicode [13], однако указанная в описании типа документа кодировка не всегда способна представить все такие символы. В этих случаях используются числовые ссылки вида “å” (символ @) с десятичными кодами Unicode или “&#E5;” (тот же символ) с шестнадцатеричными кодами. Так же, как и XML, вместо символов <, >, ", ' используются комбинации, указанные в Таблице 1. Кроме этого, определены специальные обозначения для многих символов, перечисление которых занимает слишком много места (см. [10]). Можно отметить следующие.

Символ	Представление в HTML
Неразрывный пробел	
©	&copy;
§	&sect;

Таблица 3. Представления некоторых специальных символов в HTML.

Элементы в HTML документах относятся к следующим типам.

- Комментарии. Выглядят комментарии так:
  `<!-- Это комментарий -->`
- Структурные элементы документа.
  Основные структурные элементы — это заголовок с вложенными элементами, элементы, описывающие фреймы, скрипты, различные элементы текста — заголовки разделов, абзацы и пр., списки, таблицы и формы.
- Элементы оформления.
  Такие элементы позволяют определить способ представления содержания: выделение цветом, полужирным и курсивным шрифтом и пр. В HTML 4.0 большинство таких тегов признаны нежелательными, описание оформления переносят в таблицы стилей, облегчая этим изменения в оформлении документов.
- Ссылки на другие документы, картинки, анимации и пр. внешние объекты.

Элементы заголовка

В заголовке — элементе <head/> — помещается информация о документе в целом.

- Прежде всего, это имя документа, помещаемое в элемент <title/>. Оно представляется в браузерах в качестве заголовка окна.
- Элемент <base>, не имеющий закрывающего тега, определяет с помощью атрибута href базовый адрес текущего документа, в соответствии с которым вычисляются полные адреса относительных ссылок.
- Элемент <style/> служит для определения таблиц стилей, описываемых внутри комментариев в содержимом этого элемента. Его обязательный атрибут type определяет MIME-тип содержимого — обычно он равен «text/css».
- Элемент <link>, не имеющий закрывающего тега, определяет связи текущего документа с другими. Чаще всего он указывает внешние таблицы стилей, остальные виды связей браузеры не обрабатывают, но они могут использоваться роботами поисковых систем.
  Атрибут href задает URI другого документа, связь с которым определяется.
  Атрибут rel задает тип определяемой взаимосвязи. Можно указывать следующие типы связей.
  - stylesheet — таблица стилей для данного документа.
  - home — главная страница сайта, на котором расположен документ.
  - toc или contents — оглавление для данного документа.
  - index — индекс для данного документа.
  - glossary — список терминов, используемых в данном документе.
  - copyright — информация об авторах и авторских правах на документ.
  - up (или parent), child, next, previous, first, last (или end) — связанные документы в иерархической структуре сайта (родительская страница, дочерняя страница, следующая, предыдущая, первая или последняя страница).
  - help — документ со справочной информацией о сайте.
- Элемент <meta>, не имеющий закрывающего тега, указывает значения некоторых мета-атрибутов документа, недоступных пользователям, но используемых роботами поисковых систем.
  Имя определяемого мета-атрибута указывается в атрибуте name, а его значение —

в атрибуте content. Имеется довольно много предопределенных атрибутов: Generator, Publisher-Email, Keywords, Description и пр.

Кроме того, можно указать название HTTP-заголовка в качестве имени мета-атрибута в атрибуте http-equiv (вместо name), чтобы определить содержимое этого заголовка при передаче данного документа.

