# 第一章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1. 请描述HTML、CSS、DOM、JavaScript分别表示的含义。
2. 请列举出HTML常用的标记。（至少10个）
3. 请编写出一个HTML页面，令其输出“hello world!~~”，使用css将其字体设置为宋体红色。
4. 编写一个HTML页面，页面包含数字和按钮两部分，每次单击按钮可以使数字加一。

## 【答案】

1、**HTML**：HTML是英文Hyper Text Markup Language的缩写，中文译为“超文本标记语言”，其主要作用是通过HTML标记对网页中的文本、图片、声音等内容进行描述。

**CSS**：CSS 是 Cascading Style Sheet 的缩写，译作“层叠样式表单”，是用于(增强)控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。

**DOM**：DOM是Document Object Model（文档对象模型）的简称，是W3C组织推荐的处理可扩展标志语言的标准编程接口，它可以以一种独立于平台和语言的方式访问和修改一个文档的内容和结构。

**JavaScript：**JavaScript是Web中一种功能强大的脚本语言，被设计为向 HTML 页面增加交互性，常用来为网页添加各式各样的动态功能，它不需要进行编译，直接嵌入在HTML页面中，就可以把静态的页面转变成支持用户交互并响应事件的动态页面。

2、HTML常用的标记有：

1)<br />

2) <hr />

3)<p></p>

4)<font></font>

5)<img />

6)<table></table> <tr></tr> <td><td>

7)<form></form> <input />

8)<textarea></textarea>

9)<ul></ul> <li></li> <a></a>

10)<div></div>

3、HTML页面代码如下：

<html>

<head>

<title>hello world</title>

<style type="text/css">

.red {

color: red;

font-family: STSong;

}

</style>

</head>

<body>

<font class="red">hello world!~~</font>

<br />

<font>hello world!~~</font>

</body>

</html>

4、HTML页面代码如下：

<html>

<head>

<title></title>

<script type="text/javascript">

function change(){

var font = document.getElementById("font").value;

document.getElementById("font").value = font\*1+1;

}

</script>

</head>

<body>

<br />

<input type="text" id="font" value="0" readonly="readonly"><br />

<input type="button" name="btn" value="onclick" onclick="change()"/>

</body>

</html>

# 第二章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、简述HTTP1.1协议的通信过程？

2、简述POST请求和GET请求有什么不同？（至少2点）

3、请列举出Tomcat安装目录下的子目录，并对其进行简要说明？（至少列出5个）

4、请编写一个格式良好的XML文档，要求包含足球队一支，队名为Madrid，球员5人：Ronaldo、Casillas、Ramos、Modric、Benzema；篮球队一支，队名为Lakers，队员2人：Oneal，Bryant。里面要求含有注释，注释内容自定。

## 【答案】

1、HTTP1.1协议的通信过程如下：

1) 客户端与服务器端建立TCP连接。

2) 客户端可以向服务器端发送多个请求，并且在发送下个请求时，无需等待上次请求的返回结果。

3) 服务器必须按照接受客户端请求的先后顺序依次返回响应结果。

4）客户端发出关闭TCP连接的请求。

5) 服务器端关闭TCP连接。

2、POST请求和GET请求主要有如下不同：

1) POST传输数据大小无限制

2) POST比GET请求方式更安全

3、Tomcat安装目录下的子目录及其说明如下：

bin：用于存放Tomcat的可执行文件和脚本文件(扩展名为.bat的文件)，如tomcat7.exe、startup.bat；

conf：用于存放Tomcat的各种配置文件，如web.xml、server.xml；

lib：用于存放Tomcat服务器和所有Web应用程序需要访问的JAR文件；

logs：用于存放Tomcat的日志文件；

temp：用于存放Tomcat运行时产生的临时文件；

webapps：Web应用程序的主要发布目录，通常将要发布的应用程序放到这个目录下；

work：Tomcat的工作目录，JSP编译生成的Servlet源文件和字节码文件放到这个目录下。

4、XML页面代码如下：

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<球队列表>

<!-- 足球队Madrid -->

<球队 类型="足球队">

<队名>Madrid</队名>

<队员>

<姓名>Ronaldo</姓名>

<姓名>Casillas</姓名>

<姓名>Ramos</姓名>

<姓名>Modric</姓名>

<姓名>Benzema</姓名>

</队员>

</球队>

<!-- 篮球队Lakers -->

<球队 类型="篮球队">

<队名>Lakers</队名>

<队员>

<姓名>Oneal</姓名>

<姓名>Bryant</姓名>

</队员>

</球队>

</球队列表>

# 第三章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、请列举Servlet接口中的方法，并分别说明这些方法的特点及其作用。

2、简述ServletContext接口的三个主要作用。

3、编写一个servlet，实现统计网站被访问次数的功能。

4、请编写一段程序，使程序能读取该servlet的配置信息，从中获得参数名为encoding对应的参数值，并输出到页面上。

## 【答案】

1、Servlet接口中有init、service、destroy 、getServletConfig和getServletInfo这 5个方法，这些方法的特点及其作用具体如下：

1）init(ServletConfig config)方法，该方法在服务器对Servlet第一次访问时调用，负责Servlet初始化工作。在一个Servlet的生命周期中只执行一次。该方法接收一个ServletConfig类型的参数，Servlet容器可以通过这个参数向Servlet传递初始化配置信息。

2）service(ServletRequest request，ServletResponse response)方法，该方法负责响应用户的请求，当容器接收到客户端访问Servlet对象的请求时，就会调用此方法。

3）destroy()方法，该方法方法负责释放Servlet对象占用的资源。当Servlet对象被销毁时，容器会调用此方法。

4）getServletConfig()方法，该方法返回容器调用init(ServletConfig config)方法时传递给Servlet的ServletConfig对象。

5）getServletInfo()方法，该方法返回一个字符串，其中包含了关于Servlet的信息，例如，作者、版本和版权等信息。

2、ServletContext接口的三个主要作用如下：

1）获取Web应用程序的初始化参数

2）实现多个Servlet对象共享数据

3）读取Web应用下的资源文件

3、实现步骤如下：

1. 新建一个类ShowTimesServlet继承HttpServlet，重写doGet和doPost方法
2. 在doPost方法中调用doGet方法，在doGet方法中实现统计网站被访问次数的功能，用户每请求一次Servlet，使访问次数times加1
3. 获取ServletContext，通过它的功能记住上一次访问后的次数

