САПР

УДК 669.74

Поступила 30.06.2014

А. О. БУРЕНКОВ, ОАО «БМЗ – управляющая компания холдинга «БМК»

РАЗРАБОТКА И ВНЕДРЕНИЕ СИСТЕМ ТЕХНОЛОГИЧЕСКОГО УЧЕТА И КОНТРОЛЯ С ИСПОЛЬЗОВАНИЕМ WEB–ТЕХНОЛОГИЙ

Данная статья носит обзорно-методический характер. Будет полезна работникам служб автоматизации государственных и частных предприятий.

This article has review and methodical character. It will be useful for employees of automation service of the state and private enterprises.

Введение

Начнем мы с понимания того, что такое технологический процесс. Технологический процесс (*техпроцесс*) – это упорядоченная последовательность операций, выполняющихся от исходных данных до получения результата. Завершение каждой технологической операции называется технологическим переходом. Как правило, отдельно взятый производственный процесс на предприятии представляет собой не один техпроцесс, а их совокупность. При проектировании системы контроля необходимо выделить как минимум два уровня: 1) цеховой, к которому будет относиться техпроцесс или совокупность техпроцессов, а также регистрация технологических переходов; 2) корпоративный, уровень на котором будет происходить анализ полученных данных.

Архитектура Клиент-Сервер и протокол гипертекста HTTP

Уяснив, что наша система должна делиться на два уровня, необходимо выбрать архитектуру, на основе которой она будет базироваться. Наиболее себя хорошо зарекомендовавшей и проверенной временем является архитектура Клиент-сервер. Клиент-сервер (англ. Client-server) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчиками услуг, называемыми клиентами. Технология Клиент-сервер является основой работы протокола HTTP (HyperText Transfer Protocol – «протокол передачи гипертекста»). HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с вебсайтов. Несмотря на то что весь интернет работает по этому протоколу, нет никаких препятствий использовать его для построения своего «интернета» в отдельно выбранной интрасети, использующей стек протоколов TCP/IP. В протоколах TCP и UDP (семейства TCP/IP) порт – идентифицируемый номером системный ресурс, выделяемый приложению, выполняемому на некотором сетевом хосте, для связи с приложениями, выполняемыми на других сетевых хостах (в том числе с другими приложениями на этом же хосте). Основное правило, необходимое для понимания работы порта: порт может быть занят только одной программой и в этот момент не может использоваться другой; все программы для связи между собою посредством сети используют порты. Для каждого из протоколов TCP и UDP стандарт определяет возможность одновременного выделения на хосте до 65536 уникальных портов, идентифицирующихся номерами от 0 до 65535. При передаче по сети номер порта в заголовке пакета используется (вместе с IP-адресом хоста) для адресации конкретного приложения (и конкретного, принадлежащего ему, сетевого соединения).

В обычной клиент-серверной модели приложение либо ожидает входящие данные (или запроса на соединение; «слушает порт»; роль сервера), либо посылает данные (или запрос на соединение) на известный порт, открытый приложением-сервером (роль клиента).

По умолчанию приложению выдается порт с произвольным (например, ближайшим свободным, большим 1023) номером. При необходимости

