Министерство образования и науки Российской Федерации
Владимирский государственный университет
Кафедра вычислительной техники

Курсовая работа

«Web-приложения»

Выполнил:
студент группы ИВТ-201
Морев Н. В.

Проверил:
Жирков В. Ф.

Владимир 2004

Содержание

1. Введение

Проект World Wide Web зародился в начале 90-х годов в швейцарском институте CERN. Изначально он задумывался, как средство облегчающее поиск библиотечной информации, распределенной между несколькими серверами в CERN. Для этого он использовал концепцию «гипертекста», разработанную ранее Тедом Нельсоном и Ванневаром Бушем.

У WWW было несколько важных отличий от других систем гипертекста, которые тогда уже существовали (Note Code, HyperCard, Gopher):

- В WWW использовались только однонаправленные ссылки, что позволяло ссылаться на другие ресурсы без каких-либо действий со стороны владельца этого ресурса.
- WWW не принадлежал какой-либо компании, поэтому можно было создавать программное обеспечение независимо и без всяких лицензионных ограничений.

Эти черты сыграли большую роль во взрывном росте популярности WWW в середине 90-х годов, когда вышел веб-браузер Mosaic и последовавший за ним коммерческий аналог Netscape Navigator.

WWW был построен на трех главных стандартах: URL (RFC2396), HTTP (RFC2616) и HTML. Самый молодой из них и следовательно самый развивающийся — HTML. Изначально он был задуман исключительно для семантической разметки текста (ссылки, разделы, цитаты и пр.), затем всвязи с коммерциализацией веба к этому стандарту стали добавляться различные, часто не совместимые друг с другом тэги для художественного оформления текста. В последствии эта тенденция была признана неправильной (по причинам отсутствия стандартизации и отсутствия разделения смысловой и оформительской части документа) и консорциумом W3C, который разрабатывает стандарт HTML, был взят обратный курс — на оставлении в HTML только семантической разметки, для оформления же был придуман специальный стандарт CSS (каскадные списки стилей).

Еще одна ярко выраженная тенденция в развитии WWW — это совершенствование средств поддержки интерактивности веб-страниц. На стороне клиента это HTML-формы, JavaScript — скриптовый язык программирования, встраиваемый в браузер, и DOM (Объектная модель

документа) — стандарт описывающий способ работы с частями HTML-документа из языков программирования, в частности из JavaScript. На стороне сервера — CGI-интерфейс, различные скриптовые и компилируемые языки программирования, взаимодействующие с пользователем через этот интерфейс, а впоследствии и сервера веб-приложений, ориентированные на разработку приложений с веб-интерфейсом на определенном языке программирования (Java, Python, и т.д.).

Со временем возможности WWW по взаимодействию с пользователем доросли до определенного уровня, на котором стало возможным создание полноценных Web-приложений.

2. Web-приложения

Web-приложение — это приложение, работающее на платформе Web, т.е. использующее для взаимодействия с пользователем веб-сервер, работающий по протоколу HTTP и браузер, интерпретирующий страницы HTML. По-другому можно сказать, что это некоторый сайт, содержимое которого изменяется динамически на основе взаимодействия с пользователем.

2.1. Примеры

Наиболее известные примеры веб-приложений: веб-почта (*http://mail.ru*), интернет-магазины (*http://amazon.com*), онлайновые аукционы (*http://ebay.com*). Однако область применения веб-приложений гораздо шире, чем электронный бизнес, они используются во многих научных и коммерческих областях.

2.2. Организация

Хотя возможны различные вариации, чаще всего используется трехуровневая архитектура для построения веб-приложений: веб-браузер, какая-нибудь технология предоставления динамического веб-контента и база данных. Веб-браузер посылает запросы среднему уровню, который об-

служивает их, производя запросы к базе данных, и представляя результаты в пользовательском интерфейсе.

Например, рассмотрим самую популярную на сегодняшний день платформу для веб-приложений, т. н. LAMP — Linux, Apache, MySQL, PHP. Здесь роль среднего уровня играет веб-сервер Apache с установленным на нем модулем поддержки скриптового языка программирования PHP. А в качестве базы данных выступает MySQL.

