# 1.回顾

1. cookie:存储一段字符串在本地浏览器

案例:用户记住密码在cookie有效期内的免登录,不用再次输入密码和用户名

1. 怎么存储: Cookie cookie= new Cookie(“key”,”value”);

response.addCookie(cookie);

1. 怎么获取:Cookie[] cs=request.getCookies(); 进行遍历
2. 怎么设置cookie有效期:默认-1:会话期间,设置有效期:cookie.setMaxAge(60) 1分钟

# 过滤器

抽烟:过滤嘴

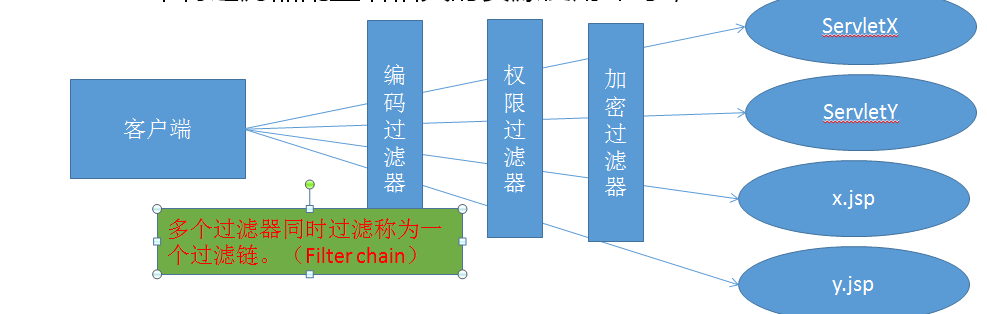
茶水间: 倒水->自来水->净水器->纯净水

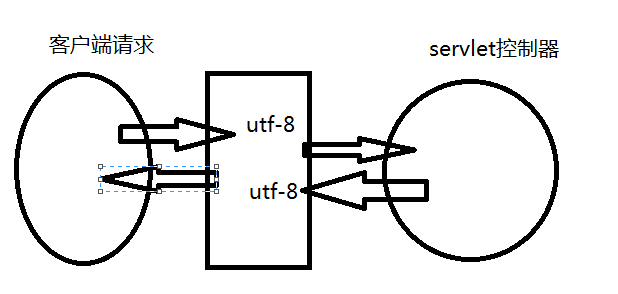
人:肾

## 2.1 概念

在Web应用中，如果对服务器端的多个资源（Servlet/JSP）有“通用”的处理，可以在每个资源中写相同的代码，而这样做显然过于冗余，修改时就需要逐一修改，效率低下；

过滤器可以解决这样的问题：把通用的、相同的处理代码用过滤器实现，然后在web.xml中将过滤器配置给相关的资源使用即可；





JSP 和 Servlet 中的过滤器都是 Java 类。

过滤器可以动态地拦截请求和响应，以变换或使用包含在请求或响应中的信息。

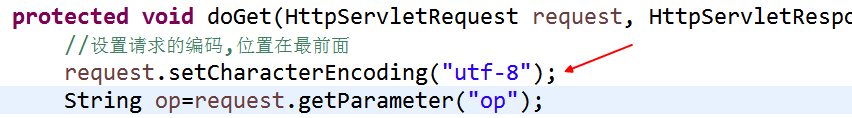
可以将一个或多个过滤器附加到一个 Servlet 或一组 Servlet。过滤器也可以附加到 JavaServer Pages (JSP) 文件和 HTML 页面。

过滤器是可用于 Servlet 编程的 Java 类，可以实现以下目的：

* 在客户端的请求访问后端资源之前，拦截这些请求。
* 在服务器的响应发送回客户端之前，处理这些响应。

## 2.2 过滤器常见案例：编码格式、访问控制

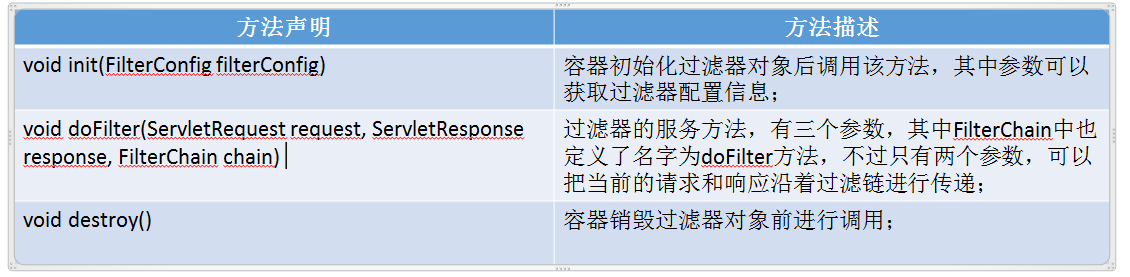
我们发现在Servlet的代码中，任何Servlet中都需要进行编码过滤;