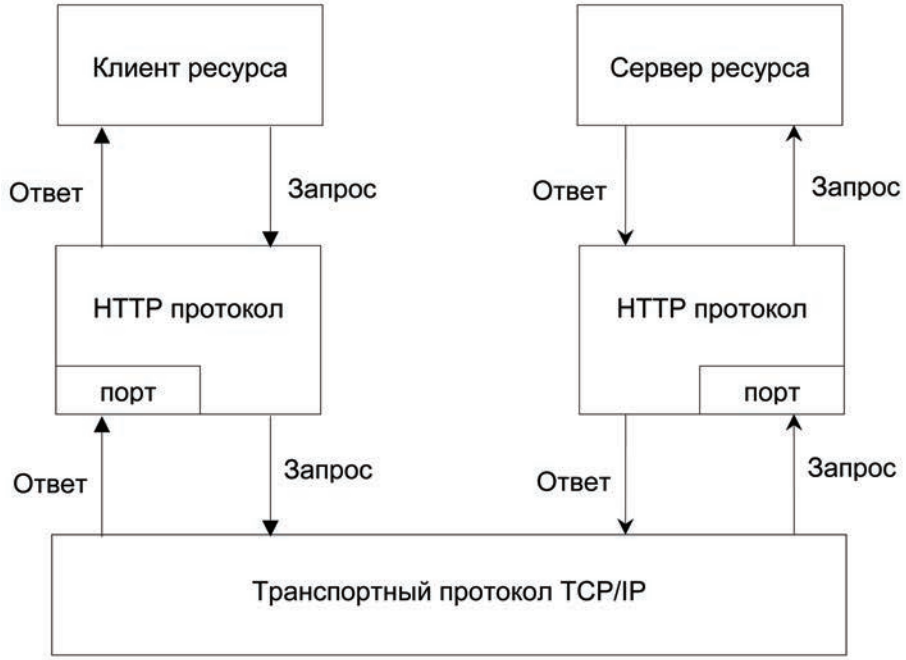

Рис. 1

приложение может запросить конкретный (предопределенный) номер порта. Так, веб-сервера обычно открывают для ожидания соединения предопределенный порт 80 протокола TCP. На рис. 1 показана принципиальная схема работы архитектуры клиент-сервер.

Веб-сервер

Веб-сервер – это сервер, принимающий HTTP-запросы от клиентов, веб-браузеров (а также мобильных телефонов, карманных ПК, бытовой техники, спецоборудования и т. д.) и выдающий им HTTP-ответы, которые могут быть представлены HTML-страницей, изображением, файлом, медиапотоком или другими данными. Веб-сервера – основа Всемирной паутины.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает.

Клиент, которым обычно является веб-браузер, передает веб-серверу запросы на получение ресурсов, обозначенных URL-адресами. Ресурсы – это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ веб-сервер передает клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP. На сегодняшний день список наиболее популярных серверов выглядит так:

1. IIS от компании Microsoft, распространяемый с серверными ОС семейства Windows.

2. Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS. Благодаря тому что изначально он являлся бесплатно-распространяемым продуктом, на сегодняшний день это один из самых распространенных веб-серверов в интернете.

3. nginx – свободно-распространяемый веб-сервер, разрабатываемый Игорем Сысоевым с 2002 г. и пользующийся большой популярностью на крупных сайтах.

4. lighttpd – свободный веб-сервер. Работает в Linux и других Unix-подобных операционных системах, а также в Microsoft Windows. Отличается высокой производительностью.

5. Google Web Server – веб-сервер, основанный на Apache и доработанный компанией Google.

6. Resin – свободный веб-сервер приложений.

7. Cherokee – свободный веб-сервер, управляемый только через web-интерфейс.

8. Rootage – веб-сервер, написанный на java.

9. THTTPD – простой, маленький, быстрый и безопасный веб-сервер.

Если у вас нет опыта разработки веб-приложений и создания веб-сайтов, то стоит остановить свой выбор на IIS от компании Microsoft. Хорошая поддержка, большая база знаний и интуитивно понятный интерфейс управления позволит избежать проблем в установке и настройке веб-сервера и быстро приступить к работе.

Серверные языки программирования

Когда пользователь дает запрос на какую-либо страницу (переходит на нее по ссылке или вводит адрес в адресной строке своего браузера), то вы-

званная страница сначала обрабатывается на сервере, т. е. выполняются все программы, связанные со страницей, и только потом возвращается к посетителю по сети в виде файла. Этот файл может иметь расширения: HTML, PHP, ASP, ASPX, Perl, SSI, XML, DHTML, XHTML.

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

Важной стороной работы серверных языков является возможность организации непосредственного взаимодействия с системой управления базами данных (или СУБД) – сервером, на котором упорядоченно хранится информация, которая может быть вызвана в любой момент. Популярными среди систем управления базами данных являются Firebird, IBM DB2, IBM DB2 Express-C, Microsoft SQL Server, Microsoft SQL Server Express, mSQL, MySQL, Oracle, PostgreSQL, SQLite, Sybase Adaptive Server Enterprise, ЛИНТЕР, MongoDB.

Клиентские языки

Как следует из названия, программы на клиентских языках обрабатываются на стороне пользователя, как правило, их выполняет браузер. Это и создает главную проблему клиентских языков – результат выполнения программы (скрипта) зависит от браузера пользователя, т. е. если пользователь запретил выполнять клиентские программы, то они исполняться не будут, как бы ни желал этого программист. Кроме того, может произойти такое, что в разных браузерах или в разных версиях одного и того же браузера один и тот же скрипт будет выполняться по-разному. Обработка CSS-стилей – основы визуального оформления страниц также сильно зависит от браузера, в котором открыта страница. Поэтому для разработки кроссбраузерных веб-приложений необходимо знать стандарты W3C, предъявляемые к верстке и скриптам, а также использовать механизмы для проверки выполнения скриптов и отображения верстки. Самыми распространенными клиентскими языками программирования являются JavaScript, VBScript, ActionScript, Java.