В качестве примера можно взять заголовок главной страницы сайта www.rbc.ru.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<meta http-equiv="Refresh" content="720; URL=http://www.rbc.ru/?/">
<link rel="alternate" type=application/rss+xml title="Главные новости"
href="http://static.feed.rbc.ru/rbc/internal/rss.rbc.ru/rbc.ru/mainnews.rss">
<link rel="alternate" type=application/rss+xml title="Краткая лента новостей"
href="http://static.feed.rbc.ru/rbc/internal/rss.rbc.ru/rbc.ru/newsline.rss">
<link rel="alternate" type=application/rss+xml title="РБК daily: главные
новости"
href="http://static.feed.rbc.ru/rbc/internal/rss.rbc.ru/rbcdaily.ru/mainnews.
rss">
<link rel="alternate" type=application/rss+xml title="Спорт: лента новостей"
href="http://static.feed.rbc.ru/rbc/internal/rss.rbc.ru/sport.rbc.ru/newsline
.rss">
<title>РосБизнесКонсалтинг - новости, акции, курсы валют, погода, доллар,
евро</title>
<meta name="Description" content="&quot;РосБизнесКонсалтинг&quot; - ведущая
российская компания, работающая в сферах масс-медиа и информационных
технологий.">
<meta name="Keywords" content="новости, доллар, политика, курс, акции,
облигации, нефть, рубль, евро, финансы, экономика, банк, кредит, вексель,
информация, фондовый, инвестиционный, рынок, недвижимость, валюта, эмитент,
комментарии, аналитика, продаж/продажа, цена, компания, товары, исследование,
прогноз, индекс, рейтинг, биржа, семинар, фондовый, ПИФ, доходность, IPO,
паевой">
<link href="http://pics.rbc.ru/img/skin/fp_v3/layout_34.css" rel="stylesheet"
type=text/css>
<link href="http://pics.rbc.ru/img/skin/fp_v3/skin_34.css" rel="stylesheet"
type=text/css>
<link href="http://pics.rbc.ru/img/skin/fp_v3/index_34.css" rel="stylesheet"
type=text/css>
<script language='JavaScript'
src='http://pics.rbc.ru/js/fp_v3/lib.js'></script>
<script language='JavaScript'
src='http://pics.rbc.ru/js/fp_v3/indices.js'></script>
</head>
<body> ... </body>
```

Структурные элементы текста

Основное содержание документа помещается в элементе <body/>. Сам этот элемент имеет несколько атрибутов, определяющих форматирование документа, цвета фона, основного текста, ссылок и пр.

- Для указания заголовков разделов различных уровней (чем меньше уровень, тем крупнее соответствующий раздел) используются элементы <h1/>, <h2/>, …, <h6/>. Они могут иметь атрибут align, определяющий выравнивание заголовка, с возможными значениями left, right и center (выравнивание влево, вправо и центрирование).
- Элемент <p/> с возможным атрибутом align используется для разметки абзацев.
- Элемент
 указывает место обязательного перевода строки.
- Элемент <hr> вставляет в текст разделительную горизонтальную линию. Его атрибуты:
    - width — указывает длину линии в точках или процентах от ширины окна;
    - size — толщина линии;
    - align — выравнивание линии;
    - color — указывает цвет линии;
    - noshade указывается без значения и обозначает, что линию нужно изображать без тени, как сплошную, а не как объемную.
- Элемент <pre/> указывает браузеру изображать его содержимое в том же форматировании, как оно представлено в файле, с использованием моноширинного шрифта и с сохранением пробелов и переводов строк.
- Элемент <code/> служит для выделения формул. Обычно они отображаются с помощью моноширинного шрифта.
- Элементы <cite/> и <blockquote/> служат для указания цитат. Второй, в отличие от первого, используется для длинных цитат, оформляемых как отдельный абзац с дополнительным отступом.
- Элемент служит для выделения блока текста.
- Ненумерованные списки создаются при помощи элемента . Каждый пункт списка представлен элементом .
- Нумерованные списки организуются с помощью элемента . Он может иметь атрибуты start, определяющий нумерацию первого пункта, и type, задающий тип нумерации. Доступные типы: «1» — арабские числа (используется по умолчанию), «A» — заглавные латинские буквы, «a» — строчные латинские буквы, «I» — латинские числа заглавными буквами, «i» — латинские числа строчными буквами. Элемент в рамках упорядоченного списка может иметь атрибут value, устанавливающий для соответствующего пункта списка особый номер в нумерации, последующие пункты нумеруются следующими за ним номерами, если для них не установлены свои собственные.

Форматирование текста

- Элемент <basefont> определяет с помощью атрибута size, принимающего значения от 1 до 7, базовый размер шрифта, которым отображается текст документа.
- Элемент <font/> позволяет задать цвет (с помощью атрибута color), размер (с помощью атрибута size) и тип (с помощью face) шрифта для своего содержимого.
- Элементы , , <big/>, <small/>, <i/>, <b/>, <u/>, <s/> используются для различного выделения текста. задает обычное выделение, — сильное. Чаще всего в первом случае текст изображается курсивом, во втором — полужирным шрифтом. Следующие два элемента служат для увеличения или уменьшения размера используемого шрифта. Элементы <i/>, <b/>, <u/>, <s/>

обозначают использование курсива, полужирного шрифта, подчеркивания и зачеркивания.

- Элементы <sup/> и <sub/> служат для выделения верхних и нижних индексов.

Таблицы

Таблицы в HTML оформляются с помощью элемента <table/>. Ячейки внутри ее описываются с помощью элементов <caption/>, <tr/>, <th/> и <td/>. Каждая ячейка может содержать произвольный HTML-текст, включая вложенные таблицы. Пустые ячейки обычно изображаются без границ, чтобы избежать этого, лучше поместить в ячейку символ неразрывного пробела, представляемый как “nbsp;”.

- Элемент <table/> имеет следующие атрибуты:
    - align и valign указывают горизонтальное и вертикальное выравнивание таблицы;
    - border определяет ширину внешней рамки в точках;
    - cellspacing задает расстояние между границами отдельных ячеек;
    - cellpadding указывает расстояние между границей ячеек и их содержимым;
    - width определяет ширину таблицы в точках или процентах от ширины окна браузера;
    - height точно так же задает ее высоту;
    - bgcolor указывает цвет фона ячеек;
    - background задает URL рисунка, который будет использован как фон таблицы.
- Элемент <caption/> указывает заголовок таблицы, а его атрибут align с возможными значениями top и bottom определяет место размещения заголовка, сверху или снизу от таблицы.
- Элемент <tr/> определяет новую строку ячеек таблицы. С помощью атрибутов align, valign и bgcolor можно задать горизонтальное и вертикальное выравнивание во всех ячейках этой строки, а также цвет их фона.
- Элементы <td/> и <th/> служат для описания отдельных ячеек. <th/> определяет ячейку-заголовок, которую браузер может как-то выделить. Оба элемента имеют следующие атрибуты:
    - align и valign указывают горизонтальное и вертикальное выравнивание содержимого ячейки; по умолчанию обычные ячейки выравниваются по горизонтали влево, а заголовки — по центру, по вертикали оба типа ячеек выравниваются по центру;
    - width определяет ширину ячейки в точках или процентах от ширины таблицы;
    - height задает высоту ячейки;
    - colspan определяет количество столбцов, занимаемое данной ячейкой;
    - rowspan задает количество строк, занимаемых ячейкой;
    - nowrap блокирует форматирование содержимого ячейки в много строк;
    - bgcolor указывает цвет фона ячеек;
    - background задает URL рисунка, который будет использован как фон таблицы.

Пример структурирования и форматирования текста документации на Spring Framework.

