【济南中心】JavaEE就业班同步笔记第一阶段：JavaWeb之Servlet

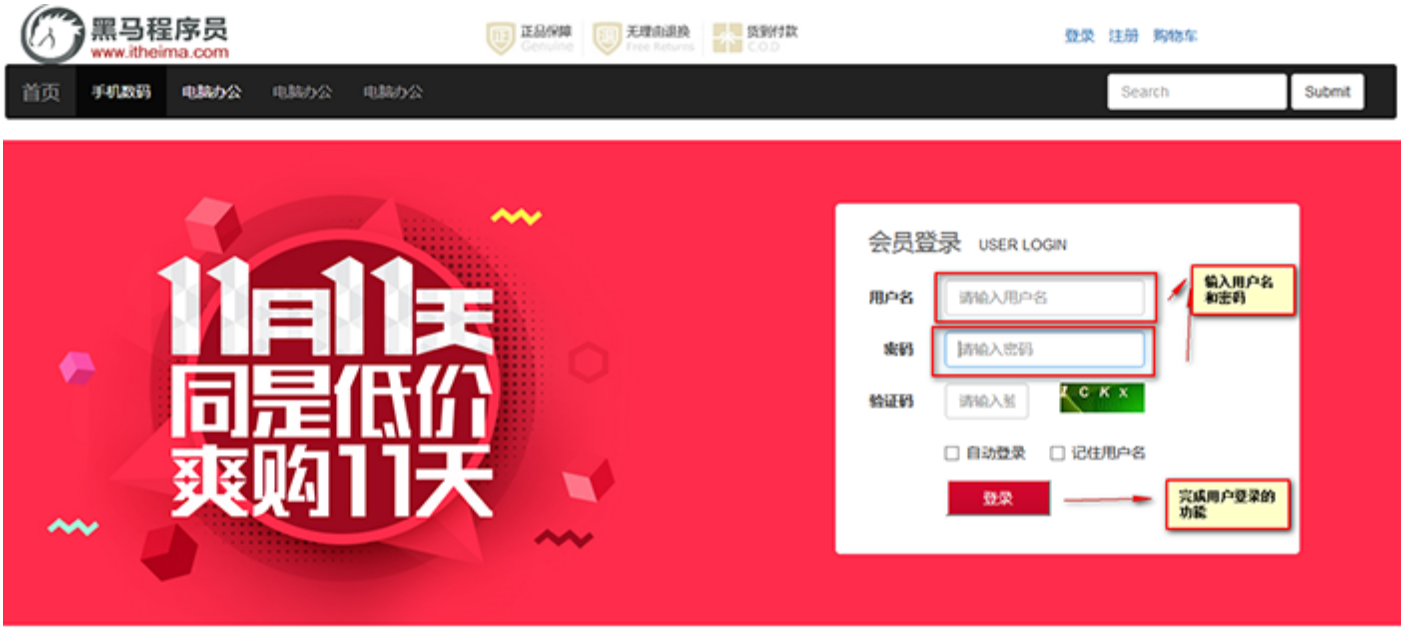
小鲁哥哥 • 2017-1-5 14:58:16

【济南中心】JavaEE就业班同步笔记第一阶段：JavaWeb之核心技术--Servlet

1 案例一：使用Servlet完成一个用户登录的案例.

1.1 需求:

在网站的首页上,登录的链接,点击登录的链接,可以跳转到登录的页面.在登录的页面中输入用户名和密码点击登录的案例.完成登录的功能.



1.2 分析:

1.2.1 技术分析:

【HTTP的协议的概述】

协议:

* 什么是协议:规定双方需要遵守的规则.

HTTP协议:

* 什么是HTTP协议:用来规定浏览器与服务器之前需要遵守的规则.