（1）ShowTimesServlet类的编写如下：

public class ShowTimesServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

ServletContext context = getServletContext();

Integer times = (Integer) context.getAttribute("times");

if (times == null) {

times = new Integer(1);

} else {

times = new Integer(times.intValue() + 1);

}

//设置编码,不然中文会变成乱码

resp.setContentType("text/html;charset=GB2312");

//得到输出流对象

PrintWriter out=resp.getWriter();

out.println("<html><head><title>");

out.println("页面访问统计");

out.println("</title></head><body>");

out.println("当前页面被访问了");

out.println("<font color=red size=20>"+times+"</font>次");

//设置属性,将times保存到当前的上下文中

context.setAttribute("times",times);

}

protected void doPost(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

this.doGet(req, resp);

}

}

（2）web.xml配置信息如下：

<servlet>

<servlet-name>ShowTimesServlet</servlet-name>

<servlet-class>

cn.itcast.chapter03.servlet.ShowTimesServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ShowTimesServlet</servlet-name>

<url-pattern>/ShowTimesServlet</url-pattern>

</servlet-mapping>

4、实现代码如下：

（1）TestServlet类如下：

public class TestServlet extends HttpServlet {

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

PrintWriter out = response.getWriter();

// 获得ServletConfig对象

ServletConfig config = this.getServletConfig();

// 获得参数名为encoding对应的参数值

String param = config.getInitParameter("encoding");

out.println("encoding=" + param);

}

protected void doPost(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

this.doGet(request, response);

}

}

（2）web.xml文件配置如下：

<servlet>

<servlet-name>TestServlet</servlet-name>

<servlet-class>cn.itcast.chapter03.servlet.TestServlet</servlet-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>TestServlet</servlet-name>

<url-pattern>/TestServlet</url-pattern>

</servlet-mapping>

# 第四章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、简述请求转发与重定向的异同？（至少写3点）

2、请写出禁止浏览器缓存页面的核心代码。

3、请编写一个类，该类能够实现访问完app应用下的Servlet后，还能在浏览器地址栏中显示出index.jsp的路径。

4、请编写一个类，该类使用HttpServletRequest对象的getHeader("referer")方法实现下载资源防盗链的功能。

## 【答案】

1、请求转发与重定向的异同点如下：

1. 请求转发和重定向都可以实现访问一个资源时转向当前应用资源
2. 请求转发是一次请求一次响应，而重定向为两次请求两次响应

3）一般情况下应该使用请求转发，减少浏览器对服务器的访问，减轻服务器压力

4）如果需要改变浏览器的地址栏，或更改浏览器的刷新按钮的功能时需要使用重定向

2、禁止浏览器缓存页面的核心代码如下：

response.setDateHeader("Expires",0);

response.setHeader("Cache-Control","no-cache");

response.setHeader("Pragma","no-cache");

3、实现代码如下：

（1）MyServlet程序类如下：

public class MyServlet extends HttpServlet {

protected void doGet(HttpServletRequest request,

HttpServletResponse response)throws ServletException, IOException {

System.out.println("这里是MyServlet！");

response.sendRedirect(request.getContextPath() + "/index.jsp");

}

}

（2）web.xml文件配置如下：

<servlet>

<servlet-name>MyServlet</servlet-name>

<servlet-class>

cn.itcast.chapter04.servlet.MyServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MyServlet</servlet-name>

<url-pattern>/MyServlet</url-pattern>

</servlet-mapping>

4、该功能的实现步骤如下：

1）创建一个DownManagerServlet类，使其继承HttpServlet类并重写该类的doGet()方法。

2）在doGet()方法中，使用HttpServletRequest对象的getHeader("referer")方法得到字段referer的值，并判断是否可以进行资源下载。

3）如果不能下载，则进入下载页（需要在Eclipse的WebContent目录下创建一个download.html）。

（1）DownManagerServlet类的实现代码如下：

public class DownManagerServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

// 获取referer头的值

String referer = request.getHeader("referer");

// 获取访问地址

String sitePart = "http://" + request.getServerName();

if (referer != null && referer.startsWith(sitePart)) {

// 处理正在下载的请求

out.println("dealing download ...");

} else {

// 非法下载请求跳转到download.html页面

RequestDispatcher rd = request

.getRequestDispatcher("/download.html");

rd.forward(request, response);

}

}

}

（2）web.xml文件中DownManagerServlet的映射信息配置如下：

<servlet>

<servlet-name>DownManagerServlet</servlet-name>

<servlet-class>

cn.itcast.chapter04.servlet.DownManagerServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>DownManagerServlet</servlet-name>

<url-pattern>/DownManagerServlet</url-pattern>

</servlet-mapping>

# 第五章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、简述什么是会话技术？

2、简述Cookie与Session的区别?（至少写出3点）

3、请设计一个类，使用Cookie技术实现显示用户上次访问时间的功能。

要求如下：

* 1. 创建一个LastAccessServlet类，使其继承HttpServlet类并重写该类的doGet()方法。
  2. 在doGet()方法中，使用request.getCookies()得到所有cookie形成的cookie数组，并进行遍历。
  3. 如果遍历过程中找到cookie信息中存在lastAccess属性则输出，否则创建cookie对象设置值为当前时间并发送给客户端。
  4. cookie的存活时间为1小时，访问当前应用中的所有资源客户端都回送cookie信息。

4、请设计一个程序，使用Session技术实现购物车功能。

## 【答案】

1、会话过程类似于生活中的打电话过程，它指的是一个客户端（浏览器）与Web服务器之间连续发生的一系列请求和响应过程。在Servlet技术中，提供了两个用于保存会话数据的对象，分别是Cookie和Session。

2、Cookie与Session主要有如下区别：

1）Cookie和HttpSession是保存会话相关数据的技术，其中Cookie将信息存储在浏览器端，是客户端技术，Session将数据保存在服务器端，是服务器端技术