```
<body bgcolor="white" text="black" link="#0000FF" vlink="#840084"
alink="#0000FF">

<div class="navheader">
```

<table width="100%" summary="Navigation header">

<tr><th colspan="3" align="center">1. Introduction to Spring Framework</th></tr>

<tr><td width="20%" align="left">Prev </td>

<th width="60%" align="center">Part I. Overview of Spring Framework</th>

<td width="20%" align="right"> Next</td></tr>

</table>

<hr>

<h2 class="title">1. Introduction to Spring Framework</h2>

Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application.

Spring enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs. This capability applies to the Java SE programming model and to full and partial Java EE.

Examples of how you, as an application developer, can use the Spring platform advantage:

<ul type="disc">

 Make a Java method execute in a database transaction without having to deal with transaction APIs.

 Make a local Java method a remote procedure without having to deal with remote APIs.

 Make a local Java method a management operation without having to deal with JMX APIs.

 Make a local Java method a message handler without having to deal with JMS APIs.

<h2 class="title" style="clear: both">1.1 Dependency Injection and Inversion of Control</h2>

Background

“The question is, what aspect of control are [they] inverting?” Martin Fowler posed this question about Inversion of Control (IoC) on his site in 2004. Fowler suggested renaming the principle to make it more self-explanatory and came up with Dependency Injection.

For insight into IoC and DI, refer to Fowler's article at http://martinfowler.com/articles/injection.html.