2.3. Достоинства и недостатки

Достоинства:

- независимость от клиентской, а часто и серверной платформы;
- нетребовательность к ресурсам клиента (используется т.н. тонкий клиент — браузер);
- не требует установки на клиентский компьютер какого-либо программного обеспечения кроме браузера;
- легкость обновления версий веб-приложения, т.к. оно обновляется только один раз — на сервере;
- «встроенные» сетевые возможности (возможность работать с несколькими клиентами одновременно);
- сохраняются все данные при переходе клиента с машины на машину.

Недостатки и способы их обхода:

- низкое время реакции — ориентация на клиентские технологии, типа JavaScript, DOM, Flash, XUL;
- протокол HTTP не хранит состояния — механизм cookies, сессии;
- малая защищенность — защищенные соединения HTTPS, авторизация встроенная в HTTP или на основе сессий;
- неразвитость языка HTML в смысле форм — разработка стандарта XForms, Flash, Java applets.

3. Современное состояние и перспективы

Сегодня веб-приложения набирают популярность. Самыми посещаемыми сайтами становятся не чисто информационные, гипертекстовые сайты, а те, которые предоставляют какой-либо сервис, как-либо взаимодействуют с пользователем. Но даже и обычные информационные сайты часто используют системы управления контентом для удобства управления информацией, так что и их тоже можно причислить к веб-приложениям.

Некоторые уже ставят веб в один ряд с Windows, Linux и проч., в качестве новой платформы для выполнения приложений (*http://www.joelonsoftware.com/articles/APIWar.html*).

Развитие веб-технологий идет полным ходом. В данный момент наиболее интересными и перспективными технологиями представляются:

- XForms — новый стандарт для описания интерфейсов на XML, учитывающий недостатки форм в HTML. Он добавляет к стандартным возможностям HTML-форм проверку правильности заполнения, обмен данными в формате XML и использование стандартных XML-технологий, независимость от платформы.

- XMLHttpRequest дает возможность послать на сервер какие-либо данные без полной перезагрузки страницы.

- XUL — еще один стандарт для описания интерфейсов, который используется в Mozilla и других, созданных на его основе браузерах, благодаря чему получил сейчас большое распространение в качестве технологии для создания т.н. «rich» веб-приложений, т.е. обладающих большим набором возможностей на стороне клиента, чем традиционные веб-приложения.

Среди интересных серверных технологий можно выделить т.н. серверы веб-приложений. Они включают в себя функциональность обычного веб-сервера, но при этом более ориентированы на выполнение приложений, а не раздачу статических страниц. Это выражается в наличии различных кэшей для ускорения приложений, удобного интерфейса администрирования и поддержке различных технологий разработки приложений.

4. Построение веб-приложений

В этой работе основное внимание будет уделено разработке веб-приложений. В качестве платформы для разработки выбрана Java, т.к. эта технология на данный момент является наиболее развитой, стандартизированной, применяемой на уровне Enterprise и классической (т.е. из нее в основном идут заимствования в других платформах). Она обладает следующими преимуществами:

- Большое сообщество опытных пользователей, программистов, администраторов. Не возникает проблем с поиском интересующей информации.
- Стандартизированность и постоянное развитие через Java Community Process — стандартный процесс, позволяющий заинтересованным сторонам участвовать в разработке будущих версий платформы.
- Большое количество и разнообразие разработанных библиотек.
- Переносимость между различными платформами.

4.1. Технологии построения Web-приложений

В данном разделе будут рассмотрены технологии среднего уровня для построения веб-приложений. Старейшей из них является CGI (Common Gateway Interface) — это спецификация, которая задает способы взаимодействия обычных приложений, выполняющихся на серверной машине, с браузером пользователя по протоколу HTTP. В частности, код для отображения на странице и HTTP-заголовки передаются программой в стандартном потоке вывода, а параметры в программу передаются через переменные окружения. CGI позволяет писать сервеные скрипты на каком угодно языке, лишь бы он поддерживался программным обеспечением, установленным на сервере. Наиболее часто используемыми такими языками были Perl, C, bash (язык командной строки Unix).