2）Cookie是基于HTTP协议中的Set-Cookie响应头和Cookie请求头进行工作的

3）默认情况下HttpSession是基于一个名称为JSESSIONID 的特殊Cookie工作的

4）浏览器对Cookie具有严格的限制，一个网站能在浏览器中保存多少Cookie是有限制的

5）HttpSession默认是基于Cookie运作的。

3、该功能的实现步骤如下：

* 1. 创建一个LastAccessServlet类，使其继承HttpServlet类并重写该类的doGet()方法。
  2. 在doGet()方法中，使用request.getCookies()得到所有cookie形成的cookie数组，并进行遍历。
  3. 如果遍历过程中找到cookie信息中存在lastAccess属性则输出，否则创建cookie对象设置值为当前时间并发送给客户端。
  4. cookie的存活时间为1小时，访问当前应用中的所有资源客户端都回送cookie信息。

（1）LastAccessServlet类的实现代码如下：

public class LastAccessServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

/\*

\* 设定一个 cookie 的name : lastAccessTime 读取客户端发送cookie

\* 获得用户上次的访问时间显示

\*/

String lastAccessTime = null;

// 获取所有的cookie，并将这些cookie存放在数组中

Cookie[] cookies = request.getCookies();

for (int i = 0; cookies != null && i < cookies.length; i++) {

if ("lastAccess".equals(cookies[i].getName())) {

// 如果cookie的名称为lastAccess,则获取该cookie的值

lastAccessTime = cookies[i].getValue();

break;

}

}

// 判断是否存在名称为lastAccess的cookie

if (lastAccessTime == null) {

response.getWriter().print("您是首次访问本站！！！");

} else {

response.getWriter().print("您上次的访问时间" + lastAccessTime);

}

// 创建cookie,将当前时间作为cookie的值发送给客户端

Cookie cookie = new Cookie("lastAccess", new Date().toLocaleString());

cookie.setMaxAge(60 \* 60);// 保存1小时

// 访问chapter05下资源时回送cookie

cookie.setPath("/chapter05");

// 发送 cookie

response.addCookie(cookie);

}

}

（2）web.xml文件配置如下：

<servlet>

<servlet-name>LastAccessServlet</servlet-name>

<servlet-class>

cn.itcast.chapter05.servlet.LastAccessServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>LastAccessServlet</servlet-name>

<url-pattern>/LastAccessServlet</url-pattern>

</servlet-mapping>

4、该功能的实现代码如下：

（1）书籍Book类的创建如下：

public class Book implements Serializable {

private static final long serialVersionUID = 1L;

private String id;

private String name;

public Book() {

}

public Book(String id, String name) {

this.id = id;

this.name = name;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

（2）书籍数据库模拟类BookDB的代码如下：

private static Map<String, Book> books = new LinkedHashMap<String, Book>();

static {

books.put("1", new Book("1", "javaweb开发"));

books.put("2", new Book("2", "jdbc开发"));

books.put("3", new Book("3", "java基础"));

books.put("4", new Book("4", "struts开发"));

books.put("5", new Book("5", "spring开发"));

}

// 获得所有的图书

public static Collection<Book> getAll() {

return books.values();

}

// 根据指定的id获得图书

public static Book getBook(String id) {

return books.get(id);

}

}

（3）显示可购买图书的列表类ListBookServlet的代码如下：

public class ListBookServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

resp.setContentType("text/html;charset=utf-8");

PrintWriter out = resp.getWriter();

Collection<Book> books = BookDB.getAll();

out.write("本站提供的图书有：<br>");

for (Book book : books) {

String url = "/chapter05/PurchaseServlet?id=" + book.getId();

out.write(book.getName() + "<a href='" + url

+ "'>点击购买</a><br>");

}

}

}

（4）购物功能类PurchaseServlet的代码如下：

public class PurchaseServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

// 获得用户购买的商品

String id = req.getParameter("id");

if (id == null) {

// 如果id为null，重定向到ListBookServlet页面

String url = "/chapter05/ListBookServlet";

resp.sendRedirect(url);

return;

}

Book book = BookDB.getBook(id);

// 创建或者获得用户的Session对象

HttpSession session = req.getSession();

// 从Session对象中获得用户的购物车

List<Book> cart = (List) session.getAttribute("cart");

if (cart == null) {

// 首次购买，为用户创建一个购物车(List集合模拟购物车)

cart = new ArrayList<Book>();

// 将购物城存入Session对象

session.setAttribute("cart", cart);

}

// 将商品放入购物车

cart.add(book);

// 创建Cookie存放Session的标识号

Cookie cookie = new Cookie("JSESSIONID", session.getId());

cookie.setMaxAge(60 \* 30);

cookie.setPath("/chapter05");

resp.addCookie(cookie);

// 重定向到购物车页面

String url = "/chapter05/CartServlet";

resp.sendRedirect(url);

}

}

（5）展示所购图书类CartServlet的代码如下：

public class CartServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

resp.setContentType("text/html;charset=utf-8");

PrintWriter out = resp.getWriter();

// 变量cart引用用户的购物车

List<Book> cart = null;

// 变量pruFlag标记用户是否买过商品

boolean purFlag = true;

// 获得用户的session

HttpSession session = req.getSession(false);

// 如果session为null，purFlag置为false

if (session == null) {

purFlag = false;

} else {

// 获得用户购物车

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

// 如果用的购物车为null，purFlag置为false

if (cart == null) {

purFlag = false;

}

}

/\*

\* 如果purFlag为false，表明用户没有购买图书 重定向到ListServlet页面

\*/

if (!purFlag) {

out.write("对不起！您还没有购买任何商品！<br>");

} else {

// 否则显示用户购买图书的信息

out.write("您购买的图书有：<br>");

double price = 0;

for (Book book : cart) {

out.write(book.getName() + "<br>");

}

}

}

}

（6）web.xml文件配置如下：

<servlet>

<display-name>ListBookServlet</display-name>

<servlet-name>ListBookServlet</servlet-name>

<servlet-class>

cn.itcast.chapter05.session.example01.ListBookServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ListBookServlet</servlet-name>

<url-pattern>/ListBookServlet</url-pattern>

</servlet-mapping>

<servlet>

<display-name>PurchaseServlet</display-name>

<servlet-name>PurchaseServlet</servlet-name>

<servlet-class>

cn.itcast.chapter05.session.example01.PurchaseServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>PurchaseServlet</servlet-name>