```
<p>Java applications -- a loose term that runs the gamut from constrained
applets to n-tier server-side enterprise applications -- typically consist of
objects that collaborate to form the application proper. Thus the objects in
an application have <span class="emphasis"><em>dependencies</em></span> on
each other.</p>
```

Вот как выглядит сформированная таким образом страница в браузере.

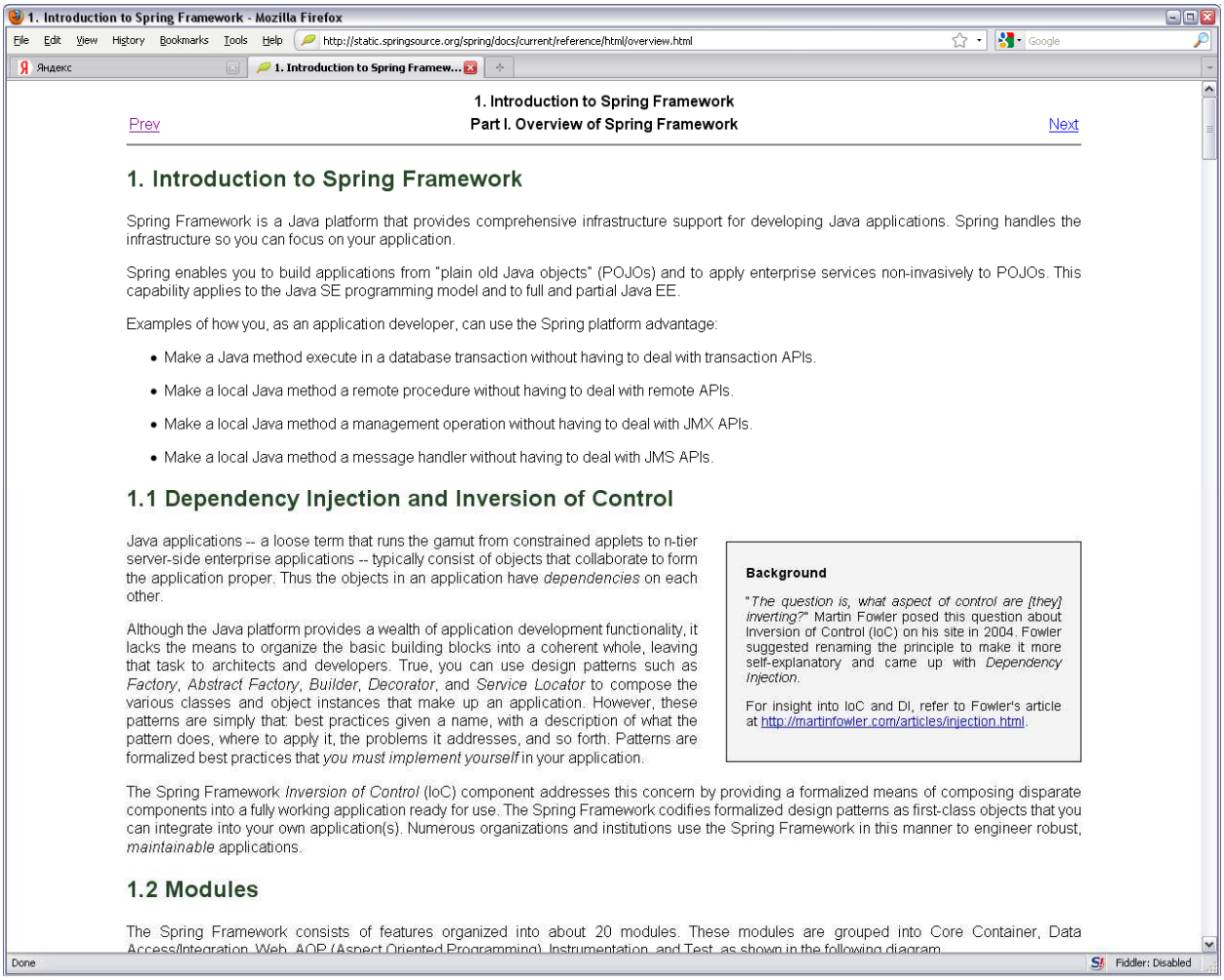

Рисунок 2. Вид приведенный выше HTML-страницы в браузере.

Ссылки

Для определения ссылок на другие документы могут использоваться следующие элементы.

- Для определения ссылок и мест, куда может указывать ссылка, используется элемент <a/>.
    - Для определения места для ссылки (якоря) нужно указать элемент <a/> с атрибутом name. При этом область текста между открывающим и закрывающим тегами будет считаться объектом ссылки.
    - Для определения ссылки нужно указать <a/> с атрибутом href, значение которого задает URL ссылки. Для ссылок внутри документа используются URL вида #name, где name — имя соответствующего якоря.
- С помощью элементов <map/> и <area> можно организовать навигационные карты, например, для определения ссылок, по которым переходи браузер при клике на различные части изображения.

- <map/> имеет атрибут name, задающий имя карты, и может иметь несколько вложенных элементов <area>.
- <area> имеет такие атрибуты:
  - shape — определяет форму активной области, может иметь значения rect, circle, poly;
  - coord — задает координаты углов или центра области;
  - href — определяет URL ссылки;
  - nohref — определяет данную область как неактивную;
  - alt — задает текст для показа вместо изображения для данной области.

Скрипты и объекты

Для описания дополнительных объектов на странице HTML есть несколько элементов.

- Для вставки изображений в HTML-документ используется элемент <img>, имеющий следующие атрибуты.
  - src — определяет URL вставляемого изображения.
  - height и width — задают высоту и ширину получаемой картинки, если параметры реального изображения не совпадают с указанными, оно масштабируется, т.е. растягивается или сжимается.
  - hspace и vspace — задают отступы по горизонтали и вертикали, отделяющие изображение от соседних объектов, например, обтекающего его текста.
  - align — определяет способ выравнивания изображения.
  - name — задет имя объекта-рисунка.
  - alt — задает текст, появляющийся вместо изображения, если браузер не может его найти или работает без подкачки изображений.
  - border — определяет ширину рамки вокруг изображения.
  - usemap — определяет имя используемой навигационной карты.

Наиболее часто для определения используемых скриптов применяют элемент <script/>. Он может иметь атрибут language, указывающий используемый скриптовый язык, и либо содержит текст срипта, либо с помощью атрибута src задает URL для внешнего срипта.

Другие элементы, используемые для вставки исполняемых объектов — <applet/> (используется для вставки Java апплетов), <object/>.

Формы

Формы служат для определения интерактивных элементов HTML-страницы, т.е. таких, с которыми может взаимодействовать пользователь.

Элемент <form/> определяет форму, которая может содержать элементы управления, а также обычные элементы HTML.

Форма может иметь следующие атрибуты

- name — имя формы.
- action — обязательный атрибут, указывающий URL, к которому нужно обратиться браузеру для обработки данных формы.
- method — определяет способ отправки данных формы на сервер, может иметь два значения: GET и POST (методы протокола HTTP). Метод POST, в отличие от GET, используется для передачи значительных объемов информации, например,

содержимого многострочного текстового поля, файлов по указанным пользователем именам и пр.

- enctype — способ кодирования содержимого при отправке, возможные значения: multipart/form-data и application/x-www-form-urlencoded.

Формы могут содержать следующие элементы управления.