Основным недостатком такого подхода была неустойчивость под высокими нагрузками на сервер, связанная с тем, что каждый запущенный скрипт создавал новый процесс на сервере (как известно операции с

процессами довольно дорогостоящи), кроме того у такого подхода возникали сложности с хранением промежуточных данных между вызовами скриптов или общих для всех скриптов данных (приходится их где-то сохранять, что также связано с дополнительными накладными расходами). Решением всего этого стало встраивание специальных модулей непосредственно в веб-сервер Apache (например, модуль PHP). Теперь промежуточные данные могли храниться в оперативной памяти, новых процессов каждый раз не созавалось, и все скрипты могли иметь общие разделяемые ресурсы, такие как соединение с базой данных (установка соединения тоже занимает определенное время). Кроме PHP существуют также модули для Perl, Python и других скриптовых языков.

Следующим шагом в развитии платформ для веб-приложений стали сервера веб-приложений (Apache Tomcat, Zope), ориентированные на какой-либо один язык программирования. Несмотря на то, что языки используются скриптовые, довольно медленные, достигается неплохое увеличение производительности. В первую очередь это происходит из-за кэширования всего, что только можно в оперативной памяти: программы не перекомпилируются каждый раз, когда к ним обращаются, существует общий пул соединений с базой данных. Кроме того достигается масштабируемость за счет использования языков с хорошей поддержкой многопоточности.

4.2. Java технологии

Язык Java изначально позиционировался производителем, как язык для Интернет. С пользовательскими приложениями у Java не сложилось, но зато на стороне сервера Java применяется очень широко и имеет большое количество различных интересных возможностей.

Прежде всего существует несколько конкурирующих серверов веб-приложений, которые несмотря на различия придерживаются некоторых стандартов, установленных Sun, а значит большинство приложений без каких-либо значительных модификаций могут быть перенесены с сервера на сервер.

Кроме того существует несколько разного уровня сложности и с разными подходами фреймворков для разработки веб-приложений (т.е. библиотек классов, на основе которых строится веб-приложение). Это

фреймворки для структурирования приложений на основе паттерна MVC (Struts, Spring), библиотеки для построения шаблонов веб-страниц (JSTL, Velocity, Java ServerFaces), библиотеки для отображения реляционной таблицы на объекты и обратно (Hibernate).

4.3. Подходы к построению Web-приложений

Подходы к построению веб-приложений развивались эволюционно. Самым простым и вероятно самым первым подходом было писать веб-приложения точно также, как обычные консольные приложения. Т.е. вся логика и весь вывод программы находится непосредственно в одном файле и никак не разделены. Пример:

```
word = "world";
display("<h1>Title</h1>");
display("hello " + word);
```

Это было связано с определенными неудобствами. Самое заметное из них заключалось в том, что программа загромождалась огромным количеством вызовов процедуры вывода текста странички. Поэтому вывод шаблонных данных и непосредственно программу решено было условно разделить с помощью специальных синтаксических конструкций (как в PHP или JSP). Пример:

```
<% word = "world"; %>
<h1>Title</h1>
hello <% display(word) %>
```

И как только избавились от самого заметного недостатка исходного подхода, внимание было обращено на другой, скрывавшийся в его тени, — проблема разделения логики и представления. Для этого в рамках использования предыдущего подхода налагают еще одно ограничение: в тексте шаблона должны присутствовать только инструкции, влияющие на представление информации, а вся логика уходит в отдельные классы, описывающие модель приложения.

4.3.1. Паттерн MVC

Наиболее подходящим для построения веб-приложений оказался паттерн объектно-ориентированного программирования под названием MVC

(Model, View, Controller). Кроме того этот паттерн является рекомендованным Sun паттерном для разработки интерактивных программ. Он также используется при построении обычных десктопных приложений.

Суть его в том, что приложение организуется в виде трех различных модулей: один для модели приложения, включающей представление данных и логику системы, второй для представления данных и ввода данных от пользователя и третий для контроллера, который обрабатывает запросы и передает управление различным модулям.

Достоинства такого подхода к проектированию системы заключаются в следующем. MVC разделяет различные аспекты проектирования (хранение данных, поведение системы, представление и управление), дает возможность повторного использования кода, централизует управление выполнением программы и упрощает внесение изменений в систему.

Существуют вариации MVC называемые Model 1 и Model 2, также известные как Page Centric и Servlet Centric соответственно. Их различие в том, что в первом код контроллеров содержится в JSP-файлах (см. рис. 1), а во втором он содержится в сервлетах, которые передают управление JSP-файлам в конце своего выполнения (см. рис. 2). Достоинство последнего способа в том, что он позволяет выбрать показываемую пользователю страницу на основе данных, содержащихся в запросе. Кроме того он улучшает разделение различных аспектов приложения и увеличивает уровень абстракции классов.