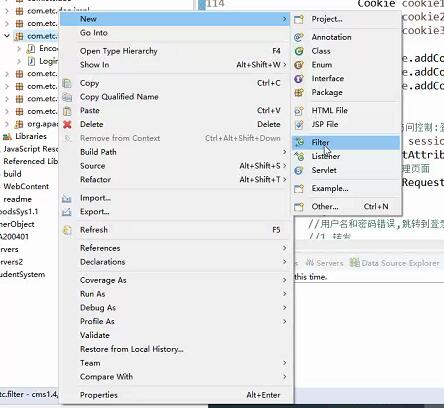
我们方法有些页面，需要进行访问控制。系统中后台管理中，几乎所有的jsp(add.jsp)页面都要加入访问控制,判断session中是否存在特定的对象信息，如果不存，页面跳转到登录界面.

## 2.3 过滤器的开发步骤

### 自定义类实现Filter接口；







### 实现接口中的方法，重点是doFilter方法

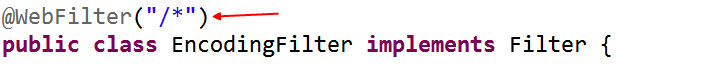


### 3、web.xml中设置过滤器对哪些请求有效:

第一种：web.xml手动配置



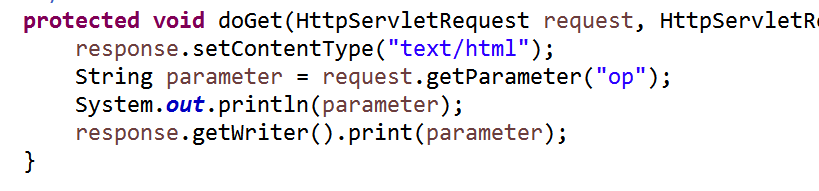
第二种：使用注解

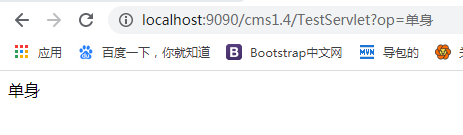


### 测试：演示post提交方式中文不会乱码

测试cms.4的搜索功能

Get请求:

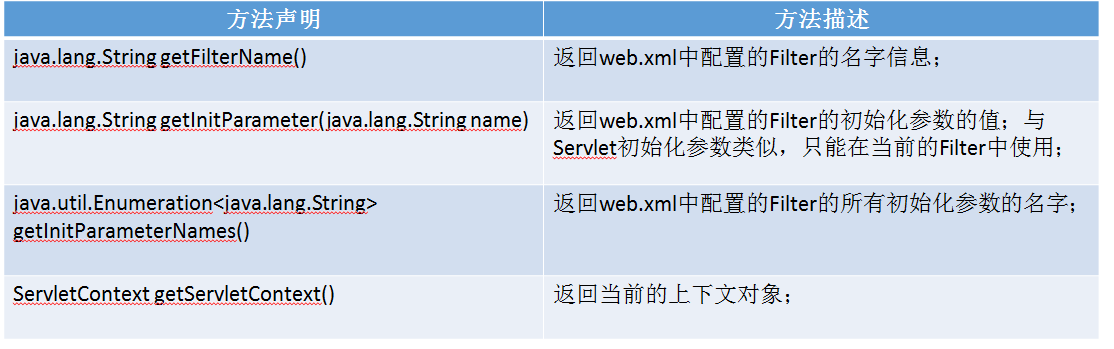




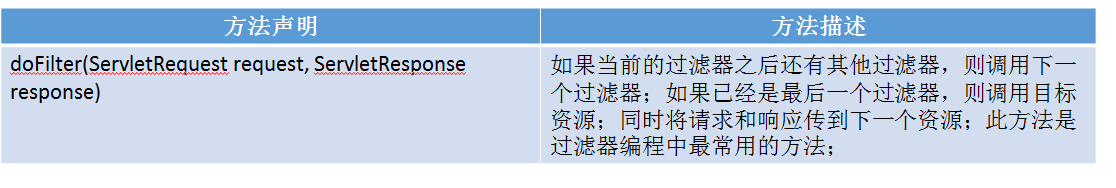


除了Filter接口外，与过滤器有关的还有FilterConfig和FilterChain接口；其中FilterConfig接口中定义了如下方法：

Filterconfig中的方法:



FilterChain中的方法 :



## 2.4过滤器课堂案例

## 2.5 编码格式

## 2.6实现CMS1.4过滤器设置编码格式

配置与注解二选一

## 2.7用户访问权限控制(只有登录的用户才能访问mvc后台页面)

以前我们jsp页面的访问控制是写在对应jsp页面,现在应该在过滤器中过滤

后台mvc目录下的jsp,如果没有登录的管理员是不可以直接通过jsp路径进行访问,在此之前,我们都是在jsp页面中写访问控制的代码

出现问题:如果mvc后台管理界面有10个,20个,30个...难道每一个页面都复制同一段代码写一次?这样代码明显冗余,修改不方便

解决访问:通过设置一个访问控制器的过滤器,过滤指定的文件,便于维护

实现CMS1.4通过过滤器设置后台的权限控制

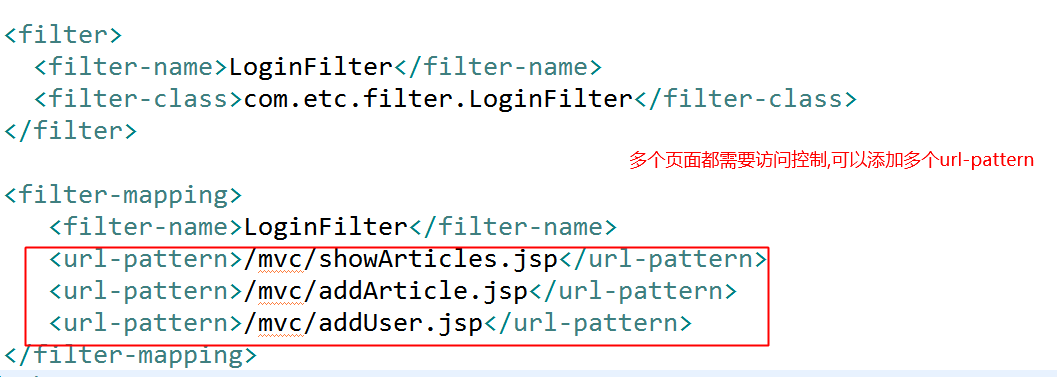
### 单个页面的过滤



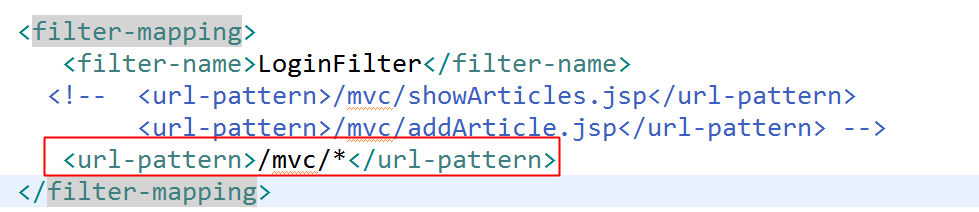
web.xml:



### 多个url的过滤:



### 同一目录的过滤

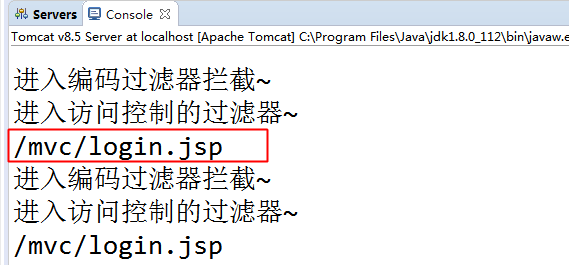


测试:发现进入登录页面的时候,一直在跳登录的请求

解决方式1:将login.jsp放到根目录,不放在mvc里面就不会拦截

解决方式2:在过滤器中对请求的url进行判断,如果发现请求的是login.jsp,就直接放行





## 2.8怎么控制多个Filter的顺序

方式1:如果是注解的方式,多个Filter的执行顺序可以通过Filter的文件名来控制,按照首字母排序(扩展)

方式2:xml配置的方式(从上往下)



# 3.拓展：监听器（难，了解）

### 监听器作用：

事件发生的时间往往是不确定的，当事件发生的时候需要进行一些处理时，就可以使用监听器；

例如session对象:被创建或者被销毁就是一个事件，但是何时被创建或销毁是不确定的，如果需要只要session对象被创建或销毁就进行相应处理，就可以使用监听器.