- <input/> — определяет элемент управления (поле ввода, кнопку или чекбокс (флажок)), содержание которого может быть изменено пользователем. Его атрибуты таковы.
    - name — задает имя элемента управления, используется для его идентификации при передаче данных на сервер.
    - value — определяет текстовый заголовок или заполнение элемента управления. Для checkbox или radio возвращаемое значение равно этому атрибуту.
    - checked — указывает, что элемент управления активизировано, если он является checkbox или radio.
    - size — задает видимый размер элемента управления в символах.
    - maxlength — задает максимально возможное количество символов для поля ввода.
    - src — URL изображения для формирования кнопки с картинкой для элемента управления image.
    - align — задает вертикальное выравнивание изображения для элемента тиап image.
    - accept — определяет MIME-тип файла для элемента управления file.
    - type — тип элемента управления. Возможные значения перечислены ниже.
        - text — поле ввода на одну строку. Часто используется вместе с атрибутами size и maxlength.
        - textarea — поле ввода в несколько строк.
        - file — поле для выбора имени файла для его последующей передачи в качестве присоединенных данных. Часто используется вместе с accept.
        - password — поле для ввода пароля на одну строку, заменяющее вводимые символы на звездочки.
        - checkbox — флажок для ввода значения булевского типа (да/нет). Несколько таких флажков с одинаковыми именами могут использоваться для определения некоторого множества. Каждое значение передается на сервер в виде пары name/value, поэтому атрибуты name и value для флажков лучше указывать.
        - radio — радио-кнопка для ввода значения из нескольких возможных. Все кнопки в группе, определяющей возможные значения, должны иметь одинаковые имена.
        - submit — определяет кнопку, при нажатии на которую результаты заполнения формы отправляются на сервер. Атрибут value задает надпись на кнопке.

        - image — кнопка с изображением на ней, при нажатии передает на сервер координаты той точки, в которой был выполнен клик, в виде пар name.x=value, name.y=value.
        - reset — кнопка, сбрасывающая значения всех элементов управления на данной форме к исходным значениям.
        - hidden — скрытый элемент управления, не видимый пользователю, но содержащий информацию, которая может быть передана на сервер.
        - button — определяет пользовательскую кнопку.
- <select/> — элемент управления для выбора одного или нескольких пунктов из заданного списка (меню). Эти пункты определяются вложенными элементами <option>. Может иметь следующие атрибуты.
    - name — имя элемента.
    - multiple — включает возможность множественного выбора.
    - size — количество видимых пунктов списка.
    - Вложенные элементы <option> с помощью атрибутов selected (исходно выбранный пункт) и value (значение, передаваемое на сервер) формируют пункты списка. Видимое пользователю название пункта — это текст, следующий за тегом <option>.
- <textarea/> — определяет поле для ввода нескольких строк текста. Для их передачи на сервер стоит использовать метод POST. Может иметь такие атрибуты.
    - name — обязательное имя, используется для идентификации поля при обработке результатов его заполнения.
    - rows — количество строк, видимых на экране.
    - cols — ширина поля в количестве символов.
    - wrap — определяет, выполнять ли перенос слов на новую строку. Возможные значения: off — нет переноса, virtual — перенос слов на экране, на сервер поступает единая строка, physical — перенос на новую строку будет зафиксирован и в тексте, отправленном на сервер.
- action — обязательный атрибут, указывающий URL, к которому нужно обратиться браузеру для обработки данных формы.

На приведенной ниже, на Рисунке 3, странице показано несколько форм.

- Первая — форма поиска — содержит поле ввода и кнопку Search (вверху справа). Она описана с помощью следующего кода.