<url-pattern>/PurchaseServlet</url-pattern>

</servlet-mapping>

<servlet>

<display-name>CartServlet</display-name>

<servlet-name>CartServlet</servlet-name>

<servlet-class>

cn.itcast.chapter05.session.example01.CartServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>CartServlet</servlet-name>

<url-pattern>/CartServlet</url-pattern>

</servlet-mapping>

# 第六章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、请简述JSP的运行原理。

1. 请简述JSP中的9个隐式对象。
2. 请使用include标签编写两个JSP页面，要求：访问b.jsp页面时先输出b.jsp页面的内容，等待5秒，再输出a.jsp页面中的内容。
3. 请在web.xml文件中使用<error-page>元素为整个Web应用程序设置错误处理页面。已知：处理状态码为404的页面为404.jsp, 处理状态码为500的页面为500.jsp。

## 【答案】

1、JSP的运行原理如下：

1）客户端发出请求，请求访问JSP文件。

2）JSP容器先将JSP文件转换成一个Java源文件（Java Servlet源程序），在转换过程中，如果发现JSP文件中存在任何语法错误，则中断转换过程，并向服务端和客户端返回出错信息。

3）如果转换成功，则JSP容器将生成的Java源文件编译成相应的字节码文件\*.class。该class文件就是一个Servlet，Servlet容器会像处理其他Servlet一样来处理它。

4）由Servlet容器加载转换后的Servlet类（.class文件）创建一个该Servlet（JSP页面的转换结果）实例，并执行Servlet的jspInit()方法。jspInit()方法在Servlet的整个生命周期中只会执行一次。

5）执行jspService()方法来处理客户端的请求。对于每一个请求，JSP容器都会创建一个新的线程来处理它。如果多个客户端同时请求该JSP文件，则JSP容器也会创建多个线程，使得每一个客户端请求都对应一个线程。JSP运行过程中采用的这种多线程的执行方式可以极大地降低对系统资源的需求，提高系统的并发量并缩短响应时间。需要注意的是，由于第（4）步生成的Servlet是常驻内存的，所以响应速度非常快。

6）如果JSP文件被修改了，则服务器将根据设置决定是否对该文件重新编译，如果需要重新编译，则使用重新编译后的结果取代内存中常驻的Servlet，并继续上述处理过程。

7）虽然JSP效率很高，但在第一次调用的时候往往由于需要转换和编译，所以会产生一些轻微的延迟。此外，由于系统资源不足等原因，JSP容器可能会以某种不确定的方式将Servlet从内存中移除，发生这种情况时首先会调用jspDestroy()方法，然后Servlet实例会被加入“垃圾收集”处理。

8）当请求处理完成后，响应对象由JSP容器接收，并将HTML格式的响应信息发送回客户端。

2、JSP2.0规范中提供了9个隐式（内置）对象，它们是JSP默认创建的，可以直接在JSP页面中使用。这9个隐式对象的名称和作用如下：

1）out：主要用于页面输出；

2）request：用于得到用户请求信息；

3）response：表示服务器向客户端的回应信息；

4）config：包含服务器的配置信息，可以使用该对象获得Servlet初始化参数；

5）session：主要用来保存用户信息；

6）application：包含所有用户的共享信息；

7）page：指当前页面转换后的Servlet类的实例；

8）pageContext：表示JSP页面容器，它提供了对JSP页面内所有的对象及名字空间的访问；

9）exception:：表示JSP页面所发生的异常，在错误页中才起作用。

3、该功能的实现代码如下：

（1）a.jsp文件如下：

<%@ page contentType="text/html;charset=UTF-8"%>

<%Thread.sleep(5000);%>

这个是a.jsp页面<br />

（2）b. jsp文件如下：

<%@ page contentType="text/html;charset=UTF-8"%>

这个是b.jsp页面<br />

<jsp:include page="a.jsp" flush="true" />

4、web.xml文件代码如下：

<error-page>

<error-code>404</error-code>

<location>/404.jsp</location>

</error-page>

<error-page>

<error-code>500</error-code>

<location>/500.jsp</location>

</error-page>

# 第七章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1. 简述Javabean规范都有哪些?（至少写出2点）
2. 简述定义EL标识符的规范。
3. 请按要求编写一个c\_ choose.jsp页面，使用<c:choose>、<c:when>和<c:otherwise>这三个标签完成多个条件选择的程序。

要求：

1）访问c\_choose.jsp页面时不传递参数时，浏览器中显示的信息为unknown user

2）访问c\_choose.jsp页面时传递一个参数username=itcast时，浏览器中显示的信息为itcast is a manager

3）访问c\_choose.jsp页面时传递username参数为itcast以外的值时，浏览器中显示的信息为you are a manager

4、使用EL表达式和JSTL实现将文本框中的内容输出到页面的功能，在此页面的文本框中输入内容，单击页面提交按钮之后，输入框以及按钮消失，文本框中输入的内容显示到页面。

5、Person类为一个标准的JavaBean类，请设计一个程序，要求使用BeanUtils工具为Person对象赋值。

1）直接生成Person对象。

2）使用BeanUtils工具为name属性赋值”Tom”,age赋值为31。

3）使用BeanUtils工具取出属性值，并在控制台输出。

## 【答案】

1. Javabean规范如下：

1）它必须具有一个公共的、无参的构造方法，这个方法可以是编译器自动产生的缺省构造方法。

2）它提供公共的setter方法和getter方法让外部程序设置和获取JavaBean的属性。

3）它是一个公有类。

4）通常需要实现java.io.Serializable，用于序列化。

1. EL标识符的规范如下：
2. 可以由任意顺序的大小写字母、数字和下划线组成
3. 不能以数字开头
4. 不能是EL中的保留字，如and、or、gt；
5. 不能是EL隐式对象，如pageContext；
6. 不能包含单引号（'）、双引号（"）、减号（-）和正斜线等特殊字符

3、c\_ choose.jsp页面代码如下：

<%@ page language="java" contentType="text/html;

charset=utf-8" pageEncoding="utf-8" import="java.util.\*"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core"

prefix="c"%>

<html>

<head></head>

<body>

<c:choose>

<c:when test="${empty param.username}">

unKnown user.

</c:when>

<c:when test="${param.username=='itcast' }">

${ param.username} is manager.

