ServletConfig、ServletContext

2019年5月20日 星期一 9:14

1. ServletConfig

```
<servlet>
  <servlet-name>RegistServlet</servlet-name>
  <servlet-class>com.easymall.servlet.RegistServlet</servlet-class>
  <!--初始化信息-->
 <init-param>
    <param-name></param-name>
    <param-value></param-value>
<init-param>
</servlet>
```

如果有任何参数在servlet启动的时候需要被加载，且这个参数可能仅在web应用每次重启的时候读取，那么这个参数就可以添加杂servlet标签身上，作为一个初始化信息参数使用。

a. ServletConfig 代表当前Servlet的初始化信息的对象。

可以通过这个对象来获取配置在web.xml中某一个servlet的初始化配置信息。

b. 获取ServletConfig对象的方式：

通过init方法获取十分繁琐，所以通过当前servlet对象自带的成员变量config去操作。

this.getServletConfig();

c. 功能一：获取初始化配置信息的方式：

config.getInitParameter(String name);

config.getInitParameterNames();

d. 功能二：用来获取ServletContext对象

this.getServletConfig().getServletContext();

2. ServletContext

ServletContext就是一个代表web应用的对象。

a. 如果有多个servlet需要使用同一段初始化配置信息，那么这个信息就可以配置成一个web应用的配置信息，通过代表web应用的对象来获取。

b. 代表web应用的对象可以被web应用的所有servlet访问，也可以通过这个web应用的对象获取到web应用的配置信息。

c. 获取web应用对象方式：

i. this.getServletConfig().getServletContext();

ii. this.getServletContext();

d. 功能一：获取web应用的配置信息操作：

context.getInitParameter(String name);	获取指定名称的参数
context.getInitParameterNames()	获取全部web应用的配置信息的参数名

e. 功能二：作为域对象来使用

域对象：在一个对象身上有一个可以被看见的范围，在这个范围内利用对象身上的map实现资源共享，像这样一个对象就叫做域对象。

i. 操作域对象：

setAttribute(String name,Object obj);

getAttribute(String name);

removeAttribute(String name);
getAttributeNames();

ii. 生命周期：

web应用加载代表web应用的对象创建，web应用销毁代表web应用的对象也会被销毁。

iii. 作用范围：

当前web应用范围内。

iv. 主要功能：

当前web应用范围内实现资源的共享。

f. 功能三：获取web资源：

i. 文件的路径的获取方式

1) 相对路径 config.properties

在文件在加载的服务器启动目录去寻找指定文件名称的文件。

2) 绝对路径 /config.properties

会到项目启动的磁盘根目录下寻找指定名称的文件。

3) 书写全部路径名称 D:\software\apache-tomcat-7.0.62\webapps\day12\WEB-INF\web.xml

确实可以找到指定路径的文件，但是这种书写方式过于死板，一旦对路径上任意一个级别的名称做出修改，则这个配置文件就会加载不到。不推荐使用。

4) ServletContext

context.getRealPath(String pathname)

获取当前web应用的全路径，在当前路径下寻找指定名称的文件。如果文件位于web应用目录内部，则仍需要在参数位置书写基于web应用目录个的子级目录名称和资源文件名称才可以找到文件。

5) 类加载器获取路径

通过当前类的类加载器可以获取当前web应用src目录下的资源文件。

getClassLoader().getResource("文件名称").getPath();

a) 在使用web工程并发布的时候，这种获取路径的方式会自动调整到classes目录中去获取指定名称的文件。

AJAX

2019年5月20日 星期一 11:24

1. AJAX是什么？

   asynchronous js and xml 异步的js和xml。

   通过ajax给服务器发送请求，并让服务器根据请求做出响应实现ajax，只是这个过程是一个异步 请求的过程，做出的操作为局部刷新。

   i. 传输数据的格式：

      xml用来传输数据

      json

         "username":username

      text:我是李帅

   名称发展：ajaj -> aj

2. 异步交互和同步交互

   同步交互：

      如果一个ajax实现的是同步交互，发出请求之后，浏览器会等待响应结束之后才允许继续操作。一次请求必须要等待一次响应结束，在这期间不允许进行任何操作。

      刷新范围：整体刷新。必须等待页面中的请求加载完毕才能继续操作。

   异步交互:

      如果一个ajax实现的是异步交互，发出请求之后，浏览器仍然可以继续操作。多次请求直接互补干扰，可以连续多次发送请求，服务器会针对各个请求做出响应。

      刷新范围：局部刷新。不必等待页面完全加载完成即可以继续操作页面。

3. 使用情景：

   a. 异步交互：

      i. 搜索框中书写内容是异步交互。

      ii. 点击搜索按钮是同步交互。

4. 异步交互的特点：

   a. 优点：用户体验良好。如果用户合理使用ajax，会降低服务器的访问压力。

   b. 缺点：会无端产生很多无效的请求，增加服务器访问的压力。

5. javascript实现ajax 四步

   a. 第一步, 获取XMLHttpRequest对象

```
var xmlHttp = ajaxFunction();
function ajaxFunction(){
    var xmlHttp;
    try{
        //现代浏览器（IE7+、Firefox、Chrome、Safari 和 Opera）都有内建的
        XMLHttpRequest 对象
        xmlHttp = new XMLHttpRequest();
```

```
        }catch(e){
            try{
                //IE6.0
                xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
            }catch(e){
                try{
                    //IE5.0及更早版本
                    xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
                }catch(e){
                    alert("...");
                    throw e;
                }
            }
        }
        return xmlHttp;
    }
```

b. 第二步, 打开与服务器的连接

```
xmlHttp.open(method, url, async);
```

> method: 请求方式, 可以是GET或POST
> url: 所要访问的服务器中资源的路径 如: /Day10/servlet/AServlet
> async: 是否为异步传输, true 表示为异步传输 一般都是true false 表示同步传输

c. 第三步, 发送请求

```
xmlHttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded"); //通知服务器发送的数据是请求参数
xmlHttp.send("xxxx"); //注意, 如果不给参数可能会造成部分浏览器无法发送请求
```

> 参数:

如果是GET请求, 可以是null, 因为GET提交参数会拼接在url后面
如果是POST请求, 传入的就是请求参数

```
"username=张飞&psw=123"
```

d. 第四步, 注册监听

> 在XMLHttpRequest对象的一个事件上注册监听器:

onreadystatechange

> 一共有五个状态:(xmlHttp.readyState)

0状态: 表示刚创建XMLHttpRequest对象, 还未调用open()方法
1状态: 表示刚调用open()方法, 但是还没有调用send()方法发送请求
2状态: 调用完了send()方法了, 请求已经开始

3状态: 服务器已经开始响应, 但是不代表响应结束

4状态: 服务器响应结束!(通常我们只关心这个状态)

> 获取xmlHttp对象的状态:

```
var state = xmlHttp.readyState;//可能得到0, 1, 2, 3, 4
```

> 获取服务器响应的状态码

```
var status = xmlHttp.status;
```

> 获取服务器响应的内容

```
var data = xmlHttp.responseText;//得到服务器响应的文本格式的数据
xmlHttp.onreadystatechange = function(){
    //当服务器已经处理完请求之后
    if(xmlHttp.readyState == 4){
        if( xmlHttp.status == 200 ){
            //获取响应数据
            var result = xmlHttp.responseText;
            result = xmlHttp.responseXML;
        }
    }
}
```

6. jQuery实现AJAX
   a. load方法
      i. $(selector).load(url,data,callback);

         selector -- 选择器, 将从服务器获取到的数据加载到指定的元素中

         url -- 发送请求的URL地址

         data -- 可选, 向服务器发送的数据 key/value数据 如:{"username" : "张飞", "psw" : "123"}

         callback -- 可选, load方法完成后所执行的函数

      示例:

```
$("#username_msg").load("<%= request.getContextPath() %>/AjaxCheckUsernameServlet", {"username": username});
```