```
<form action="http://mysearch.intel.com/corporate/default.aspx"
method="get" name="FORMSearchHeader">
  <input type="hidden" id="culture" name="culture" value="en_US" />
  <label for="q">Search</label>
  <span><input type="text" id="q" name="q" size="10" maxlength="255"
              class="HAT-searchbox" />
  </span>
  <input type="hidden" name="category" id="category" />
  <input type="hidden" name="adv" id="adv" />
  <input type="hidden" name="value" id="value" />
  <input class="searchbutton" type="image" style="width: 57px; height:
20px; border: 0px" name="searchsubmit" id="searchsubmit" alt="Search"
src="/common/images/en-us/searchbtn.gif" />
</form>
```

- Другая хорошо видимая форма — форма входа на сайт для зарегистрированных пользователей — находится справа. Ее код примерно такой

```
<form action="https://ssl.software.intel.com/en-
us/login/?Lang=ENG&amp;TARGET=http://software.intel.com/en-
us/articles/non-commercial-software-development" method="post">
  <input type="hidden" name="qlogin" value="true" />
  <div class="row">
    <div class="key">Login ID:</div>
    <div class="value"><input id="txtUserName" name="txtUserName"
class="login" size="12" type="text" /></div>
  </div>
  <div class="row">
    <div class="key">Password:</div>
    <div class="value"><input id="txtPassword" name="txtPassword"
class="login" size="12" type="password" /></div>
  </div>
  <div class="row">
    <div class="key">Remember Me?</div><div class="value"><input
id="chkRememberMe" type="checkbox" name="chkRememberMe" value="on"
/></div>
  </div>
  <div style="clear: both"></div>
  <input type="submit" class="login" value="Login" /><br />
</form>
```

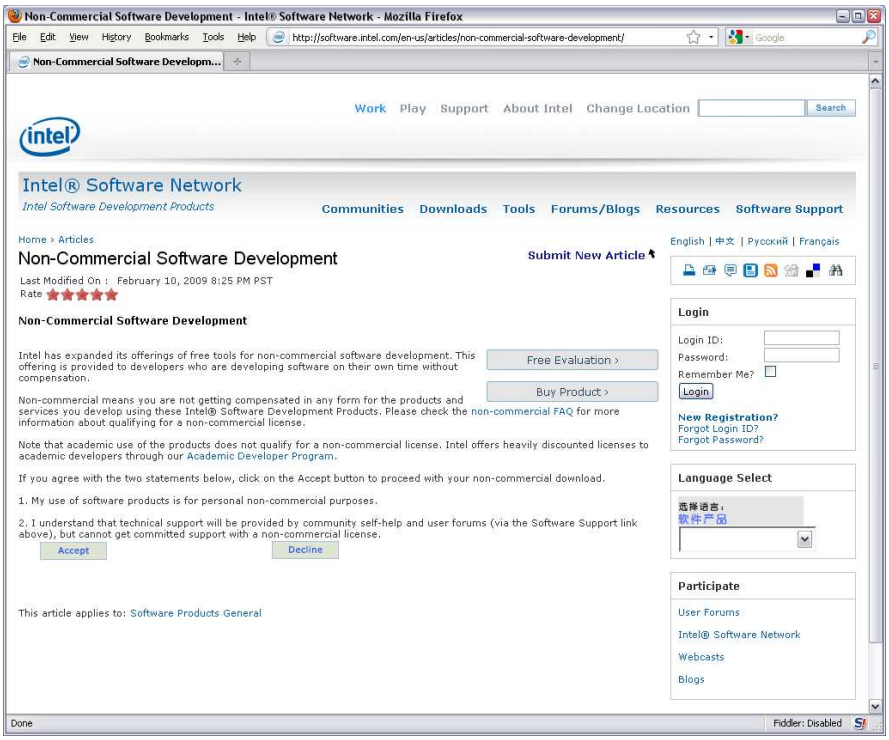

Рисунок 3. Вид HTML-страницы с формами в браузере.

- Еще одна форма, для выбора языка, тоже находящаяся справа, описана так

