**5 ЛАБОРАТОРНАЯ РАБОТА №5**

**ИЗУЧЕНИЕ ТЕХНОЛОГИЙ SERVLET И JSP**

**Цель работы:** Изучить Java технологии Servlet и JSP для реализации интерфейса пользователя.

**5.1 Краткие теоретические сведения**

Сервлеты – это особым образом написанные (согласно спецификации) Java-программы, которые выполняются удаленно на сервере и вызов которых осуществляется удаленно из web-броузера с помощью HTTP протокола через web-сервер.

Вскоре после появления технологии сервлетов web-разработчикам всетаки пришлось столкнуться с одним большим неудобством: для генерации HTML-страниц на лету с помощью сервлета весь HTML-код приходилось помещать в сами сервлеты. Получалось, что HTML-код страницы (presentation) смешивался с Java-кодом (logic), что затрудняло работу, как программиста, так и дизайнерам сайта. Для решения этой проблемы была придумана технология JSP – Java Server Pages. Она во многом напоминала существовавшие тогда технологии ASP (от Microsoft) и ColdFusion (от Allaire). Но сходство было чисто внешним. Так же как и в ASP и в ColdFusion в JSP вы могли вставлять Java-код прямо в HTML-код страницы. Если в ASP и ColdFusion этот код при каждом вызове страницы интерпретировался, то в JSP при первом вызове jsp- страницы этот код незримо от разработчика транслировался в сервлет и компилировался, после чего при последующих обращениях веб-сервер вызывает уже не транслятор самой jsp-страницы, а откомпилированный сервлет. Разумеется, при внесении изменений в jsp-страницу веб-серверобнаруживает, что страница изменилась и снова обновляет сервлет, соответствующий этой странице.

JSP-страница обслуживает запросы также, как это делает сервлет. Таким образом, жизненный цикл и большинство возможностей JSP-страниц (в частности, динамические аспекты) определяются технологией Java Servlet. Когда запрос отображается на JSP-страницу, он обрабатывается специальным сервлетом, который вначале проверяет, не является ли сервлет JSP-страницы старше, чем сама JSP-страница. Если это так, он транслирует JSP-страницу в класс сервлета и компилирует класс. Одним из преимуществ процесса разработки JSP-страниц по сравнению с разработкой сервлетов является то, что процесс создания JSP-страниц осуществляется автоматически.

Можно настроить процесс инициализации путем переопределения метода jspInit интерфейса JspPage. Это позволит JSP-странице считывать постоянные данные конфигурации, инициализировать ресурсы и осуществлять любую другую единовременную деятельность. Освобождают ресурсы, используя метод jspDestroy.

В виду того, что корпоративный компонент совместно используется всеми JSP-страницами, его необходимо инициализировать при запуске приложения, а не в каждой отдельной JSP- странице. Технология Java Servlet обеспечивает для этой цели события жизненного цикла приложения и классы-слушатели.

**5.2 Выполнение работы**

**package** model;

**import** java.io.Serializable;

**import** javax.persistence.\*;

/\*\*

\* The persistent class for the mobile database table.

\*

\*/

@Entity

@NamedQuery(name = "Mobile.findAll", query = "SELECT m FROM Mobile m")

**public** **class** Mobile **implements** Serializable, IJpaModel {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Id

**private** Integer id;

**private** String battary;

**private** String cpu;

**private** String display;

@Column(name = "display\_size")

**private** String displaySize;

**private** String guarantee;

**private** String img;

**private** String name;

**private** String os;

**private** Integer price;

**private** String ram;

@Column(name = "sim\_count")