### 监听器相关的API包括事件类以及监听器接口；

事件类定义了事件类型，监听器接口定义了监听事件的方法；

### Servlet API中定义了6种事件类型

* **上下文相关的事件 application**

ServletContext**Event**：该类表示上下文事件，当应用上下文对象发生改变，如创建或销毁上下文对象时，将触发上下文事件。

ServletContextAttribute**Event**：该类表示上下文属性事件， 当请求中的属性改变，如增加、删除、覆盖请求中的属性时，触发请求属性事件。

application.setAttibute(“key”value)

application.removeAttibute(“key”value)

* **请求相关的事件 request->HttpServerRequest -> ServletRequest**

ServletRequest**Event**：该类表示请求事件，当请求对象发生改变，如创建或销毁请求对象时，触发请求事件。

ServletRequest**Attribute**Event：该类表示请求属性事件，当请求中的属性改变，如增加、删除、覆盖请求中的属性时，触发请求属性事件。

request.setAttibute(“key”value)

request.removeAttibute(“key”value)

* **会话相关的事件 session-> HttpSession**

HttpSession**Event**：该类表示会话事件，当会话对象发生改变，如创建或销毁会话对象，活化或钝化会话对象时，将触发会话事件。

HttpSession**BindingEvent**：该类表示会话绑定事件，当会话中的属性发生变化时，如增加、删除、覆盖会话中的属性时，将触发会话绑定事件。

session.setAttibute(“key”value)

session.removeAttibute(“key”value)

小结:

1.有3对象:上下文(application)/request/session

2.每个对象2个事件:创建或销毁(与本身相关的事件)/跟属性相关,总的6个事件

### 8种监听器接口:

Servlet API中定义了8种监听器接口，用来监听不同的事件类型

* 上下文相关的监听器

ServletContextListener：上下文监听器，监听ServletContextEvent事件。

ServletContextAttributeListener：上下文属性监听器，用来监听ServletContextAttribute事件。

* 请求相关的监听器

ServletRequestListener：请求监听器，监听ServletRequestEvent事件。

ServletRequestAttributeListener：请求属性监听器，用来监听ServletRequestAttributeEvent事件。

* 会话相关的监听器

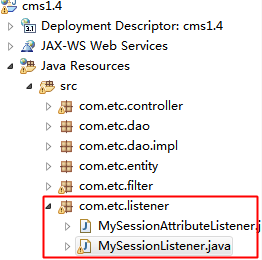
**HttpSessionListener：会话监听器，监听HttpSessionEvent。**

HttpSessionActivationListener：会话活动(化)监听器，监听HttpSessionEvent事件。

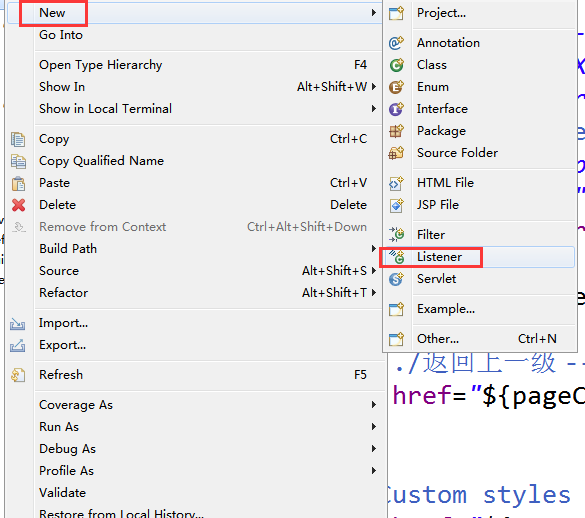
**HttpSessionAttributeListener：会话属性监听器，监听HttpSessionAttributeEvent事件。**