```
<form method="get" action="">
  <select style="width:160px;" name="lang"
onChange="window.location.href=this.options[this.selectedIndex].value;"
>
    <option selected="selected" value="#"></option>
    <option
value="http://www.intel.com/cd/software/products/apac/zho/index.htm">&#
31616;&#20307;&#20013;&#25991;</option>
    <option value="http://www.intel.com/cd/software/products/asmo-
na/eng/index.htm">English</option>
    <option
```

```
    value="http://www.intel.com/cd/software/products/ijkk/jpn/index.htm">&#
    26085;&#26412;&#35486;</option>
        <option
    value="http://www.intel.com/cd/software/products/apac/kor/index.htm">&#
    54620;&#44397;&#50612;</option>
        <option
    value="http://www.intel.com/cd/software/products/emea/rus/index.htm">&#
    1056;&#1091;&#1089;&#1089;&#1082;&#1080;&#1081;</option>
      </select>
    </form>
```

Обработка форм

При обработке форм занесенные в них данные пересылаются браузером с помощью указанного в атрибуте method метода (GET или POST) по адресу, указанному в атрибуте action.

Если используется метод GET, то данные формы записываются в сам запрос в виде параметров со значениями, т.е. в конце указанного в action адреса добавляется знак вопроса, а затем через символ ‘&’ для каждого установленного элемента управления перечисляются его имя (то, что указано в атрибуте name), знак равенства ‘=’, и установленное значение.

Установленными считаются только элементы управления, находящиеся внутри форм, имеющие имя, не имеющие атрибута disabled. Для форм с несколькими кнопками типа submit только одна такая кнопка, нажатая пользователем, считается установленной. Из группы радио-кнопок с одинаковыми именами установлена только та, что включена. Для меню имя указывается в элементе <select/>, а значение является списком значений value выбранных элементов <option/>. Если же ни один пункт списка не выбран, такой элемент управления также считается не установленным.

Для элементов выбора файлов текущим значением является список выбранных пользователем имен, но при обработке формы в качестве значения посылается содержимое этих файлов. Поскольку передаваемый так объем данных может быть достаточно большим, для этого лучше использовать метод POST. При этом лучше указывать в атрибуте enctype в качестве способа кодирования данных multipart/porm-data.

В таком случае формируется составное сообщение MIME, в котором значения установленных элементов управления помещаются в отдельные части. В каждой части может быть указан заголовок Content-Type с некоторым значением (по умолчанию, text/plain), должны быть указаны заголовок Content-Disposition со значением “form-data” и имя элемента управления, чье значение содержится в данной части.

Например, вот такая форма