HTTP协议的作用：规范浏览器和服务器之间的数据传递.
HTTP协议的特点：
* 基于请求和响应的模型.
  * 必须先有请求后有响应.
  * 请求和响应必须成对出现.
* 默认的端口号是80.
HTTP协议的版本:
* 1.0 :每次响应后即刻关闭了连接.
* 1.1 :现在使用.不是每次响应后挂断,等待长时间以后没有请求会挂断.
【HTTP协议的演示】
抓包分析:GET方式：
* 请求部分:
[mw_shl_code=html,true]GET /day09/demo1-http/demo2.html?name=aaa&age=23 HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
X-HttpWatch-RID: 59176-10011
Referer: http://localhost:8080/day09/demo1-http/demo1.html
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: localhost:8080
DNT: 1
Connection: Keep-Alive[/mw_shl_code]

抓包分析:POST方式：
[mw_shl_code=html,true]POST /day09/demo1-http/demo2.html HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
X-HttpWatch-RID: 59176-10031
Referer: http://localhost:8080/day09/demo1-http/demo1.html
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: localhost:8080
Content-Length: 15
DNT: 1

Connection: Keep-Alive
Cache-Control: no-cache

name=bbb&age=38[/mw_shl_code]

* 响应部分:
[mw_shl_code=java,true]HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"145-1461807615933"
Last-Modified: Thu, 28 Apr 2016 01:40:15 GMT
Content-Type: text/html
Content-Length: 145
Date: Thu, 28 Apr 2016 01:43:52 GMT

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>

<h1>Demo2.html</h1>
</body>
</html>[/mw_shl_code]
【HTTP协议的详解】
请求部分
* 请求行
  * 提交方式:
    * 提交方式有很多,常用的GET和POST:
    * GET和POST的区别:
      * GET的提交的参数会显示到地址栏上,而POST不显示.
      * GET往往是有大小限制的,而POST没有大小的限制.
      * GET没有请求体,而POST有请求体.
  * 提交路径:

    * 协议版本:
* 请求头
    * 都是键值对的形式显示的.一般一个key对应一个value,也有个别的是一个key对应多个value.
    * User-Agent   :代表浏览器的类型. --- 文件下载:下载中文文件:IE使用URLEncodor进行编码,而Firefox使用Base64编码.
    * Referer   :代表的是网页的来源. --- 防盗链.
    * If-Modified-Since   :通常与响应中的头Last-Modified一起使用查找本地缓存.
* 请求体
    * 就是POST提交方式的提交的参数.

响应部分

* 响应行:
    * 协议版本
    * 状态码   :
        * 200   : 成功
        * 302   : 重定向
        * 304   : 查找本地缓存
        * 404   : 资源不存在
        * 500   : 服务器内部错误
    * 状态码描述
* 响应头：键值对,一般一个key对应一个value,也有一个key对应多个value.
    * Last-Modified   :与请求中的If-Modified-Since一起使用查找本地缓存.
    * Content-Dispostion   :文件下载的使用使用的一个头信息.
    * Location   :重定向的跳转的路径.
    * Refresh   :定时刷新/定时跳转.
* 响应体:显示浏览器的页面的内容.

【Servlet的概述】

什么是Servlet:

* 就是一个运行在WEB服务器上的小的Java程序,用来接收和响应从客户端发送过来的请求,通常使用HTTP协议.
* Servlet就是SUN公司提供的一个动态网页开发技术.

Servlet的作用：

* 用来处理从客户端浏览器发送的请求,并且可以对请求作出响应
使用Servlet：
* 编写一个类实现Servlet接口.
* 将编写的这个类配置到服务器中.
Servlet的入门:
* 编写类：
[mw_shl_code=java,true]public class ServletDemo1 implements Servlet{

```
    @Override
    /**
     * 为用户处理请求和响应的方法.
     */
    public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
{
        res.getWriter().println("Hello Servlet...");
    }
...
}[/mw_shl_code]
```