</c:when>

<c:otherwise>

you are a employee.

</c:otherwise>

</c:choose>

</body>

</html>

4、页面代码如下：

<%@ page language="java" contentType="text/html; charset=utf-8"

pageEncoding="utf-8"%>

<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html>

<head>

<title>welcome</title>

</head>

<body>

<form action="${pageContext.request.contextPath}/welcome.jsp">

<c:if test="${empty param.num1}">

name：<input type="text" name="num1">

<input type="submit" value="submit" /><br />

</c:if>

<c:if test="${not empty param.num1}">

${param.num1} welcome！！<br />

</c:if>

</form>

</body>

</html>

5、该功能的实现步骤如下：

1）直接生成Person对象。

2）使用BeanUtils工具为name属性赋值”Tom”,age赋值为31。

3）使用BeanUtils工具取出属性值，并在控制台输出。

（1）Person类如下：

public class Person {

private String name;

private int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

（2）BeanUtilsDemo01类如下：

public class BeanUtilsDemo01 {

public static void main(String[] args) throws Exception {

Person p = new Person();

// 使用BeanUtils为属性赋值

BeanUtils.setProperty(p, "name", "Tom");

BeanUtils.setProperty(p, "age", 31);

// 使用BeanUtils获取属性值

String name = BeanUtils.getProperty(p, "name");

String age = BeanUtils.getProperty(p, "age");

System.out.println("名字是" + name + ",今年" + age + "岁");

}

}

# 第八章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、简述过滤器的作用。

2、简述Servlet事件监听器的作用。（写出三点）

3、已知有一个名为MyServlet的程序，程序可向浏览器输出“Hello MyServlet”。请编写一个用于拦截MyServlet程序的MyFilter拦截器。

要求如下：

1）编写名为MyFilter的过滤器，过滤器可向浏览器输出“Hello MyFilter”。

2）编写web.xml文件，配置MyFilter对MyServlet的拦截。

4、如何对ServletContext、HttpSession和ServletRequest这三个域对象属性的变更进行监听？请用代码展示。

## 【答案】

1、Filter被称作过滤器或者拦截器，其基本功能就是对Servlet容器调用Servlet的过程进行拦截，从而在Servlet进行响应处理前后实现一些特殊功能。

2、Servlet事件监听器的作用有如下几点：

1）监听Web应用程序中ServletContext、HttpSession和ServletRequest等域对象的创建和销毁过程。

2）监听ServletContext、HttpSession和ServletRequest等域对象属性的修改。

3）感知绑定到HttpSession域中某个对象的状态。

3、该功能的实现步骤如下：

（1）创建MyServlet程序类。

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyServlet extends HttpServlet {

public void doGet(HttpServletRequest request,

HttpServletResponse response)throws ServletException, IOException {

response.getWriter().write("Hello MyServlet ");

}

public void doPost(HttpServletRequest request,

HttpServletResponse response)throws ServletException, IOException {

doGet(request, response);

}

}

（2）创建过滤器MyFilter类。

import java.io.\*;

import javax.servlet.Filter;

import javax.servlet.\*;

public class MyFilter implements Filter {

public void destroy() {

// 过滤器对象在销毁时自动调用，释放资源

}

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

// 用于拦截用户的请求，如果和当前过滤器的拦截路径匹配，该方法会被调用

PrintWriter out=response.getWriter();

out.write("Hello MyFilter");

}

public void init(FilterConfig fConfig) throws ServletException {

// 过滤器对象在初始化时调用，可以配置一些初始化参数

}

}

（3）编写web.xml文件，设置对MyServlet的拦截

<servlet>

<servlet-name>MyServlet</servlet-name>

<servlet-class>cn.itcast.chapter08.servlet.MyServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MyServlet</servlet-name>

<url-pattern>/MyServlet</url-pattern>

</servlet-mapping>

<filter>

<display-name>MyFilter</display-name>

<filter-name>MyFilter</filter-name>

<filter-class>cn.itcast.chapter08.filter.MyFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>MyFilter</filter-name>

<url-pattern>/MyServlet</url-pattern>

</filter-mapping>

（4）启动服务器后，访问MyServlet，即可查看拦截效果。

4、该功能的实现步骤如下：

（1）编写一个testattribute.jsp页面，以观察各个域对象属性事件监听器的作用。

<%@ page language="java" contentType="text/html; charset=utf-8"

pageEncoding="utf-8"%>

<html>

<head>

<title>Insert title here</title>

</head>

<body>

<h3>这是一个测试对象属性信息监听器的页面</h3>

<%

getServletContext().setAttribute("username", "itcast");

getServletContext().setAttribute("username", "itheima");

getServletContext().removeAttribute("username");

session.setAttribute("username", "itcast");

session.setAttribute("username", "itheima");

session.removeAttribute("username");

request.setAttribute("username", "itcast");

request.setAttribute("username", "itheima");

request.removeAttribute("username");

%>

</body>

</html>

（2）编写一个MyAttributeListener类，该类实现了ServletContextAttributeListener、HttpSessionAttributeListener和ServletRequestAttributeListener接口，并实现该接口中的所有方法。

public class MyAttributeListener implements ServletContextAttributeListener,

HttpSessionAttributeListener, ServletRequestAttributeListener {

public void attributeAdded(ServletContextAttributeEvent sae) {

String name = sae.getName();

System.out.println("ServletContext添加属性：" + name + "="

+ sae.getServletContext().getAttribute(name));

}

public void attributeRemoved(ServletContextAttributeEvent sae) {

String name = sae.getName();

System.out.println("ServletContext移除属性: " + name);

}

public void attributeReplaced(ServletContextAttributeEvent sae) {

String name = sae.getName();

System.out.println("ServletContext替换属性：" + name + "="

+ sae.getServletContext().getAttribute(name));

}

public void attributeAdded(HttpSessionBindingEvent hbe) {

String name = hbe.getName();

System.out.println("HttpSession添加属性：" + name + "="

+ hbe.getSession().getAttribute(name));

}

public void attributeRemoved(HttpSessionBindingEvent hbe) {

String name = hbe.getName();

System.out.println("HttpSession移除属性: " + name);

}

public void attributeReplaced(HttpSessionBindingEvent hbe) {

String name = hbe.getName();

