## Servlet生命周期：

* init（等初始化完成才会接收请求） 、
* service（doGet、doPost等）、
* destroy(等处理完所有请求后才执行销毁操作，可以通过shutdown.bat测试效果)

## ServletConfig：

对应某个serlvet的配置信息，用于初始化serlvet初始化数据

## ServeltContext：

表示可访问整个web应用的资源，属于整个应用共享的区域。

用来读取属性，读取初始化参数，访问程序文件等

以”/”作为开头的路径的问题。

1. 本应用的根 this.getServletContext().getResourceAsStream("/a.txt");
2. Web服务器的应用集合的根 form action=”/ctx/sss.jsp”

## 跳转区别

redirect重定向 全新请求，有不同的请求数据，可以跳出到外部url

forward请求转发：同一请求，有相同的请求数据，只能在本web应用内

## 监听器Listener

ServletRequestListener 监听所有请求，从请求开始到结束

ServletContextListener监听web应用初始化、应用销毁

ServletSessionListener：监听回话初始化、销毁

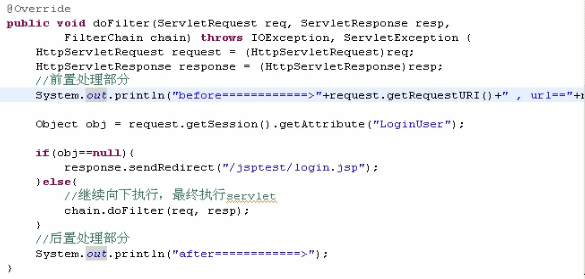
## 过滤器Filter

是什么：在servlet处理请求之前、应答之后实现一定功能的技术。

能干什么：可以在request到达servlet之前预处理request，也可以在离开servlet时处理response。

好处：功能复用、可实现“可插拔式”架构、实现AOP，在不之情的情况下增加功能（类似装饰模式）

MyFilter简单写法（判断是否登录）：

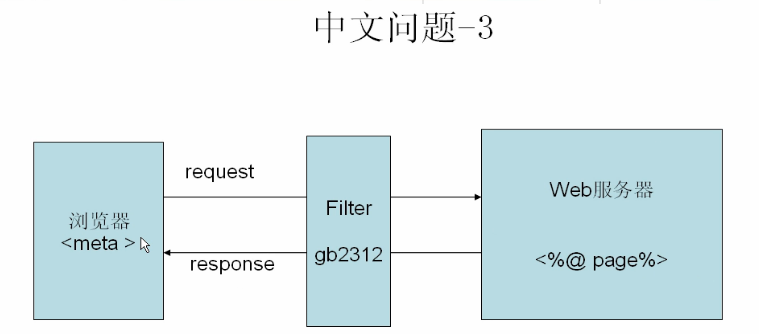


* 过滤请求中非法参数（预处理request）：

增加一个包装类MyReqeust继承HttpServletRequestWrapper，修改获取参数方法即可。

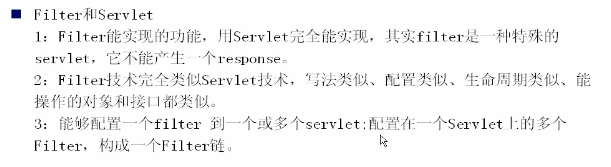
chain.doFilter(new MyRequest(req) , resp);