* 配置:
[mw_shl_code=xml,true]<!-- 配置Servlet -->
<servlet>
<!-- Servlet的名称 -->
<servlet-name>test1</servlet-name>
<!-- SErvlet的全路径 -->
<servlet-class>com.itheima.a_servlet.ServletDemo1</servlet-class>
</servlet>

<!-- Servlet的映射 -->
<servlet-mapping>
<!-- Servlet的名称 -->
<servlet-name>test1</servlet-name>
<!-- Servlet的访问路径 -->
<url-pattern>/ServletDemo1</url-pattern>
</servlet-mapping>[/mw_shl_code]
* 访问:
http://localhost:8080/day09/ServletDemo1

【使用ServletRequest接收参数】

* String getParameter(String name);　　---用于接收一个名称对应一个值的数据.
* String[] getParameterValues(String name);---用于接收一个名称对应多个值的数据.
* Map getParameterMap();　　---用于接收表单中的所有的数据,Map的key是表单提交的参数名称,Map的value是提交参数的值.
* Enumeration getParameterNames()　　---用于获取表单中提交的所有的参数的名称.

【Servlet的访问流程】

【Servlet的实现的关系】

Servlet　　:接口
　|
GenericServlet　　:通用的Servlet
　|
HttpServlet　　:HttpServlet

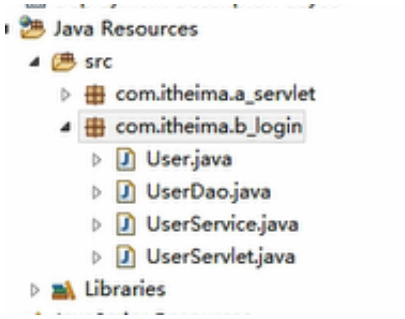
* 编写一个类继承HttpServlet，重写doGet和doPost方法.
* 配置

1.3 代码实现

1.3.1 步骤一：创建数据库和表:

```
[mw_shl_code=shell,true]create database day09;
use day09;
create table user(
	id int primary key auto_increment,
	username varchar(20),
	password varchar(20),
	nickname varchar(20)
);
insert into user values (null,'aaa','111','小凤');
insert into user values (null,'bbb','111','小童童');[/mw_shl_code]
```

1.3.2 步骤二：创建包和类:



1.3.3 步骤三：引入jar包

* mysql的数据库的驱动包

* c3p0连接池的jar包

* dbutils的包

1.3.4 引入login的页面

1.3.5 编写Servlet-->Service-->DAO

1.4 总结:

1.4.1 Servlet的生命周期:(*****)

生命周期:就是一个对象从创建到销毁的过程.

Servlet生命周期:Servlet从创建到销毁的过程.

* 何时创建:用户第一次访问Servlet创建Servlet的实例

* 何时销毁:当项目从服务器中移除的时候，或者关闭服务器的时候.

用户第一次访问Servlet的时候,服务器会创建一个Servlet的实例,那么Servlet中init方法就会执行.任何一次请求服务器都会创建一个新的线程访问Servlet中的service的方法.在service方法内部根据请求的方式的不同调用doXXX的方法.(get请求调用doGet,post请求调用doPost).当Servlet中服务器中移除掉,或者关闭服务器,Servlet的实例就会被销毁,那么destroy方法就会执行.

1.4.2 Servlet的相关的配置:

【启动时创建Servlet】

Servlet默认是在第一次访问的时候创建的.现在让Servlet在服务器启动的时候创建好.进行对Servlet的配置:

在web.xml中在<servlet></servlet>标签中配置:

* [mw_shl_code=xml,true]<load-on-startup>2</load-on-startup>[/mw_shl_code] --- 传入正整数,整数越小,被创建的优先级就越高.