      i. $.get方法

```
$.get(url, [data], [callback]);
```

         url -- 发送请求的URL地址

         data -- 可选, 向服务器发送的数据

         callback -- 可选, 请求成功后所执行的函数

      示例:

```
$.get("<%= request.getContextPath() %>/AjaxCheckUsernameServlet",
```

```
        {"username": username}, function(result){
            $("#username_msg").html("<font  style='color:red'>"+result+"</font>");
        });
```

b. $.ajax方法

```
$.ajax(url, [data], [async], [callback]);
    url -- 发送请求的URL地址
    data -- 可选, 发送至服务器的key/value数据
    async -- 可选, 默认为true, 表示异步交互
    type -- 可选, 请求方式 , 默认为"GET"。
    success -- 可选, 请求成功后执行的函数, 函数参数:
        result -- 服务器返回的数据
示例:
    $.ajax({
        "url" : "<%= request.getContextPath() %>/AjaxCheckUsernameServlet",
        "data" : {"username": username},
        "async" : true,
        "type" : "POST",
        "success" : function(result){
            $("#username_msg").html("<font  style='color:red'>"+result+"</font>")
        },
        error:function() {

        }
    });
```

修改EasyMall

2019年5月20日 星期一 14:56

1. 添加ajax用户名是否存在校验

a. 需求：

在用户鼠标离开用户名输入框的时候发生ajax校验，校验的内容是访问数据库，查看数据库中是否包含用户名所输入的用户名称，如果存在就在span中提示用户名已存在，如果不存在提示用户名可以使用。

b. 代码实现：

i. 修改regist. jsp页面：在鼠标离开焦点事件中添加ajax校验。

```
//鼠标离开焦点事件
$("input[name='username']").blur(function(){
    formObj.checkNull("username","用户名不能为空");
    //获取用户名框中的数据，作为ajax请求的参数传递。
    var username = $("input[name='username']").val();
    //如果用户名为空则不应该发生ajax校验
    if($.trim(username)==""){
        return ;
    }
    //ajax的实现
    //鼠标离开用户名输入框之后去数据库完成查重操作。
    $("#username_msg").load("<%
    =request.getContextPath()%>/AjaxCheckUsernameServlet",{"username":username});
});
```

ii. 创建AjaxCheckUsernameServlet：添加如下代码

```
//ajax用户名查重校验
public class AjaxCheckUsernameServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //乱码处理
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        //获取ajax中的请求参数
        String username = request.getParameter("username");
        //连接数据库查重
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try {
            conn = JDBCUtils.getConnection();
            ps = conn.prepareStatement("select * from user where username = ?");
            ps.setString(1, username);
            rs = ps.executeQuery();
            //如果rs.next()为true则标签用户名已经存在
            if(rs.next()){
                //此处的响应数据为ajax的响应数据，会作为结果书写在前台触发ajax的选择器中
                response.getWriter().write("用户名已存在");
            }else{
            //此处的响应数据为ajax的响应数据，会作为结果书写在前台触发ajax的选择器中
```

```
                //返回一个用户名不存在提示到页面中
                response.getWriter().write("恭喜，用户名可以使用");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally{
            JDBCUtils.close(conn, ps, rs);
        }
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);

    }

}
```

2. 验证码实现

用户输入的数据可以由一段代码自动填写实现，所以添加上一个验证码访问自动注册用户，保证垃圾数据不进入数据库当中。

a. 代码实现：

i. 修改regist.jsp页面：

图片在加载资源的时候是通过src访问的，所以src可以指向一个动态资源，这个动态资源生成一张图片。

```
<td>
    <input type="text" name="valistr"/>
    <img id="img" src="<%=request.getContextPath() %>/ValidateServlet" width="" height="" alt="" />
    <span ></span>
</td>
```

i. 创建ValidateServlet

```
//生成验证码
public class ValidateServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //控制缓存，防止浏览器使用验证码图片的缓存
        response.setDateHeader("Expires", -1);
        response.setHeader("Cache-control", "no-cache");
        //调用工具类产生一个验证码的图片
        VerifyCode vc = new VerifyCode();
        vc.drawImage(response.getOutputStream());
        //获取验证码的纯文本内容
        String code = vc.getCode();
        System.out.println(code);

    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);

    }
```

```
}
```