В Java самым известным фреймворком для реализации паттерна MVC является Jakarta Struts. В его состав входит готовый конфигурируемый сервлет контроллера и базовые классы, расширяя которые можно писать обработчики запросов пользователя. С использованием этой библиотеки классов и будет продемонстрирован пример веб-приложения.

5. Пример построения Web-приложения: Интернет-магазин

Программа представляет собой уже ставший классическим пример — интернет-магазин. Данный пример упрощен — в нем отсутствуют некоторые возможности, необходимые для настоящего интернет-магазина,

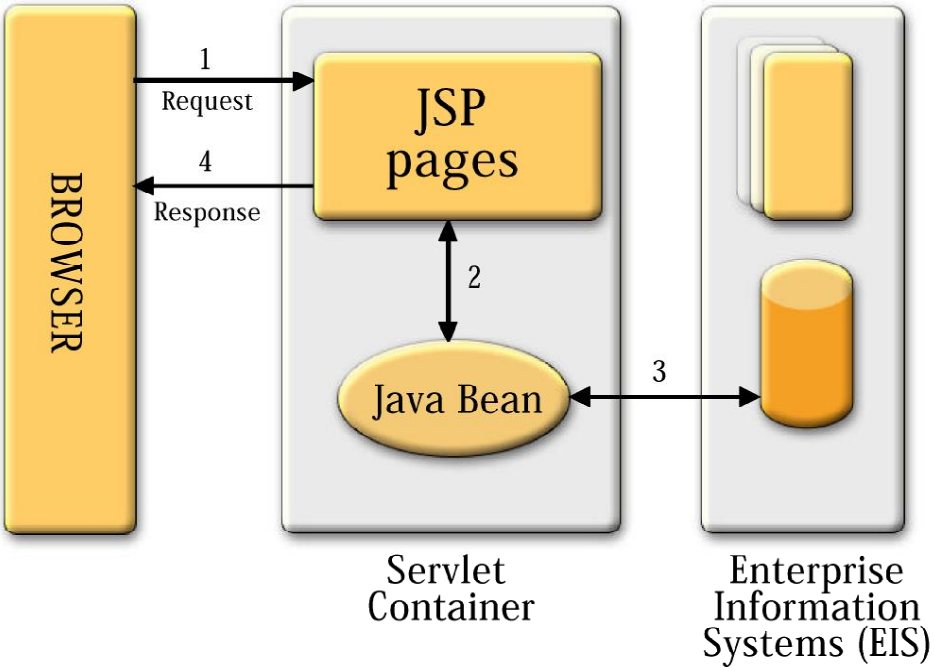

Рис. 1. MVC Model 1 (Page Centric)