Мы кратко рассмотрели основные термины, технологии и определения, которые нам необходимы для разработки системы учета. Конечно, мне бы хотелось более детально рассмотреть вопросы выбора веб-сервера, серверного языка программирования, уделить большее внимание классификации техпроцессов, но в отведенных объемах данной статьи сделать это невозможно. Поэтому резюмируем все приведенное выше на рис. 2.

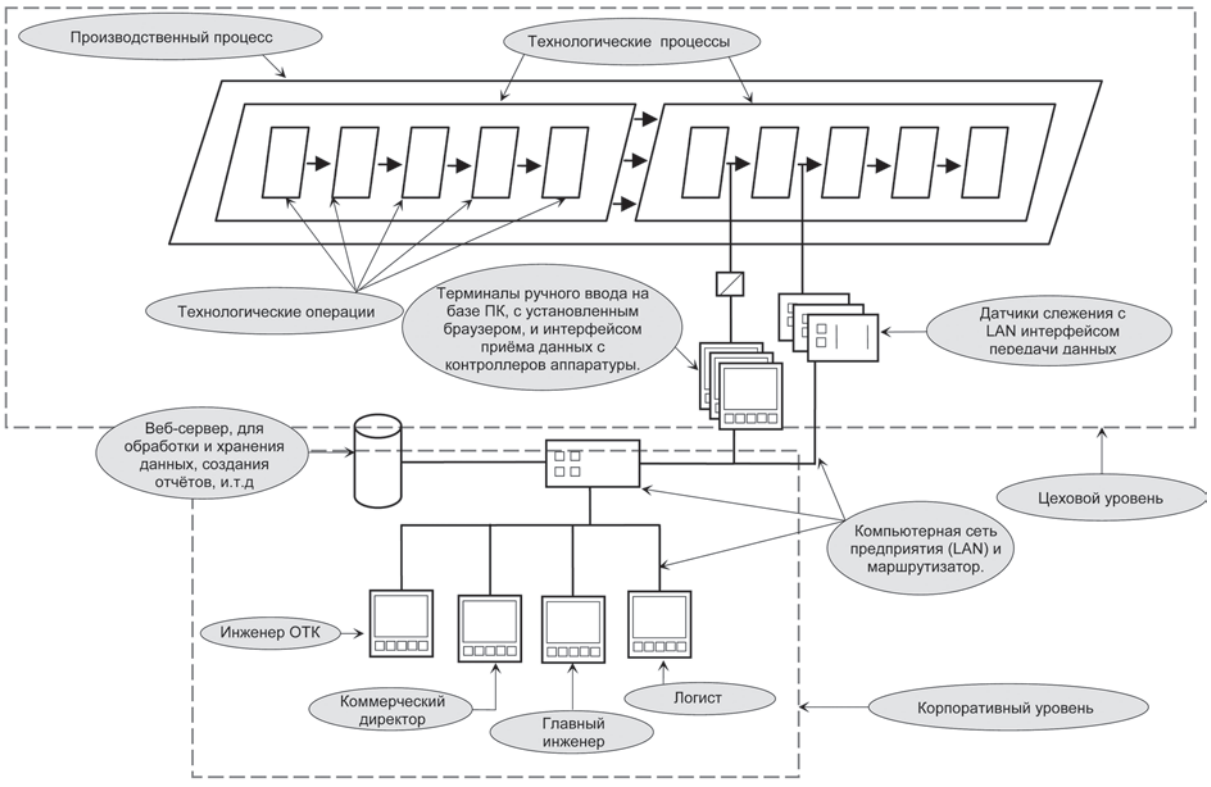

Рис. 2

Поясним диаграмму:

1. Производственный процесс состоит из двух технологических процессов, разбитых на операции.

2. Операторы на терминалах регистрируют технологические переходы, внося полученные значения. Параллельно с ручным вводом осуществляется автоматическая передача данных с датчиков слежения.

3. По средствам компьютерной сети данные поступают на веб-сервер, где происходит их обработка, хранение, а также создание отчетов и представление их в виде таблиц и графиков.

