

# Servlet

## 1.Servlet介绍

## 2.HttpServletRequest和HttpServletResponse

## 3.Cookie 和 Session

cookie 和 session 都是用于在web中跟踪用户状态的技术。

### 3.1 Cookie

Cookie是存放在客户端浏览器上的小文本文件，由服务器发往浏览器，当用户再一次访问网站的时候，浏览器会自动的把cookie的内容发往服务器。比如cookie中可记住用户的登录状态或语言偏好等。

Cookie是有一定的有效期限的，可以设置过期时间。

Cookie是按照键值对的方式存储一些字符串

#### 3.1.1 Cookie的常用方法

```
Cookie[] cookies = request.getCookies(); //获取当前网站的所有cookies数据
cookie.getName(); //获取cookie中的key
cookie.getValue(); //获取cookie中的value
new Cookie("lastLoginTime", System.currentTimeMillis()); //新建一个cookie
cookie.setMaxAge(24*60*60); //设置cookie的有效期限，以秒为单位，如果不设置关闭浏览器就失效
response.addCookie(cookie); //响应给客户端一个cookie
```

### 3.2 Session

Session即会话，是用户打开一个浏览器，访问一个站点之后，在这个站点内又点击了多个链接访问多个web资源，直到浏览器关闭。整个过程称之为一次会话。

**原理：**第一次访问站点时，服务器会生成一个sessionID，随着响应，将该sessionID的值保存在浏览器的cookie中。

1.当用户第一次访问该站点使用session时，服务器内部会判断是否有该浏览器对应的session存在，如果没有则生成一个sessionID

2.将该sessionID的值伴随着响应，添加到客户端浏览器的cookie中

3.接下来的每一次访问，浏览器都会将cookie中的sessionID带给服务器

4.服务器得到sessionID后会找到对应的空间，进行数据的存储操作

#### 3.2.1 Session的常用方法

类型：HttpSession

获取：request.getSession();

```
public String getId();//获取sessionID
public ServletContext getServletContext();//获取ServletContext
public Object getAttribute();//获取属性
public void setAttribute(String name,Object value);//添加属性
public void removeAttribute(String name);移除属性
public void invalidate();//注销
public boolean isNew();//判断是否为新建
```

### 3.2.2 web.xml中配置session自动过期时间

```
<!-- 设置session超时时长 -->
<session-config>
    <!-- 以分钟为单位 -->
    <session-timeout>15</session-timeout>
</session-config>
```

## 4.作用域对象

#### 1. pageContext

生命周期：当请求时开始，当响应结束时销毁

作用范围：整个页面

#### 2. request

生命周期：在service方法调用前由服务器创建，传入service方法。整个请求结束时，request生命结束

作用范围：整个请求的URL中，包括转发

#### 3. session

生命周期：在第一次调用request.getSession()方法时，服务器会检查是否已有对应的session，如果没有就在内存中创建一个并返回。

作用方位：一次会话

作用：保存登录信息

#### 4. application

生命周期：当Web应用被加载进容器时创建代表整个应用的application对象，当服务器关闭或Web应用被移除时，application对象跟着销毁。

作用范围：整个Web应用

作用：存储公共数据

类型：ServletContext, ServletContext application = request.getServletContext();

常用方法：

```
void setAttribute(String name,Object value);
Object getAttribute(String name);
```

## 5.过滤器 - Filter

### 5.1 什么是过滤器

Filter，过滤器，是JavaWeb中的技术之一。

作用1：是对访问web服务器请求进行拦截，过滤。例如：Jsp，Servlet，html等请求进行拦截，从而实现一些特殊的功能。例如实现URL级别的权限访问控制，过滤敏感词汇，压缩响应信息，以及处理编码等。

作用2：对HttpServletResponse进行拦截，处理。

### 5.2 过滤器如何实现功能

Filter接口中提供了一个doFilter方法，当开发人员配置好对哪个资源进行拦截后，当web服务器要请求该资源之前（即调用service方法之前），会先调用Filter的doFilter方法。doFilter方法中会有一个filterChain对象，用于继续传递给下一个filter。

### 5.3 语法

定义Filter

```
import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class CharacterEncodingFilter implements Filter {

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain filterChain)
        throws IOException, ServletException {
        // TODO Auto-generated method stub
    }
}
```

web.xml

```
<filter>
    <filter-name>CharacterEncodingFilter</filter-name>
    <filter-class>com.zx.filter.CharacterEncodingFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

完整例子：

```

package com.zx.filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class CharacterEncodingFilter implements Filter {

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain filterChain)
        throws IOException, ServletException {
        // TODO Auto-generated method stub
        System.out.println("Filter In(Request)");
        request.setCharacterEncoding("utf-8");
        response.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        System.out.println("Filter Chain");
        filterChain.doFilter(request, response);
        System.out.println("Filter In(Response)");
    }

}

```

## 6.Servlet 初始化参数

### 6.1 Servlet初始化参数

每个Servlet允许设置自己的初始化参数，并且在servlet加载的过程中获取出来并使用。

通常情况下会将servlet要用到的配置内容存放在此。

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="webApp_ID" version="4.0">
    <servlet>
        <servlet-name>IndexAction01</servlet-name>
        <servlet-class>com.zx.action.IndexAction01</servlet-class>
        <init-param>
            <param-name>Encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>IndexAction01</servlet-name>
        <url-pattern>/IndexAction01.do</url-pattern>
    </servlet-mapping>

```

```
</web-app>
```

```
public class IndexAction01 extends HttpServlet {

    private String encoding ;

    @Override
    public void init(ServletConfig config) throws ServletException {
        // TODO Auto-generated method stub
        this.encoding = config.getInitParameter("Encoding");
    }

    @Override
    protected void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        system.out.println(this.encoding);
    }
}
```

## 6.2 全局初始化参数

## 7. 监听器 - Listener

---