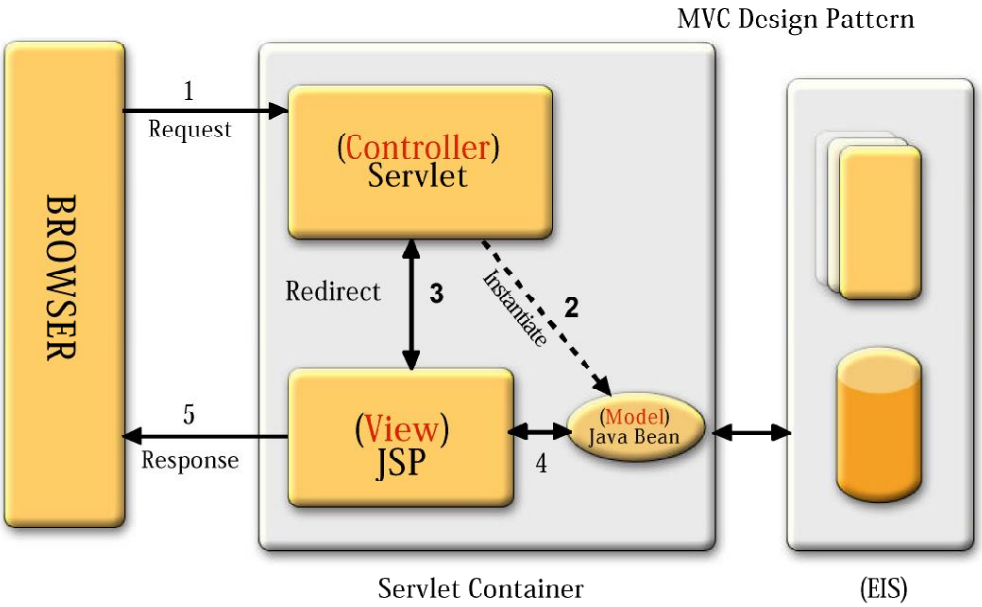

Рис. 2. MVC Model 2 (Servlet Centric)

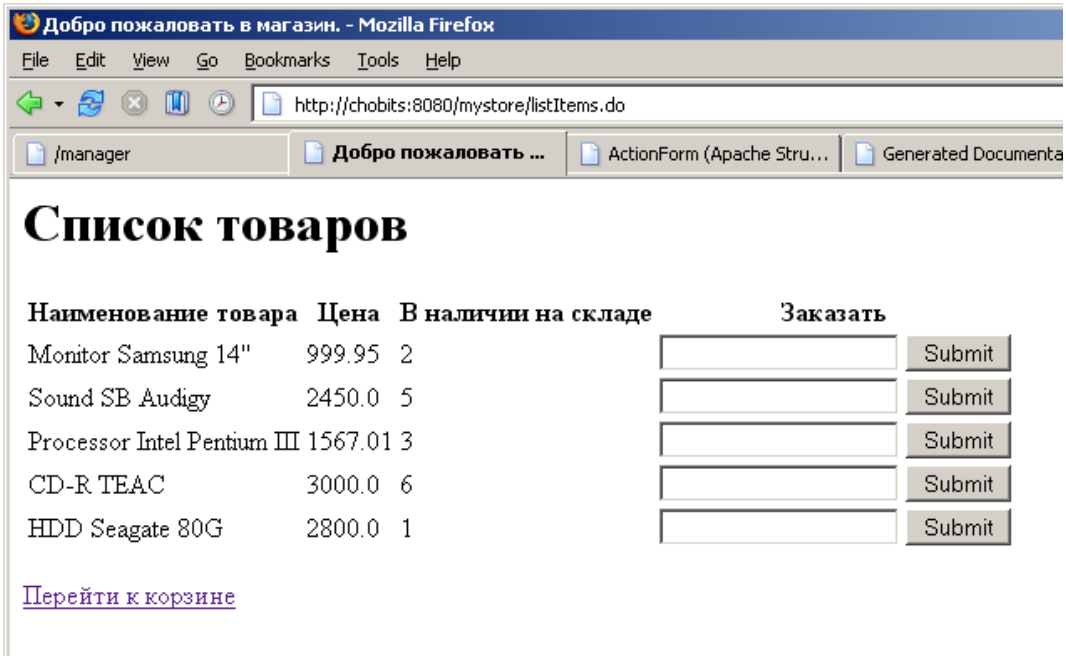

Рис. 3. Список товаров

но зато он иллюстрирует основные возможности сервлетов, библиотеки стандартных тегов, библиотеки Struts и пр.:

- Использование тэгов JSTL для вывода одиночных данных и списков, передаваемых из сервлета в JSP-страницу.
- Разбиение кода на представления (JSP), обработчики запросов, формы, модель.
- Выборка данных из БД с использованием JDBC.
- Хранение данных, связанных с пользовательской сессией.
- Работа с формами, в т.ч. проверка правильности заполнения.

5.1. Описание приложения

Магазин позволяет пользователю просмотреть список товаров (рис. 3) и добавить определенное количество товара в свою корзину (рис. 4).

