一、ServletConfig

2019年1月28日 16:31

1. ServletConfig

代表当前Servlet在web.xml文件中配置信息的对象

a. 获取ServletConfig对象

ServletConfig scf = this.getServletConfig();

b. ServletConfig功能

i. 获取当前Servlet的初始化参数

在web.xml中的<Servlet>标签的内部可以为当前Servlet配置零个或多个初始化参数, 这些参数相当于我们为Servlet配置了一些初始化信息, 可以通过ServletConfig对象(在当前Servlet内部可以获取该对象)来获取这些信息。

- 获取的方法:

getInitParameter(String name);

getInitParameterNames();

ii. 获取ServletContext对象

- 获取的方法:

this.getServletConfig().getServletContext();

2. ServletContext

代表当前web应用的对象

在web应用被加载后, 服务器会立即创建出代表当前web应用的ServletContext对象, 创建后该对象会一直驻留在内存中唯一的代表当前web应用, 直到服务器关闭或者web应用移出容器时为止, 随着web应用的销毁, ServletContext对象也跟着销毁!

a. 获取ServletContext对象

this.getServletConfig().getServletContext();

或者

this.getServletContext();

b. ServletContext功能

i. 获取web应用的初始化参数

在当前Servlet中配置的参数信息, 只能在当前Servlet中通过ServletConfig对象来获取, 在其他的Servlet中无法获取!

如果有一段初始化参数, 希望不是属于某一个Servlet对象, 而是整个web应用都可以使用, 可以将这些参数配置到web.xml的根目录下, 配置在根目录下的这些信息是属于整个web应用的, 可以通过代表整个web应用的ServletContext来获取。

```
<context-param>
    <param-name>scparam1</param-name>
    <param-value>scvalue1</param-value>
</context-param>
...
```

1) 获取参数方法:

getInitParameter();

getInitParameterNames();

ii. 作为域对象来使用

ServletContext对象是一个域对象, 利用这个对象上的map就可以在整个web应用内实现资源的共享.

**获取方法:

setAttribute();

getAttribute();

removeAttribute();

getAttributeNames();

生命周期:

和WEB应用的生命周期一样长

（WEB应用被加载之后创建，WEB被销毁该对象也被销毁）

作用范围:

在整个WEB应用内都可以被看见

主要功能:

在整个WEB应用内实现资源的共享

iii. 获取web资源

在web开发中, 如果要在程序中获取web应用的资源文件, 需要写一个路径时:

1) 如果写一个相对路径 File file = new File("conf.properties"); 这时会到程序启动的目录下去寻找这个资源, 对于web应用来说, web应用是在容器中运行的, 这时候就会到tomcat/bin目录下去找这个文件, 找不到!!
2) 如果写一个绝对路径, 则会到程序启动目录的硬盘根目录去找这个资源文件, 也找不到!!
3) 写一个盘符开始的硬盘路径, 可以解决这个问题, 但是这种写法一旦换一个发布环境, 路径很可能是错误的. 这种写法也不可取!!
4) 可以通过ServletContext提供的方法来解决这个路径问题:

   sc.getRealPath("xxx"); 传入一个相对于web应用根目录的资源文件的路径, 这个方法会在传入的路径的前面动态的拼接上当前web应用根目录的硬盘路径, 从而拼接出当前资源文件的硬盘路径, 由于web应用的根目录的硬盘路径是动态获取的. 没有写死, 即使换一个发布环境, 也能获取到正确的路径
5) 当没有ServletContext对象可以使用时, 可以通过类加载器来加载资源, 类加载器提供了方法来加载资源, 但是要求传入的文件路径必须是相对于类加载器加载类的路径

   getResource(xxx)

   getResourceAsStream(xxx)