【url-pattern的配置】

url-pattern配置方式共有三种:

1.完全路径匹配　　：以 / 开始　　例如:　　/ServletDemo4 , /aaa/ServletDemo5 , /aaa/bbb/ServletDemo6

2.目录匹配　　：以 / 开始 需要以 * 结束.　　例如: /* ,/aaa/* ,/aaa/bbb/*

3.扩展名匹配　　：不能以 / 开始 以 * 开始的. 例如: *.do , *.action

***** 错误的写法　　：　/*.do

有如下的配置:

```
 [mw_shl_code=xml,true]<servlet>
  <servlet-name>ServletDemo4</servlet-name>
  <servlet-class>com.itheima.a_servlet.ServletDemo4</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>ServletDemo4</servlet-name>
  <url-pattern>/ServletDemo4</url-pattern>
 </servlet-mapping>

 <servlet>
  <servlet-name>ServletDemo5</servlet-name>
  <servlet-class>com.itheima.a_servlet.ServletDemo5</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>ServletDemo5</servlet-name>
  <url-pattern>/*</url-pattern>
 </servlet-mapping>

 <servlet>
  <servlet-name>ServletDemo6</servlet-name>
  <servlet-class>com.itheima.a_servlet.ServletDemo6</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>ServletDemo6</servlet-name>
  <url-pattern>*.do</url-pattern>
 </servlet-mapping>[/mw_shl_code]
```

如果访问地址:

http://localhost:8080/day09/ServletDemo4　　:第一个

http://localhost:8080/day09/aaa.do :第二个

***** 完全路径匹配 > 目录匹配 > 扩展名匹配

1.4.3 开发中的路径的编写:

相对路径:都是需要找位置相对关系.不能以 / 开始的.

* ./ 当前路径 ../上一级目录

* 使用相对路径访问:

  * http://localhost:8080/day09/demo4-url/demo1.html

  * http://localhost:8080/day09/ServletDemo6

绝对路径:不需要找位置相对关系. 以 / 开始的.

* 绝对路径中分为客户端路径和服务器端路径:

  * 客户端路径一定要加工程名. /day09/ServletDemo6

  * 服务器端路径不需要加工程名. /ServletDemo6

2 案例二：登录成功以后5秒钟跳转到另一个页面.

2.1 需求:

在登录成功后,页面5秒钟跳转到其他的一个页面.

2.2 分析:

2.2.1 技术分析:

【使用Http协议中的Refresh头信息】

Refresh之前已经介绍可以定时页面跳转.需要使用程序设置头信息才可以.

【response中设置响应头】

* addHeader(String name,String value); --- 针对一个key对应多个value的响应头.

  * addDateHeader(String name,long date);

  * addIntHeader(String name,int value);

* setHeader(String name,String value); --- 针对一个key对应一个value的响应头.

  * setDateHeader(String name,long date);

  * setIntHeader(String name,int value);

例如:头信息: xxx:aaa

  * addHeader(“xxx”,”bbb”); -->xxx:aaa,bbb

  * setHeader(“xxx”,”bbb”); -->xxx:bbb

2.3 代码实现:

在登录成功后的代码上,定时的跳转.

[mw_shl_code=java,true]public class UserRefreshServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        try {
            // 解决向页面输出中文的乱码问题!!!
            response.setContentType("text/html;charset=UTF-8");
            // 1.接收表单提交的参数.
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            // 2.封装到实体对象中.
            User user = new User();
            user.setUsername(username);
            user.setPassword(password);

            // 3.调用业务层处理数据.
            UserService userService = new UserService();

            User existUser = userService.login(user);
            // 4.根据处理结果显示信息(页面跳转).
            if(existUser == null){
                // 登录失败
                response.getWriter().println("<h1>登录失败:用户名或密码错误！~</h1>");
            }else{
                // 登录成功
                // response.getWriter().println("Login Success...");
                response.getWriter().println("<h1>登录成功!您好:"+existUser.getNickname()+"</h1>");

                response.getWriter().println("<h3>页面将在5秒后跳转！</h3>");
                response.setHeader("Refresh", "5;url=/day09/demo5-refresh/index.html");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

```
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletE
xception, IOException {
        doGet(request, response);
    }

}[/mw_shl_code]
```

2.4 总结:

2.4.1 使用JS控制读秒的效果.

```
[mw_shl_code=javascript,true]<script type="text/javascript">
    var time = 5;
    window.onload = function(){
        setInterval('changeTime()',1000);
    }

    function changeTime(){
        time--;
        document.getElementById("s1").innerHTML = time;
    }
</script>[/mw_shl_code]
```

3 案例三：记录网站的登录成功的人数.