**private** Integer simCount;

**private** String space;

**public** Mobile() {

}

**public** Integer getId() {

**return** **this**.id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getBattary() {

**return** **this**.battary;

}

**public** **void** setBattary(String battary) {

**this**.battary = battary;

}

**public** String getCpu() {

**return** **this**.cpu;

}

**public** **void** setCpu(String cpu) {

**this**.cpu = cpu;

}

**public** String getDisplay() {

**return** **this**.display;

}

**public** **void** setDisplay(String display) {

**this**.display = display;

}

**public** String getDisplaySize() {

**return** **this**.displaySize;

}

**public** **void** setDisplaySize(String displaySize) {

**this**.displaySize = displaySize;

}

**public** String getGuarantee() {

**return** **this**.guarantee;

}

**public** **void** setGuarantee(String guarantee) {

**this**.guarantee = guarantee;

}

**public** String getImg() {

**return** **this**.img;

}

**public** **void** setImg(String img) {

**this**.img = img;

}

**public** String getName() {

**return** **this**.name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getOs() {

**return** **this**.os;

}

**public** **void** setOs(String os) {

**this**.os = os;

}

**public** Integer getPrice() {

**return** **this**.price;

}

**public** **void** setPrice(Integer price) {

**this**.price = price;

}

**public** String getRam() {

**return** **this**.ram;

}

**public** **void** setRam(String ram) {

**this**.ram = ram;

}

**public** Integer getSimCount() {

**return** **this**.simCount;

}

**public** **void** setSimCount(Integer simCount) {

**this**.simCount = simCount;

}

**public** String getSpace() {

**return** **this**.space;

}

**public** **void** setSpace(String space) {

**this**.space = space;

}

@Override

**public** String[] getHeadrs() {

**return** **new** String[] { "id", "Name", "img", "cpu", "display", "display\_size", "ram", "space", "sim\_count",

"price", "guarantee", "os", "battary" };

}

@Override

**public** Object[] getData() {

**return** **new** String[] { String.*valueOf*(id), name, img, cpu, display, displaySize, ram, space,

String.*valueOf*(simCount), String.*valueOf*(price), guarantee, os, battary };

}

**public** **void** update(Mobile mob) {

**this**.id = mob.getId();

**this**.name = mob.getName();

**this**.img = mob.getImg();

**this**.cpu = mob.getCpu();

**this**.display = mob.getDisplay();

**this**.displaySize = mob.getDisplaySize();

**this**.ram = mob.getRam();

**this**.space = mob.getSpace();

**this**.simCount = mob.getSimCount();

**this**.price = mob.getPrice();

**this**.guarantee = mob.getGuarantee();

**this**.os = mob.getOs();

**this**.battary = mob.getBattary();

}

**public** String toJson() {

**return** "{ \"id\":" + "\"" + String.*valueOf*(**this**.id) + "\", " + "\"name\":" + "\"" + name + "\", " + "\"img\":"

+ "\"" + img + "\", " + "\"cpu\":" + "\"" + cpu + "\", " + "\"display\":" + "\"" + display + "\", "

+ "\"displaySize\":" + "\"" + displaySize + "\", " + "\"ram\":" + "\"" + ram + "\", " + "\"space\":"

+ "\"" + space + "\", " + "\"simCount\":" + "\"" + String.*valueOf*(**this**.simCount) + "\", " + "\"price\":"

+ "\"" + String.*valueOf*(**this**.price) + "\", " + "\"guarantee\":" + "\"" + guarantee + "\", " + "\"os\":"

+ "\"" + os + "\", " + "\"battary\":" + "\"" + battary + "\"}";

}

}

**package** Mobile;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.List;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** model.Mobile;

**import** ua.cn.stu.database.JpaController;

/\*\*

\* Servlet implementation class MobileServlet

\*/

@WebServlet("/MobileServlet")

**public** **class** MobileServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** MobileServlet() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

//System.out.println(request.getQueryString());

JpaController cntrl = **new** JpaController();

List <Mobile> lst = cntrl.getList();

StringBuilder strBuild = **new** StringBuilder();

strBuild.append("[");

**for** (**int** i = 1; i < lst.size(); i++){

**if**(i == lst.size() - 1) {

strBuild.append(lst.get(i).toJson());

} **else** {

strBuild.append(lst.get(i).toJson() + ", ");

}

}

strBuild.append("]");

PrintWriter out = response.getWriter();

System.***out***.println(strBuild.toString());

out.println(strBuild.toString());

}

/\*\*

\* **@see** HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

doGet(request, response);

}

}

<script>

angular.module('httpExample', []).controller('FetchController', [

'$scope',

'$http',

**function**($scope, $http) {

$scope.msg = [];

$scope.onClick = **function**() {

console.log("On click");

**var** req = {

method: 'GET',

url: '/WebMarket/MobileServlet'

};

$http(req).then(**function**(answ) {

console.log(answ);

$scope.msg = answ.data;

}, **function**(err) {

console.log(err);

});

};

}

]);

</script>

<div class=*"middle"*>

<div class=*"container"* ng-controller=*"FetchController"*

ng-init=*"onClick()"*>

<div class=*"content"*>

<div ng-repeat=*"item in msg"*>

<table>

<tr>

<td><img ng-src=*"images/{{item.img}}"* /></td>

<td><lable>{{item.name}}</label>

<td>

<td><lable>{{item.cpu}}</label>

<td>

<td><lable>{{item.display}}</label>

<td>

<td><lable>{{item.displaySize}}</label>

<td>

<td><lable>{{item.price}}</label>

<td>

</table>

</tr>

</div>

</div>

</div>

**5.3 Выводы**

В ходе лабораторной работы были изучены основы технологии JSP и Servlets. Было создано приложение, которое отображает данные из базы данных на страницу. Для конфигурации веб сервера использовался файл web.xml, который обеспечивает отображение сервлетов, а также конфигурацию контейнера сервлетов Tomcat.