二、ajax

2019年1月28日 16:30

1. AJAX(!!!重要)
   a. AJAX是什么:

* asynchronous js and xml: 异步的js和xml
* 可以利用js访问服务器, 而且是异步访问!
* 通常服务器给浏览器响应的是一个完整的页面, 而在AJAX中, 由于是利用js访问服务器, 再由js接受响应, 局部刷新页面, 所以服务器不用给浏览器响应整个页面了, 而只是数据。
* 服务器响应的数据:
  > text 纯文本 "用户名已存在!!"
  > xml
  > json: js提供的一种数据交互格式, 在js中很受欢迎

ajax --> ajaj --> aj

   b. 同步交互和异步交互

同步:
* 向服务器发一个请求, 必须等待响应结束, 才能发送第二个请求, 在服务器处理期间, 浏览器不能做其他操作
刷新范围：刷新整个页面

异步:
* 向服务器发一个请求, 不用等待响应结束, 就可以发送第二个请求, 在服务器处理期间, 浏览器可以做其他操作
刷新范围：可以使用js接受服务器的响应, 再利用js局部刷新页面

   c. AJAX的应用场景:

* 百度的搜索框
* 注册用户时, 校验用户名是否被注册过

...

   d. AJAX的优点和缺点

优点:
* 异步交互, 提高了用户体验!
* 服务器只响应部分数据, 而不是整个页面, 所以降低了服务器的压力!

缺点:3
* ajax不能应用所有的场景
* ajax会无端的增加访问服务器的次数, 给服务器带来了压力!!

   e. .JavaScript实现AJAX(只需四步)
      i. 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;
}
```

ii. 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

iii. 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"

iv. 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;
            }
        }
    }
```

f. jQuery实现AJAX

i. load方法

```
$(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});
```

ii. $.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>");
    });
```

iii. $.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>")
        }
    });
```

三、Easymall注册功能实现_02

2019年1月28日 16:32

1. AJAX实现校验用户名是否存在
   a. 编辑regist.jsp页面

```
$().ready(function(){
    $("input[name=username]").blur(function(){
        var username = $("input[name=username]").val();
        //检查用户名是否为空
        if(!formObj.checkNull("username", "用户名不能为空")){
            return;
        }

        //校验用户名是否存在
        //$("#username_msg").load("<%= request.getContextPath()
        %>/AjaxCheckUsernameServlet", {username: username});
    });
});
```

   b. 创建AjaxCheckUsernameServlet，处理请求

```
//1.处理乱码
request.setCharacterEncoding("utf-8");
response.setContentType("text/html;charset=utf-8");

//2.获取请求参数
String username = request.getParameter("username");

//3.到数据库中查询用户名是否存在
.....
.....
//4.做出响应
if(rs.next()){//用户名存在
    response.getWriter().write("<font style='color:red'>用户名已存在</font>");
}else{
    response.getWriter().write("<font style='color:red'>恭喜,用户名可以使用!</font>");
}
```

2. 利用java程序开发验证码图片 -- 不需要练习,重点掌握禁止验证码缓存

```
//1.将VerifyCode工具类直接拷贝到cn.tedu.utils包下
//2.创建ValiImageServlet,代码实现如下:
```

```
VerifyCode vc = new VerifyCode();
vc.drawImage(response.getOutputStream());
//获取图片验证码
String valistr = vc.getCode();
//TODO--将验证码保存在session中
```

3. 禁止缓存验证码图片

编辑ValiImageServlet, 在最前面添加控制浏览器不缓存图片的代码:

```
response.setDateHeader("Expires", -1);
response.setHeader("Cache-Control", "no-cache");
```

4. 点击图片换一张

```
//1.编辑image标签, 添加点击事件
<img onclick="changeImage(this)" src="<%= request.getContextPath()
%>/ValiImageServlet" alt="" />
//2.js代码实现换一张图片
<script>
    function changeImage(thisObj){
        thisObj.src = "<%= request.getContextPath() %>/ValiImageServlet?time="+new
        Date().getTime();
    }
</script>
```