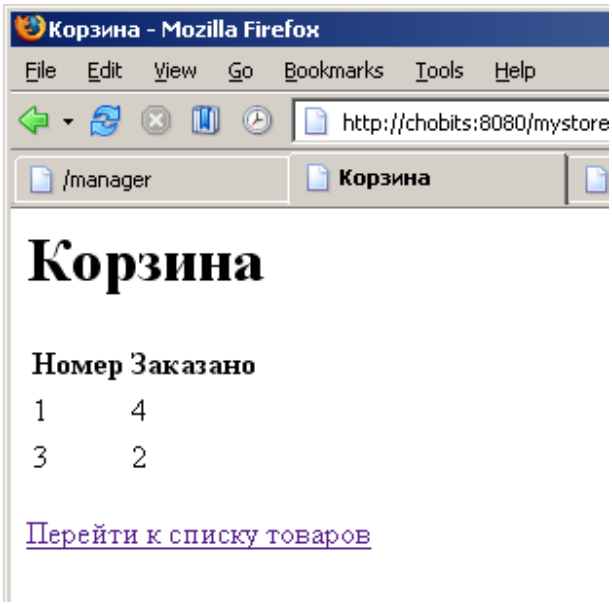

Рис. 4. Корзина

При вводе в форму неправильного значения (например пустого) пользователю выводится соответствующее сообщение об ошибке и предлагает исправить допущенную ошибку (рис. 5).

5.2. База данных

```
CREATE TABLE `items` (
  `id` int(11) NOT NULL default '0',
  `name` varchar(255) default NULL,
  `price` float default NULL,
  `quantity` int(11) default NULL,
  PRIMARY KEY  (`id`)
);

+----+-----------------------------+---------+----------+
| id | name                        | price   | quantity |
+----+-----------------------------+---------+----------+
|  0 | Monitor Samsung 14''        |  999.95 |        2 |
|  1 | Sound SB Audigy             |    2450 |        5 |
|  2 | Processor Intel Pentium III | 1567.01 |        3 |
|  3 | CD-R TEAC                   |    3000 |        6 |
|  4 | HDD Seagate 80G             |    2800 |        1 |
+----+-----------------------------+---------+----------+
5 rows in set (0.00 sec)
```

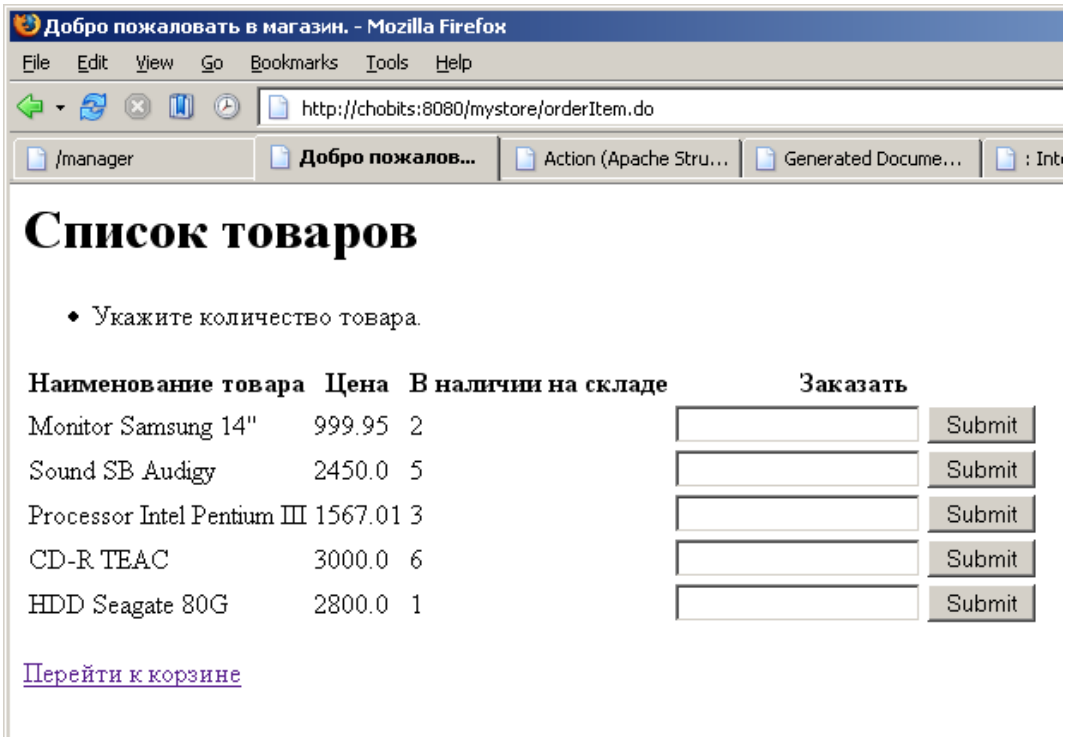

Рис. 5. Ошибка заполнения формы

5.3. Структура

Уровень представления состоит из JSP-страниц list.jsp — список товаров, cart.jsp — содержимое корзины и класса OrderForm — форма заказа товара.