**HttpSessionBindingListener：会话绑定监听器，监听HttpSessionAttributeEvent事件**

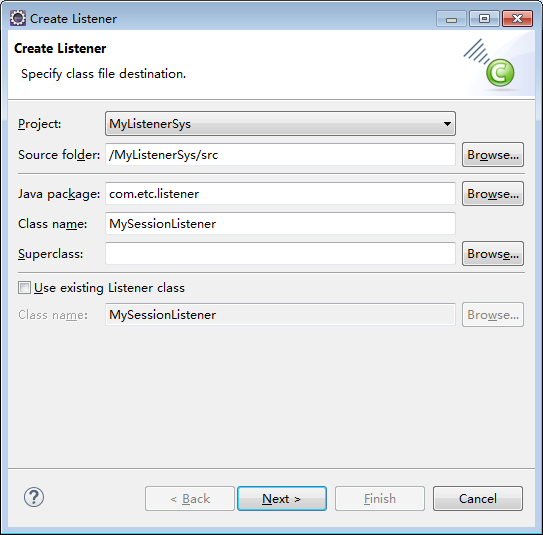
### 监听器的开发及配置

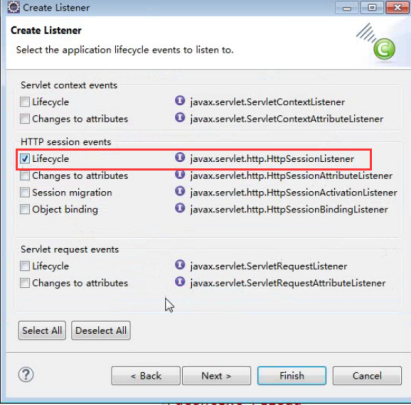


New Listener1、写一个类实现相应的XXXLisntener接口；重写接口中的方法，实现监听的功能；



使用向导的方式,直接选择对应的接口实现即可 .

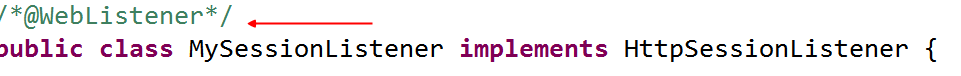






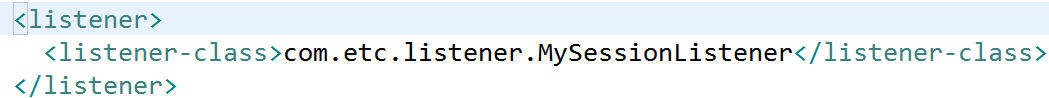
### 配置listener的2种方式

1. 使用注解来配置:

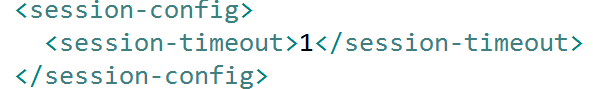


2、Web.xml中配置监听器:

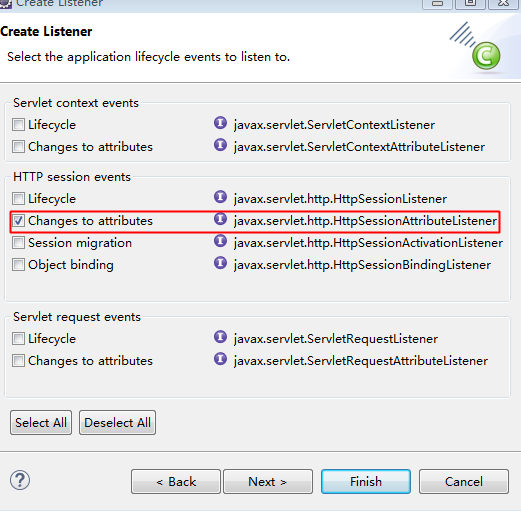
Web.xml中配置

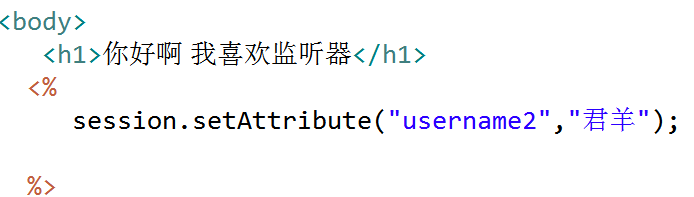


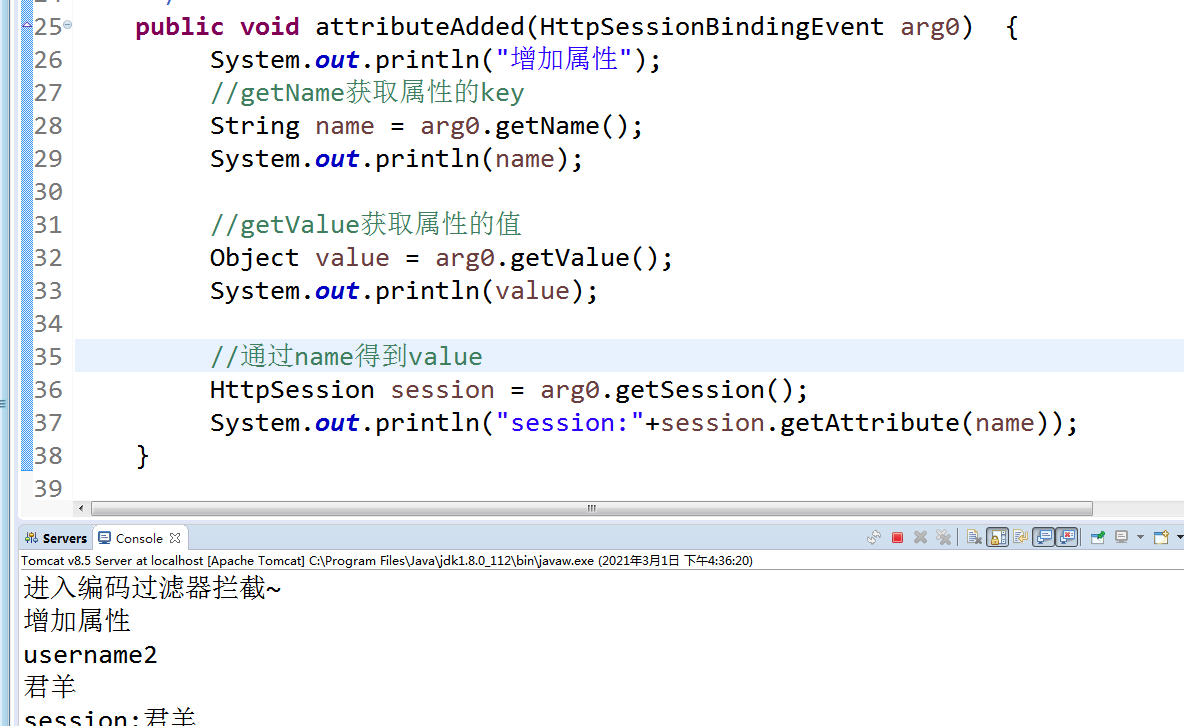
session设置超时时间:



### Attribute属性add监听

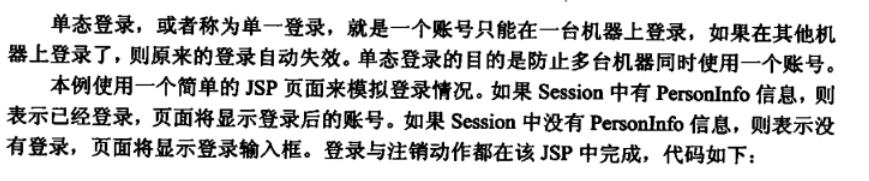






### 监听器课堂案例:

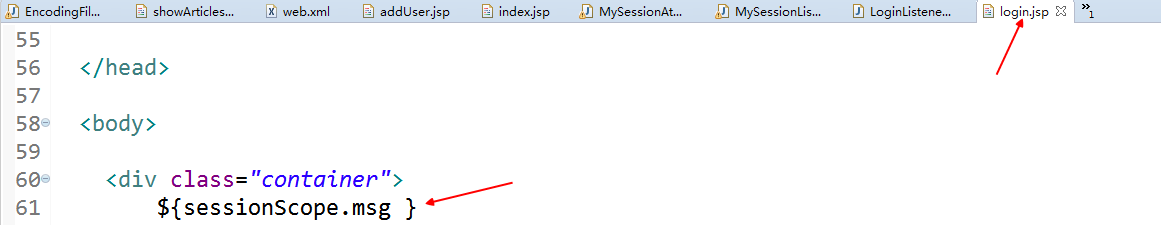
使用监听器实现单态登录:



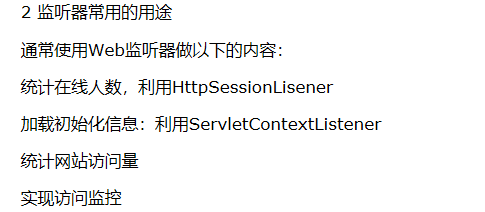
分析:

在session的监听器中增加一个Map用来存储用户登录的信息;





自己演示一遍



## 监听器小结(难):

应用场景:对session或者request以及application操作进行监听; 日志;单态登录等.

怎么用: 创建一个类 实现接口 XXXAttributeListener; 重写方法: 注解或者xml配置webListener.