4. Полученные отчеты и результаты доступны для просмотра специалистам руководящего звена. Основываясь на полученных результатах, они вносят изменения в цикл производства для обеспечения надежной и продуктивной работы предприятия.

Серверная и клиентская части веб-приложения

Создаваемое нами веб-приложение необходимо разделить на две части:

• Серверная часть – скрипты, которые будут выполняться на сервере.

• Клиентская часть – скрипты, которые будут работать в браузере на компьютере клиента.

Связываться между собой эти две части будут посредством протокол-HTTP и Ajax-запросов. Разумеется, чтобы разобраться в примерах к этой статье и реализовать поставленную задачу, необходимо обладать хотя бы базовыми знаниями php или asp, а также javascript, ajax, jquery, css, html. Поэтому если знания основ всего перечисленного нет, лучше за дело не браться. Вероятность на успех будет крайне мала и будет жаль потеренного времени.

Серверная часть

Сформируем основную концепцию, как у нас должна работать серверная часть нашего веб-приложения. Концепция должна быть универсальна и не зависеть от того, на каком языке программирования она будет реализована.

1. Структурированные URI-запросы, содержащие в себе название команды, имя модуля, которые должны быть выполнены на сервере. Модулем будет выступать отдельно взятый класс, командой – функция или подпрограмма, входящая в этот класс. Пример такой строки параметров может выглядеть так: http:// <имя хоста>/< исполняемый документ>?module=statistic &cmd=get_report &report_name=stock &….<другие параметры>.

2. Исполняемый файл должен содержать в себе обработчик запроса. Обработчик запроса может также быть представлен в виде функции или подпрограммы. Он должен уметь:

✓ Определять тип http-запроса (POST/GET).

✓ Проверять требует ли этот запрос авторизации, особых привилегий на выполнение и не содержит ли он вредоносного кода, который может навредить работе сайта или базы данных.

✓ Иметь список зарегистрированных в нашем приложении классов.

✓ Возвращать ответ сервера в виде структурированного документа, на основе которого клиентская часть будет формировать окончательный ответ клиенту**.** Структурированный документ может иметь формат XML или JSON.

✓ В целях безопасности исполняемый файл на сайте должен быть только один.

3. Отдельное внимание необходимо уделить логированию операции, запросов, которые отправляют пользователи на сервер, чтобы в случае возникновения ошибок человеческого фактора найти виноватых.

4. Наше серверное приложение должно иметь один сайт и быть модульным. Модульная структура поможет:

✓ Сделать наше приложение гибким и управляемым. Отказ работы отдельного модуля не повредит работе сайта.

✓ Упростить процеес разработки. Несколько программистов могут реализовывать поставленные задачи в отдельных модулях. Им не придется создавать интерфейс, вести учет пользователей, логирование операций и т. д.

✓ Расширять функционал основного модуля (ядра) приложения, обеспечивая целостность и совместимость написаных ранее модулей.

Все описанные выше требования легко реализуемы на трех наиболее популярных платформах веб-программирования, таких, как PHP, ASP, ASP. Net.

Основной модуль (класс) веб-приложения

Теперь подробнее рассмотрим основные элементы нашей концепции. Работа основного модуля представлена на рис. 3.

Поясним диаграмму:

1. Сервер получает запрос от клиента, в котором указаны исполняемый модуль M_5 и команда в этом модуле C_1.

2. Главный модуль находит требуемый M_5 и передает в него команду C_1.

3. Модуль M_5 запрашивает данные из запроса, выполняет их в команде C_1, а полученный результат отдает ядру.

4. Ядро помещает этот результат в XML-структуру и в виде ответа возвращает клиенту.

5. На стороне клиента происходит окончательная обработка ответа веб-сервера.