System.out.println("HttpSession替换属性：" + name + "="

+ hbe.getSession().getAttribute(name));

}

public void attributeAdded(ServletRequestAttributeEvent sra) {

String name = sra.getName();

System.out.println("ServletRequest添加属性：" + name + "="

+ sra.getServletRequest().getAttribute(name));

}

public void attributeRemoved(ServletRequestAttributeEvent sra) {

String name = sra.getName();

System.out.println("ServletRequest移除属性: " + name);

}

public void attributeReplaced(ServletRequestAttributeEvent sra) {

String name = sra.getName();

System.out.println("ServletRequest替换属性：" + name + "="

+ sra.getServletRequest().getAttribute(name));

}

}

（3）在web.xml文件中，部署 MyAttributeListener事件监听器。

<listener>

<listener-class>

cn.itcast.chapter08.listener.MyAttributeListener

</listener-class>

</listener>

（4）访问testattribute.jsp页面，查看控制台窗口的显示结果。

# 第九章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、请简述什么是JDBC。

2、简述JDBC的实现步骤。

3、请编写一个用于读取数据库中users表信息的JDBC程序，要求分别获取字段id、name、password和email字段的值。

4、请按照以下要求设计实现PreparedStatement对象的相关批处理操作。

要求如下：

1）指定所要执行的SQL语句如下：

String sql = "INSERT INTO users(name,password) VALUES(?,?)";

2) 编写JDBCUtils工具类，类中要包含获取连接和释放资源的方法。

3）编写Example02类，要求在类中使用JDBCUtils工具类获取连接和释放资源，并使用PreparedStatement对象批量添加5条记录。

## 【答案】

1、JDBC的全称是Java数据库连接（Java Database Connectivity），它是一套用于执行SQL语句的Java API。应用程序可通过这套API连接到关系数据库，并使用SQL语句来完成对数据库中数据的查询、更新和删除等处理。

2、JDBC的实现步骤如下：

1）加载并注册数据库驱动

2）通过DriverManager获取数据库连接

3）通过Connection对象获取Statement对象

4）使用Statement执行SQL语句

5）操作ResultSet结果集

6）关闭连接，释放资源

3、该程序代码如下所示：

public class Example01 {

public static void main(String[] args) throws SQLException {

// 1. 注册数据库的驱动

DriverManager.registerDriver(new com.mysql.jdbc.Driver());

// 2.通过DriverManager获取数据库连接

String url = "jdbc:mysql://localhost:3306/jdbc";

String username = "root";

String password = "itcast";

Connection conn = DriverManager.getConnection(url, username, password);

// 3.通过Connection对象获取Statement对象

Statement stmt = conn.createStatement();

// 4.使用Statement执行SQL语句。

String sql = "select \* from users";

ResultSet rs = stmt.executeQuery(sql);

// 5. 操作ResultSet结果集

System.out.println("id | name | password | email |");

while (rs.next()) {

int id = rs.getInt("id"); // 通过列名获取指定字段的值

String name = rs.getString("name");

String psw = rs.getString("password");

String email = rs.getString("email");

System.out.println(id + " | " + name + " | " + psw + " | " + email);

}

// 6.回收数据库资源

rs.close();

stmt.close();

conn.close();

}

}

4、该功能的实现步骤如下：

（1）JDBCUtils工具类的代码如下所示：

import java.sql.\*;

public class JDBCUtils {

// 加载驱动，并建立数据库连接

public static Connection getConnection() throws SQLException,

ClassNotFoundException {

Class.forName("com.mysql.jdbc.Driver");

String url = "jdbc:mysql://localhost:3306/jdbc";

String username = "root";

String password = "itcast";

Connection conn = DriverManager.getConnection(url, username,

password);

return conn;

}

// 关闭数据库连接，释放资源

public static void release(Statement stmt, Connection conn) {

if (stmt != null) {

try {

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

stmt = null;

}

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

conn = null;

}

}

public static void release(ResultSet rs, Statement stmt,

Connection conn){

if (rs != null) {

try {

rs.close();

} catch (SQLException e) {

e.printStackTrace();

}

rs = null;

}

release(stmt, conn);

}

}

（2）Example02的代码如下所示：

public class Example02{

public static void main(String[] args) {

Connection conn = null;

PreparedStatement preStmt = null;

try {

// 加载并注册数据库驱动

conn=JDBCUtils.getConnection();

String sql = "INSERT INTO users(name,password)VALUES(?,?)";

preStmt = conn.prepareStatement(sql);

for (int i = 0; i < 5; i++) {

preStmt.setString(1, "name" + i);

preStmt.setString(2, "password" + i);

preStmt.addBatch();

}

preStmt.executeBatch();

} catch (Exception e) {

e.printStackTrace();

} finally { // 释放资源

JDBCUtils.release(null, preStmt, conn);

}

}

}

# 第十章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1. 请思考数据库连接池的工作机制是什么？

2、简述DriverManager和DataSource中的getConnection()方法的区别。

3、请按照以下要求设计一个程序，用于将表中的第一条记录相关数据封装到对象数组中。

要求如下：

1）首先向user表中插入一条记录：INSERT INTO user(id,name,password) VALUES(1,'zhangsan','123456');

2）已知存在BaseDao 类，且类中存在查询的query(String sql, ResultSetHandler<?> rsh, Object... params)方法。

3）编写ResultSetTest类，使用ResultSetHandler的相关实现类将查询结果封装到对象数组，并在控制台输出结果。

4、请按照以下要求设计一个程序，用于将表中的所有数据封装到对象数组中。

要求如下：

1. 已知表中的相关记录

mysql> SELECT \* FROM user;

+----+----------+----------+

| id | name | password |

+----+----------+----------+

| 1 | zhangsan | 123456 |

| 2 | lisi | 123456 |

| 3 | wangwu | 123456 |

+----+----------+----------+

2）已知BaseDao 类中存在查询的query(String sql, ResultSetHandler<?> rsh, Object... params)方法。

3）编写ResultSetTest2类，使用ResultSetHandler的相关实现类将表中的所有记录封装到对象数组，并在控制台输出结果。

## 【答案】

1、[数据库](http://baike.baidu.com/view/1088.htm)连接池在初始化时将创建一定数量的数据库连接放到连接池中，当应用程序访问数据库时并不是直接创建Connection，而是向连接池“申请”一个Connection。如果连接池中有空闲的Connection，则将其返回，否则创建新的Connection。使用完毕后，连接池会将该Connection回收，并交付其他的线程使用，以减少创建和断开数据库连接的次数，提高数据库的访问效率。