```
<html>
  <head><title>Factorial</title></head>
  <body>
    <h1>Factorial Calculation </h1>
    <p>Input a number </p>

    <form method="get">
      <input type="text" name="value" size="25">
      <p></p>
      <input type="submit" value="Submit">
      <input type="reset" value="Reset">
    </form>
  </body>
</html>
```

при вводе значения 200 в текстовое поле и нажатии на кнопку Submit формирует запрос http://localhost:8080/factorial/?value=200.

Язык каскадных таблиц стилей

Язык каскадных таблиц стилей (CSS, Cascading Style Sheets) [11,14] предназначен для описания представления данных, описанных на каком-либо языке разметки. Описание представления означает, что инструкции CSS указывают браузеру как и где показывать текст, рисунок или иное содержимое, описанное с помощью XML или HTML.

Инструкции CSS могут определяться в отдельном файле, находимом браузером по ссылке, указываемой в элементах <link rel="stylesheet" type="text/css" href="…"/> в заголовке HTML-страницы, или внутри самой HTML-страницы в элементах <style type="text/css"/>.

Сама инструкция CSS представляет собой определение стиля, т.е. набор свойств представления (шрифт, размер текста, цвет текста и подложки, форматирование абзаца, границы и пр.) и их значений, привязываемое к селекторам, задающим виды элементов, к которым данный стиль применяется.

Например, так может быть определен стиль, применяемый ко всем элементам со значением атрибута class="important".

```
.important { font-weight: bold; color: red; }
```

А вот так этот стиль может применяться в HTML-страничке

```
<p> Это неважное утверждение.
<span class="important"> А вот это — важное. </span>
```

Аналогично, можно определять стили для идентификаторов. Отличие от классов в том, что идентификатор элемента должен быть уникален в HTML-документе.

```
#most_important_statement { font-weight: bold; color: red; }
```

Инструкция, привязывающая стиль к конкретным элементам HTML выглядит так.

```
h1, h2 { font: Tahoma; color: red; background-color: yellow; }
```

Она означает, что заголовки типов h1 и h2 будут представлены в данном стиле.

Если же нужно модифицировать представление только заголовков, имеющих значение атрибута class, равное special, то можно воспользоваться такой инструкцией.

```
h1.special, h2.special { color: red; background-color: yellow; }
```

Селекторы HTML-элементов могут быть вложенными или связанными.

```
table p { font-weight: bold; }
```

Приведенная выше инструкция выделяет жирным содержимое элементов <p> внутри таблиц, другие элементы <p> останутся незатронутыми.

```
i > b { font-weight: normal; }
```

Эта инструкция отключает полужирный шрифт для элементов <b/>? непосредственно вложенных в элементы <i/>.

```
p + h2 { margin-top: 20px; }
```

Приведенная выше инструкция установит отступ сверху для всех заголовком второго уровня, непосредственно следующих за элементами <p> в 20 пикселей.

Литература

[1] Web-сайт консорциума World Wide Web http://www.w3.org/.

[2] http://www.xml.com/.

[3] XML 1.1, 2004. Доступен через http://www.w3.org/TR/xml11/.

[4] Annotated XML 1.0, 1998. Доступен через http://www.xml.com/axml/axml.html.

[5] Расширяемый язык разметки (XML) 1.0 (русский перевод первой версии стандарта). Доступен через http://www.rol.ru/news/it/helpdesk/xml01.htm.

[6] Материалы по XMLSchema http://www.w3.org/XML/Schema.

[7] Namespaces in XML, 1999. Доступен через http://www.w3.org/TR/REC-xml-names/.

[8] Hypertext Transfer Protocol — HTTP/1.1. RFC 2616. Доступно по ссылке http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[9] http://www.opennet.ru/docs/RUS/http/index.html

[10] HTML 4.01 Specification http://www.w3.org/TR/html401/.

[11] http://html.manual.ru/.

[12] http://www.htmlbook.ru/.

[13] Unicode Standard http://www.unicode.org/standard/standard.html.

[14] Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification http://www.w3.org/TR/CSS21/.