* 中文问题（三端各自设置meta、Filter、page三者设置字符集一致）：



针对post提交，如果是get提交的话，修改tomcat中URI编码方式

## Servlet和Filter比较



## 回话Session和Cookie

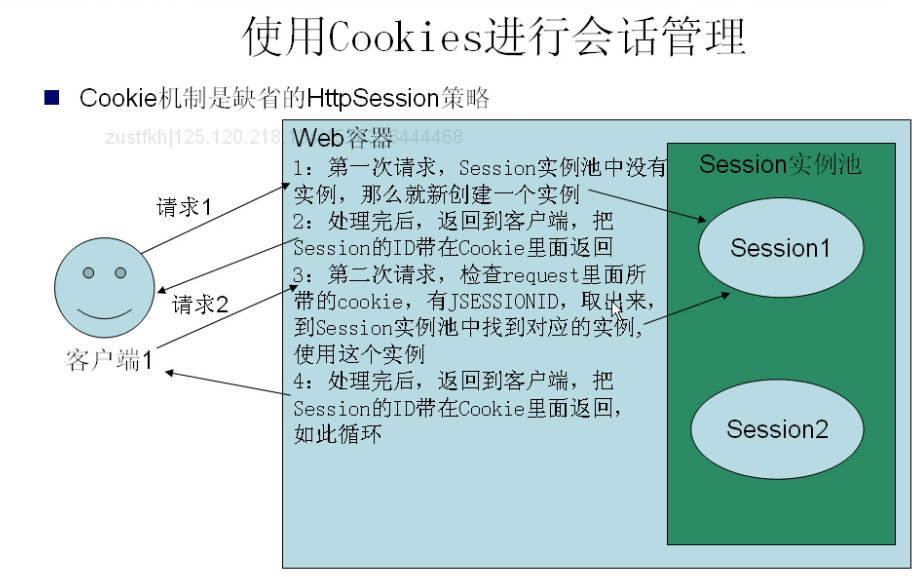
Session：每个客户端根据jsessionID对应一个session内存块，存储于服务器内存中。

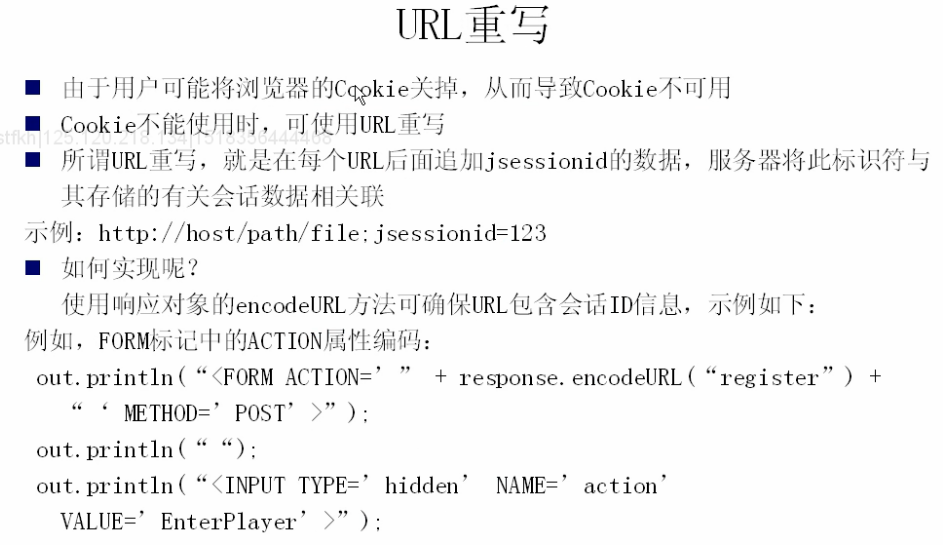
ServletContext服务端共享，不区分客户端

Session服务端共享块，区分客户端

Cookies：该域名下（路径）的所有Cookies在每次一请求中均发送给web服务器。

Cookie机制是缺省的HttpSession策略。默认会将sessionID保存到cookies中来区分浏览器





如果客户端禁止cookies，可以使用如下替代：



|  |  |  |
| --- | --- | --- |
|  |  |  |
| ServeltContext |  |  |

## servlet并发问题

web应用中六中属性范围

|  |  |  |
| --- | --- | --- |
| **范围** | **线程安全** | **说明** |
| 局部变量 | 是 | 每个线程各自保存 |
| 实例变量 | 否 | 多线程共享 |
| 类变量 | 否 |  |
| 请求范围 | 是 | ApI：ServletRequest接口  （request） |
| 会话范围 | 否（理论上存在） | 用户可能有多个浏览器活动，访问同一个web资源。  HttpSession（session） |
| 应用范围 | 否 | Web应用中所有servlet共用资源  ServletContext（application） |

## Web安全

* 认证（权限匹配）、
* 授权（权限分配）、
* 维护数据完整性（https）、
* 访问跟踪（每次访问记录过程、产生日志）、
* 处理web攻击

可以使用tomcat中配置用户信息来控制权限。

非对称加密RSA

## Jsp隐含变量

变量：一个标识符，用来指向一块对象实例的内存区域，或者一个值，通过操作这个变量可以操作变量所代表的实例和值。

## 连接池

开发难点：动态控制实例数量

## 观察者模式（发布-订阅）

视图（观察者） - 模型（被观察者）

## Jsp和Servlet开发最佳实践