3.1 需求:

登录成功后,5秒后跳转到某个页面,在页面中显示您是第x位登录成功的用户.

3.2 分析:

3.2.1 技术分析:

【ServletContext对象】

***** ServletContext对象存取数据,存的数据都是有一定的作用的范围的.这种对象称为是域对象.

* 用来存取数据:

setAttribute(String name,Object object)

* 用来向ServletContext中存入数据.

* 用来从ServletContext中获取数据.

getAttribute(String name)

* 用来从ServletContext中移除数据.

removeAttribute(String name)

3.3 代码实现:

[mw_shl_code=java,true]/**

* 登录代码的Servlet

*/

```
public class UserCountServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	@Override
	public void init() throws ServletException {
		// 初始化一个变量count的值为0.
		int count = 0;
		// 将这个值存入到ServletContext中.
		this.getServletContext().setAttribute("count", count);
	}

	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx
ception, IOException {
		try {
			response.setContentType("text/html;charset=UTF-8");
			// 1.接收表单提交的参数.
			String username = request.getParameter("username");
			String password = request.getParameter("password");
			// 2.封装到实体对象中.
			User user = new User();
			user.setUsername(username);
			user.setPassword(password);

			// 3.调用业务层处理数据.
			UserService userService = new UserService();

			User existUser = userService.login(user);
			// 4.根据处理结果显示信息(页面跳转).
			if(existUser == null){
				// 登录失败
				response.getWriter().println("<h1>登录失败：用户名或密码错误!</h1>");
			}else{
```

```
                    // 登录成功

                    // 记录次数:
                    int count = (int) this.getServletContext().getAttribute("count");
                    count++;
                    this.getServletContext().setAttribute("count", count);

                    response.getWriter().println("<h1>登录成功:您好:"+existUser.getNickname()+"</h1>")
;
                    response.getWriter().println("<h3>页面将在5秒后跳转!</h3>");
                    response.setHeader("Refresh", "5;url=/day09/CountServlet");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }

        }

        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletE
xception, IOException {
            doGet(request, response);
        }

}

public class CountServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;

        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx
ception, IOException {
            // 获得Count的值。
            response.setContentType("text/html;charset=UTF-8");
            int count = (int) this.getServletContext().getAttribute("count");
            response.getWriter().println("<h1>您是第"+count+"位登录成功的用户！ </h1>");
        }

        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletE
```

```
xception, IOException {
        doGet(request, response);
    }

}
```
[/mw_shl_code]

3.4 总结:

3.4.1 ServletConfig:了解.获得Servlet的配置信息.

* String getServletName(); ---获得Servlet在web.xml中配置的name的值.
* String getInitParameter(String name); ---获得Servlet的初始化参数的.
* Enumeration getInitParameterNames(); ---获得所有Servlet的初始化参数的名称.

3.4.2 ServletContext:重要

ServletContext的作用:

* 1.用来获得全局初始化参数.
* 2.用来获得文件的MIME的类型.
* 3.作为域对象存取数据.

ServletContext是一个域对象.

* 作用范围:整个web工程.
* 创建:服务器启动的时候,tomcat服务器为每个web项目创建一个单独ServletContext对象.
* 销毁:服务器关闭的时候,或者项目从服务器中移除的时候.

* 4.用来读取web项目下的文件.

回帖