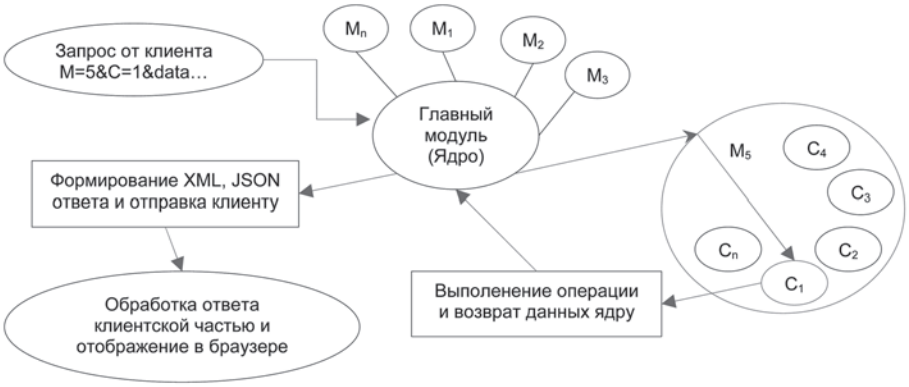

Рис. 3

```
' Пример написания подключаемого модуля
' Название класса (модуля)
Class Test1

   ' внешняя переменная, хранящая псевдоним команды
   ' вместо внешней переменной желательно использовать
   '  Public Property
   ' должна быть в каждом классе
   Public Unit

   ' внешняя функция Init
   ' обработчик псевдонимов команд
   ' должна быть в каждом классе
   Public Function Init()

      ' Обработка псевдонимов
      Select Case Unit
         ' если unit=com1_t1 будет выполнена функция Com1()
         Case "com1_t1"
            Call Com1()
         ' если unit=com2_t1 будет выполнена функция Com2()
         Case "com2_t1"
            Call Com2()
         End Select

      End Function
   ' внутренняя функция Com1() видимая только внутри модуля
   Private Function Com1()

       Com1 = "Result Com1 from Test1"
   End Function
   ' внутренняя функция Com2() видимая только внутри модуля
   Private Function Com2()
        Com2 = "Result Com2 from Test1"
   End Function
End Class
```

```
' Название главного класса (модуля)
Class CORE

   Public Sub Run()
      Dim strExec, module, unit, retVal, core_request_method
      ' опрделяем тип запроса от клиента
      core_request_method =
UCase(Request.ServerVariables("REQUEST_METHOD"))
      ' в зависимости от того какой тип запроса
      ' выбираем необходимые нам значения переменных
      ' module и unit
      If core_request_method = "POST" Then
         module = LCase(Request.Form("module"))
         unit = LCase(Request.Form("unit"))
      ElseIf core_request_method = "GET" Then
         module = LCase(Request.QueryString("module"))
         unit = LCase(Request.QueryString("unit"))
      End If

      ' подготовка диномического кода, после запуска которого
      ' в переменной retVal ,будет харниться результат выполнения
      ' команды клиентского модуля
      strExec = "Dim objModule : Set objModule = New " & module & Chr(13)
& " " & _
                "objModule.Unit = """ & unit & """" & Chr(13) & " " & _
                "retVal = objModule.Init() " & Chr(13) & " " & _
                "Set objModule = Nothing" & Chr(13)

      ' выполнение динамического кода
      Execute(strExec)
      ' создание строковой переменной strOut и подготовка ответа
      Dim strOut : strOut = "<response>"
      strOut = strOut & "<value>" & retVal & "</value>"
      strOut = strOut & "</response>"

      ' указваем тип ответа который мы отправим в браузер клиента
      Response.ContentType = "application/xml"
      ' отправляем ответ
      Response.Write(strOut)

   End Sub
End Class
```

Рис. 4

Необходимо обратить внимание на то, что сторонние модули должны быть написаны однотипно. На рис. 4 приведены упрощенный пример написания модулей, а также пример их вызова. Пример будет написан языке VBScript, который применяется в технологии создания веб-приложений ASP, корпорации «Майкрософт»:

Содержание исполняемого файла, который обычно называется default. asp или index. asp, дано на рис. 5.

Клиентская часть веб-приложения

Наиболее сложная и трудоемкая в разработке часть веб-приложения. Когда программист разрабатывает настольное приложение windows, он не задумывается о том, как ему сделать интерфейс. У него есть готовые формы, окна, поля, текстовые литералы и прочее. Поэтому разработка сводится к тому, чтобы создать нужное количество форм и разместить на них необходимые поля ввода

```
<%@language="VBScript" %>
<%
   ' Scripted by AB
   ' директива проверки уникальности переменных
   Option Explicit
   ' кодовая страница и кодировка ответов сервера, необходимо помнить, что кодировка файлов скриптов должна совпадать с параметром
   ' Charset. Во-избежании проблем с правильным отображением данных и корректной работе с базами данных настоятельно рекомендую
   ' использовать только кодировку UTF-8 или UTF-16
   Response.Charset = "UTF-8"
   Session.CodePage = 65001

   ' серверная деректива #include служит для склеивания  файлов скриптов в один.
   ' максимальный размер файла при котором заметно ухуджение производительности веб-сервера IIS v6 - 200 000 строк.
   ' поэтому не бойтесь включать ваши скрипты в один - скорее всего ваши первые проекты не будут превышать рубеж 200 тыс. строк :-)
%>
<!-- #include virtual='/app_code/core/inc_class_CORE.asp' -->
<!-- #include virtual='/app_code/addons/inc_class_MyClasses.asp' -->
<%
   ' создание экземпляра класса ядра
   Dim objCORE : Set objCORE = New CORE
   ' запуск ядра
   objCORE.Run()
   ' уничтожение экземпляра объекта в памяти веб-сервера
   Set objCORE = Nothing
%>
```