2、DriverManager和DataSource中的getConnection()方法的区别如下：

1）DriverManager每次调用getConnection方法都会初始化一个新的连接，而DataSource的getConnection只是从池中取出一个已存在的连接

2）DriverManager的close()是释放Connection，而DataSource的close()只会把Connection归还给连接池。

3、ResultSetTest类的实现代码如下：

public class ResultSetTest {

public static void testArrayHandler() throws SQLException {

BaseDao basedao = new BaseDao();

String sql = "select \* from user where id=?";

Object[] arr = (Object[]) basedao.query(sql, new ArrayHandler(),

new Object[] { 1 });

for (int i = 0; i < arr.length; i++) {

System.out.print(arr[i] + ", ");

}

}

public static void main(String[] args) throws SQLException {

testArrayHandler();

}

}

4、ResultSetTest2类的实现代码如下：

public class ResultSetTest2 {

public static void testArrayListHandler() throws SQLException {

BaseDao basedao = new BaseDao();

String sql = "select \* from user";

List list = (List) basedao.query(sql, new ArrayListHandler());

Object[] arr = (Object[]) list.get(0);

Object[] arr1 = (Object[]) list.get(1);

Object[] arr2 = (Object[]) list.get(2);

for (int i = 0; i < arr.length; i++) {

System.out.print(arr[i] + ", ");

}

for (int i = 0; i < arr1.length; i++) {

System.out.print(arr1[i] + ", ");

}

for (int i = 0; i < arr2.length; i++) {

System.out.print(arr2[i] + ", ");

}

}

public static void main(String[] args) throws SQLException {

testArrayListHandler();

}

}

# 第十一章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、简述什么是MVC设计模式。

2、简述MVC设计模式中模型(Model)模块的作用。

## 【答案】

1、MVC设计模式是编程语言的一种软件设计模式，提供了一种按功能对软件进行模块划分的方法。MVC设计模式将软件程序分为三个核心模块：模型(Model)、视图(View)和控制器(Controller)。

2、MVC设计模式中模型(Model)模块的作用如下：

1. 管理应用程序的业务数据。
2. 定义访问控制和修改这些数据的业务规则。
3. 当模型的状态发生改变时，它会通知视图发生改变，并为视图提供查询模型状态的方法。

# 第十二章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、请编写一个用于实现文件下载的程序，并且保证下载文件的文件名不能出现中文乱码问题。

2、请按照以下要求设计一个实现文件上传的类UploadServlet。

要求如下：

1. 已知form.html文件中form表单内定义了一个名为name的文本框及名为myfile的文件域，具备文件上传的前提条件。

2）在doPost()方法中，写出文件上传的相关代码。

3）上传的文件保存在当前应用程序的upload文件夹下。

3、 实现文件上传的表单页面都需要哪些配置?

4、简述文件下载的实现原理。

## 【答案】

1、该功能的实现步骤如下：

（1）创建下载页面download.jsp。

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>文件下载</title>

</head>

<body>

<a href="${pagContext.request.contextPath}/chapter06/DownloadServlet"}>

文件下载

</a>

<br />

</body>

</html>

（2）编写DownloadServlet类，该类主要用于设置所要下载的文件以及文件在浏览器中的打开方式，并使用encode(String s, String enc)方法，将URL中的字符串以指定的编码形式输出，防止文件名出现中文乱码问题。

import java.io.\*;

import java.net.URLEncoder;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class DownloadServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

// 获得绝对路径创建文件对象

String path=getServletContext().getRealPath("/download/人物.jpg");

File file=new File(path);

// 通知浏览器以下载的方式打开文件

response.addHeader("Content-Type", "application/octet-stream");

response.addHeader("Content-Disposition","attachment;filename="

+URLEncoder.encode(file.getName(),"utf-8"));

// 通过文件对象获取文件相关的输入流

InputStream in=new FileInputStream(file);

// 获取response对象的输出流

OutputStream out = response.getOutputStream();

byte [] buffer=new byte[1024];

int len;

while((len=in.read(buffer))!=-1){

out.write(buffer,0,len);

}

}

public void doPost(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

doGet(req, resp);

}

}

（3）使用IE浏览器访问该download.jsp，单击页面中的文件下载链接后，即可保存下载该文件。

2、UploadServlet代码如下：

public class UploadServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

try {

// 创建工厂

DiskFileItemFactory factory = new DiskFileItemFactory();

factory.setRepository(new File("e:\\Target"));

// 创建 fileupload 组件

ServletFileUpload fileupload = new ServletFileUpload(factory);

fileupload.setHeaderEncoding("utf-8");

// 解析 request

List<FileItem> fileitems = fileupload.parseRequest(request);

PrintWriter writer = response.getWriter();

// 遍历集合

for (FileItem fileitem : fileitems) {

// 判断是否为普通字段

if (fileitem.isFormField()) {

// 获得字段名和字段值

String name = fileitem.getFieldName();

String value = fileitem.getString("utf-8");

} else {

// 上传的文件路径

String filename = fileitem.getName();

writer.print("文件来源：" + filename + "<br>");

// 截取出文件名

filename = filename

.substring(filename.lastIndexOf("\\") + 1);

writer.print("成功上传的文件：" + filename + "<br>");

// 文件名需要唯一

filename = UUID.randomUUID().toString() + "\_" + filename;

// 在服务器创建同名文件

String webPath = "/upload/" + filename;

String path = getServletContext().getRealPath(webPath);

// 创建文件

File file = new File(path);

file.getParentFile().mkdirs();

file.createNewFile();

// 获得上传文件流

InputStream in = fileitem.getInputStream();

// 获得写入文件流

OutputStream out = new FileOutputStream(file);

// 流的对拷

byte[] buffer = new byte[1024];

int len;

while ((len = in.read(buffer)) > 0)

out.write(buffer, 0, len);

// 关流

in.close();

out.close();

// 删除临时文件

fileitem.delete();

}

}

} catch (Exception e) {

throw new RuntimeException(e);