Уровень логики состоит из классов ListItemsAction — считывает из базы данных список товаров и передает его для отображения в list.jsp, OrderItemAction — добавляет товар и количество в корзину, которая является списком видимым на уровне сессии, ShowCartAction — считывает товары и количество из корзины и передает для отображения в cart.jsp.

Уровень модели приложения состоит из класса Item, представляющего собой модель товара.

5.4. Конфигурация

Веб-приложения в Java конфигурируются через специальный дескриптор веб-приложений web.xml. Для использования с библиотекой Struts берется стандартный дескриптор, входящий в поставку Struts.

Конфигурация библиотеки Struts осуществляется через файл struts-config.xml. В нем указываются: источники данных для выборки из базы данных, классы форм и классы действий.

Листинг 1. struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">

<struts-config>
<data-sources>
    <data-source>
        <set-property property="driverClass" value="com.mysql.jdbc.Driver"/>
        <set-property property="url" value="jdbc:mysql://localhost/store"/>
        <set-property property="user" value="root"/>
    </data-source>
</data-sources>

<form-beans>
    <form-bean
        name="orderForm"
```

```
        type="com.koliamorev.OrderForm"/>
</form-beans>

<action-mappings>
    <action
        path="/listItems"
        type="com.koliamorev.ListItemsAction"
        scope="session"
        input="/list.jsp">

        <forward name="success" path="/list.jsp"/>
    </action>

    <action
        path="/orderItem"
        type="com.koliamorev.OrderItemAction"
        name="orderForm"
        scope="session"
        input="/list.jsp">

        <forward name="success" path="/showCart.do"/>
    </action>

    <action
        path="/showCart"
        type="com.koliamorev.ShowCartAction"
        input="/list.jsp">

        <forward name="success" path="/cart.jsp"/>
    </action>

</action-mappings>

<message-resources parameter="resources.application"/>
```

5.5. Текст сервлетов

Листинг 2. Item.java

```
package com.koliamorev;

public class Item {
    int id;
    String name;
    float price;
    int quantity;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
```

```
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public float getPrice() { return price; }
    public void setPrice(float price) { this.price = price; }

    public int getQuantity() { return quantity; }
    public void setQuantity(int quantity) { this.quantity = quantity; }

};
```

Листинг 3. ListItemsAction.java

```
package com.koliamorev;

import javax.servlet.http.*;
import org.apache.struts.action.*;

import java.util.List;
import java.util.ArrayList;
import java.sql.*;
import javax.sql.DataSource;

import com.koliamorev.Item;

public class ListItemsAction extends Action {
    private List items;
    private Connection connection;

    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, Ht

        try {
            DataSource dataSource = getDataSource(request);
            connection = dataSource.getConnection();
            populate();
        } catch (Exception e) {
        }

        request.getSession().setAttribute("goods", items);

        return mapping.findForward("success");
    }

    void populate() throws SQLException {
        Statement s = connection.createStatement();
        String sql = "SELECT id, name, price, quantity FROM items";
        ResultSet rs = s.executeQuery(sql);
        items = new ArrayList();
        while (rs.next()) {
            Item item = new Item();
            item.setId(rs.getInt(1));
            item.setName(rs.getString(2));
            item.setPrice(rs.getFloat(3));
```

```
            item.setQuantity(rs.getInt(4));
            items.add(item);
        }
        rs.close();
        s.close();
    }
}
```

Листинг 4. OrderForm.java