Рис. 5

и контейнеры для отображения. В нашем случае все куда сложнее. Просто на сегодняшний день не существует надежной, завершенной и бесплатной среды разработки для веб, которая бы позволила сделать оконный интерфейс, как в операционной системе Windows или в графической среде GNOME, применяемой в Unix-системах.

Однако стоит упомянуть о наиболее удачных javascript-фреймворках, которые нам помогут в разработке:

1. jQuery – самая популярная и распространенная библиотека на JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Также jQuery предоставляет удобный API по работе с Ajax. Данный фреймворк один из первых удачно реализовал принцип ненавязчивого JavaScript.

2. Prototype JS. Второй по популярности js-фреймворк, который существенно упрощает работу с Ajax и некоторыми другими продвинутыми js-возможностями.

3. MooTools является модульным, объектно-ориентированным фреймворком. Он очень похож в своем подходе к дизайну ядра на jQuery и Prototype. Он не включает в себя UI-контролы и виджеты, ограничиваясь минимальным набором эффектов и возможностей.

4. Библиотека Yahoo! UI Library (YUI) – это набор функций и элементов управления, написанных на JavaScript и CSS, для создания интерактивных web-приложений, использующих техники DOM-скриптинга, DHTML и Ajax.

5. Ext JS. Проект начинался, как попытка расширить вышерассмотренный фреймворк Yahoo! User Interface, но на данном этапе – это уже полностью самобытный, практически никак не связанный с YUI проект.

Я рекомендую вам остановить свой выбор на jQuery. Несмотря на то что этот фреймворк имеет скудный набор UI-элементов, он очень прост в понимании и весьма популярен. Очень много решений можно найти в интернете в блогах пользователей, которые делают на этом фреймворке очень хорошие элементы интерфейса. А доработать их для своих нужд – это всего лишь дело техники и желания.

Мы рассмотрели кратко основные теоретические основы для создания системы контроля и учета техпроцессов на базе веб-приложения. Лично я всегда был за практический подход к делу. Поэтому, те кого это тема заинтересовала, могут скачать пример готового проекта:

http:// infobmz/web/example.zip или

http://www.mediafire.com/download/3fb275bd13m2o4q/example.zip

Код в примере снабжен огромным количеством комментариев. Также написана инструкция по настройке и установке веб-сервера IIS в Windows XP x32, Windows 2003 x32 и Windows 7 x64. Кроме того, я рекомендую установить браузер Mozilla FireFox и два дополнения к нему, которые вам помогут в разборе и тестировании примера:

http:// www.mozilla.org/ru/firefox/new

https:// addons.mozilla.org/ru/firefox/addon/firebug

https:// addons.mozilla.org/ru/firefox/addon/web-developer

Пример написан на языке Vbscript технологии ASP, которая есть по умолчанию в версиях IIS начиная с 3.0. На мой взгляд, это самый простой

язык веб-программирования, который обладает всем необходимым функционалом, простым и понятным синтаксисом. Именно по этим причинам он и был выбран для написания примера. Изучив ASP, и поняв как работают веб-приложения, вы сможете легко сделать следующий шаг – освоить технологию ASP. Net. В примере используется база данных Microsoft Access 2000, не требующая настройки и установки сервера БД. Интерфейс активно использует фреймворк jquery. Дополнительную информацию вы можете найти в списке используемой для написания статьи литературы.

Литература

1. Термины и определения – Википедия, свободная энциклопедия: http://ru.wikipedia.org.
2. Интернет ресурс Blogerator. ru – анализ и сравнение java-script фреймворков: http://blogerator.ru/page/javascript_frameworks_2.
3. Документация по по фреймворку jQuery: http://jquery-docs.ru/.
4. Active Server Pages 3.0 для профессионалов ISBN: 5–85582–150–1 – отличный учебник по технологии ASP.