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

3、实现文件上传的表单页面需要配置如下：

1）首先要创建一个用于提交上传文件的表单页面。

2）在页面中，需要使用<input type="file">标签在Web页面中添加文件上传输入项。

3）必须要设置input输入项的name属性，否则浏览器将不会发送上传文件的数据。

4）必须把将表单页面的method属性设置为post方式，enctype属性设置为“multipart/form-data”类型。

4、首先获取下载文件的地址，并根据该地址创建文件字节输入流，然后通过该流读取下载文件内容，最后将读取的内容通过输出流写到目标文件中。

# 第十三章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1. 请描述传智书城项目的开发环境。

2、请写出传智书城项目搭建所需要导入的jar包。

3、请按照以下要求写出c3p0-config.xml文件的配置。

信息如下：

1. 数据库名itcaststore。
2. 用户名root。
3. 密码itcast。
4. 请按照要求写出数据源工具类。

要求如下：

1. 工具类名为DataSourceUtils。
2. 工具类提供获取Connection对象、开启事务、提交事务、回滚事务。

## 【答案】

1. 传智书城项目的开发环境如下：
2. 操作系统：Windows XP、Windows 7或更高的Windows版本。
3. Web服务器：Tomcat7.0。
4. Java开发包：JDK1.7。
5. 数据库：MySQL5.5。
6. 开发工具：Eclipse Java EE IDE for Web Developers。
7. 浏览器：IE8.0或更高版本。

2、传智书城项目搭建所需要导入的jar包如下：

1. c3p0-0.9.1.2.jar
2. commons-beanutils-1.8.3.jar
3. commons-dbutils-1.4.jar
4. commons-fileupload-1.2.1.jar
5. commons-io-1.4.jar
6. commons-logging-1.1.1.jar
7. jstl.jar
8. mail.jar
9. mysql-connector-java-5.0.8-bin.jar
10. standard.jar

3、c3p0-config.xml文件的配置如下：

<?xml version="1.0" encoding="UTF-8"?>

<c3p0-config>

<default-config>

<property name="user">root</property>

<property name="password">itcast</property>

<property name="driverClass">com.mysql.jdbc.Driver</property>

<property name="jdbcUrl">jdbc:mysql:///itcaststore</property>

</default-config>

</c3p0-config>

4、数据源工具类DataSourceUtils实现代码如下：

package cn.itcast.itcaststore.utils;

import java.sql.Connection;

import java.sql.SQLException;

import javax.sql.DataSource;

import com.mchange.v2.c3p0.ComboPooledDataSource;

public class DataSourceUtils {

private static DataSource dataSource = new ComboPooledDataSource();

private static ThreadLocal<Connection> tl = new ThreadLocal<Connection>();

public static DataSource getDataSource() {

return dataSource;

}

/\*\*

\* 当DBUtils需要手动控制事务时，调用该方法获得一个连接

\*

\* @return

\* @throws SQLException

\*/

public static Connection getConnection() throws SQLException {

Connection con = tl.get();

if (con == null) {

con = dataSource.getConnection();

tl.set(con);

}

return con;

}

/\*\*

\* 开启事务

\*

\* @throws SQLException

\*/

public static void startTransaction() throws SQLException {

Connection con = getConnection();

if (con != null)

con.setAutoCommit(false);

}

/\*\*

\* 从ThreadLocal中释放并且关闭Connection,并结束事务

\*

\* @throws SQLException

\*/

public static void releaseAndCloseConnection() throws SQLException {

Connection con = getConnection();

if (con != null) {

con.commit();

tl.remove();

con.close();

}

}

/\*\*

\* 事务回滚

\* @throws SQLException

\*/

public static void rollback() throws SQLException {

Connection con = getConnection();

if (con != null) {

con.rollback();

}

}

}

# 第十四章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1、请简要描述注册功能的设计思路。

2、请简要描述购物车模块的设计思路。

## 【答案】

1、当新用户首次访问该网站时，需要先注册账号。在注册页面中新用户需要填写有邮箱、用户名、密码、重复密码、性别、联系电话、个人介绍以及验证码。首先对填写的注册信息进行校验，比如用户名和密码不能为空；密码和重复密码表单输入的内容必须一致；输入的邮箱地址应该是合法的等。然后查询数据库判断用户名是否存在。如果存在，则注册失败。如果不存在，则注册成功。

1. 当用户选择商品时，首先需要校验此用户是否登陆，如果未登录则提示用户登录，如果已经登陆则将商品添加至购物车，此时购物车中如已经有此商品，则商品数量加一，否则创建此商品。商品添加后可选择是继续购物、清空购物车还是进行结算，如果进行结算则让用户填写订单信息，之后生成订单，让用户完成在线支付。

# 第十五章

## 【测一测】

学习完前面的内容，下面来动手测一测吧，请思考以下问题：

1. 请简要描述一下商品管理模块的设计思路和实现流程。
2. 请简要描述一下销量榜单下载功能的设计思路和实现流程。

## 【答案】

1、 商品管理模块主要的功能包括查询商品信息、添加商品信息、编辑商品信息和删除商品信息这4个功能。查询商品信息时可通过条件查询也可以查询所用商品。查询之后可进入商品列表页面，在此页面中提供了增加、修改和删除商品信息的功能。单击添加按钮，打开商品添加页面，填写商品信息之后，单击确定按钮后，新添加的商品信息即可在列表页面中显示出来。单击商品列表中修改按钮，页面会跳转到该商品的编辑页面，修改其中显示出商品的具体信息之后，单击保存按钮，商品管理首页将显示修改后的商品信息。单击列表页面中商品后面的删除按钮，会提示是否删除，确定删除后，系统会将此条商品信息删除，商品管理首页将不再显示此信息。

2、模块主要实现的功能是下载历史销售数据，将已销售商品的信息，按照商品销量从高到低排序后导出到扩展名为“.csv”的文件中（csv文件即逗号分隔值文件格式，通常是纯文本文件）。下载销售数据时可以填写年份和月份这两个查询条件，在年份输入框和月份下拉框中分别填写相应信息，然后单击“下载”按钮，会弹出文件下载提示框，在提示框中选择文件的下载目录，单击“确定”后就可以将指定年份和月份的销售历史数据下载到文件中。