```
package com.koliamorev;

import org.apache.struts.action.*;
import javax.servlet.http.*;

public class OrderForm extends ActionForm {
    private String id;
    private String quantity;

    public String getId() {
        return id;
    }

    public void setId(String string) {
        id = string;
    }

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String string) {
        quantity = string;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        id = null;
        quantity = null;
    }

    public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
        ActionErrors errors = new ActionErrors();

        if (quantity==null || quantity.trim().equals("")) {
            errors.add("quantity",
                    new ActionError("orderItem.quantity.problem"));
        }

        return errors;
    }
}
```

Листинг 5. OrderItemAction.java

```
package com.koliamorev;

import javax.servlet.http.*;
import org.apache.struts.action.*;

import java.util.List;
import java.util.ArrayList;

public class OrderItemAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, Ht

        OrderForm orderForm = (OrderForm) form;
        HttpSession session = request.getSession();
        List cart = (List)session.getAttribute("cart");

        if (cart == null) {
            cart = new ArrayList();
        }

        Item item = new Item();
        item.setId(Integer.decode(orderForm.getId()).intValue());
        item.setQuantity(Integer.decode(orderForm.getQuantity()).intValue());

        cart.add(item);
        session.setAttribute("cart", cart);

        return mapping.findForward("success");
    }
}
```

Листинг 6. ShowCartAction.java

```
package com.koliamorev;

import javax.servlet.http.*;
import org.apache.struts.action.*;

import java.util.List;
import java.util.ArrayList;

public class ShowCartAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, Ht

        HttpSession session = request.getSession();

        List cart = (List)session.getAttribute("cart");

        if (cart == null) {
            cart = new ArrayList();
            session.setAttribute("cart", cart);
        }
```

```
        request.setAttribute("cart", cart);

        return mapping.findForward("success");
    }
}
```

5.6. Текст JSP-страниц

Листинг 7. list.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="html" uri="/tags/struts-html" %>
<%@ taglib prefix="bean" uri="/tags/struts-bean" %>

<html>
<head>
<title>Добро пожаловать в магазин.</title>
</head>

<body>
<h1>Список товаров</h1>

<html:errors />

<table>
<thead>
<tr><c:forEach var="column"
        items="Наименование товара,Цена,В наличии на складе,Заказать">
    <th><c:out value="${column}"/></th>
    </c:forEach>
</tr>
</thead>

<tbody>
<c:forEach var="item" items="${goods}">
<tr><td><c:out value="${item.name}"/></td>
    <td><c:out value="${item.price}"/></td>
    <td><c:out value="${item.quantity}"/></td>
    <td><html:form action="orderItem">
    <input type="hidden" name="id" value="<c:out value='${item.id}'/>"/>
    <html:text property="quantity"/>
    <html:submit />
    </html:form></td>
</tr>
</c:forEach>
</tbody>
</table>

<p><html:link page="/showCart.do">Перейти к корзине</html:link>
</p>
</body>
</html>
```

Листинг 8. cart.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="html" uri="/tags/struts-html" %>
<%@ taglib prefix="bean" uri="/tags/struts-bean" %>

<html>
<head>
<title>Корзина</title>
</head>
<body>
<h1>Корзина</h1>

<table>
<thead>
<tr><c:forEach var="column"
        items="Номер,Заказано">
    <th><c:out value="${column}"/></th>
    </c:forEach>
</tr>
</thead>

<tbody>
<c:forEach var="item" items="${cart}">
<tr><td><c:out value="${item.id}"/></td>
    <td><c:out value="${item.quantity}"/></td>
</tr>
</c:forEach>
</tbody>
</table>

<p><html:link page="/listItems.do">Перейти к списку товаров</html:link>
</p>
</body>
</html>
```

6. Заключение

В данной работе было проведено исследование средств разработки веб-приложений и истории их развития, а также приведен пример веб-приложения на Java, написанного с применением фреймворка Struts, реализующего паттерн MVC. Хотя рассмотренное приложение не выглядит полностью законченным, оно реализует все основные концепции, которые могут применяться для создания подобных веб-приложений.

Список литературы

[1] World Wide Web — Wikipedia, the free encyclopedia. (*http://en.wikipedia.org/wiki/Www*, 31.10.2004)

[2] Web application — Wikipedia, the free encyclopedia. (*http://en.wikipedia.org/wiki/Web_application*, 31.10.2004)

[3] Greg Murray, Mark Johnson. Designing Enterprise Applications with the J2EE(TM) Platform, Second Edition. The Web Tier. (*http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/web-tier/web-tier.html*, 2.11.2004)

[4] Neal Ford. Art of Java Web Development: Struts, Tapestry, Commons, Velocity, JUnit, Axis, Cocoon, InternetBeans, WebWork. Manning, 2004.

[5] Budi Kurniawan. Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions. New Riders Publishing, 2002.