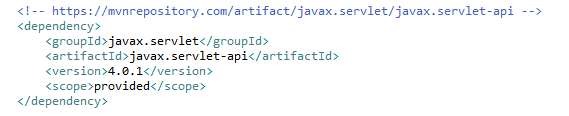
Servlet笔记

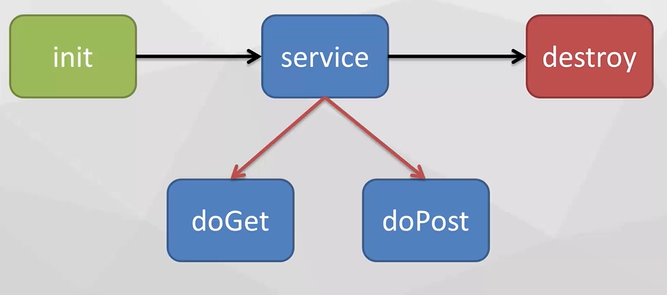
# 配置依赖pom.xml



# 配置web.xml

1. 配置servlet名称和类名
2. 配置servlet的url映射

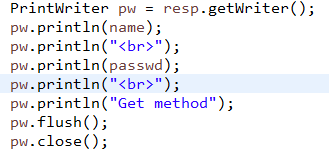
# Servlet生命周期



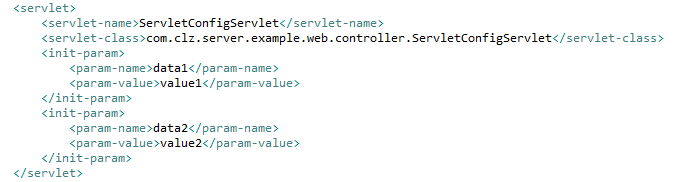
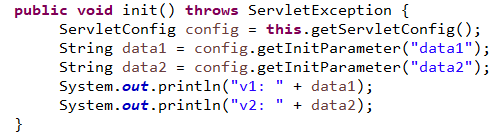
# Servlet类

1. 继承HttpServlet类
2. 重载init方法
   1. Servlet初始化时调用
3. 重载service方法
   1. 使用servlet时调用
4. 重载doGet方法
   1. 调用get方法时调用
5. 重载doPost方法
   1. 调用doPost时调用
6. 重载destroy方法
   1. Servlet销毁时调用

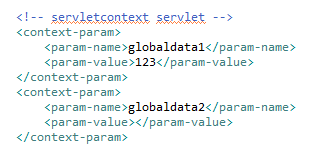
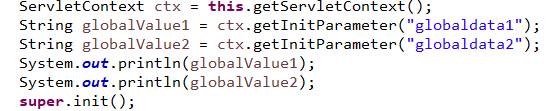
# Get和Post方法

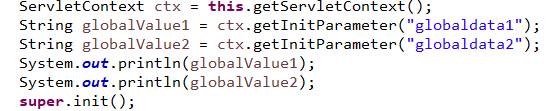
1. 获取参数
2. 返回信息
   1. 获取PrintWriter对象
   2. pw.println()写入信息
   3. pw.flush()刷新缓存
   4. pw.close()关闭对象

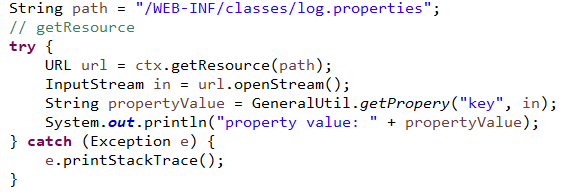
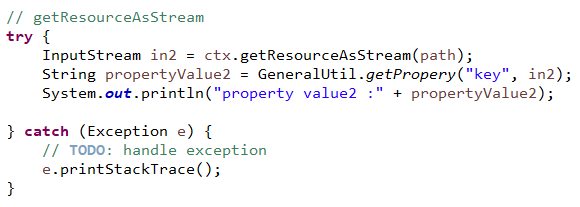
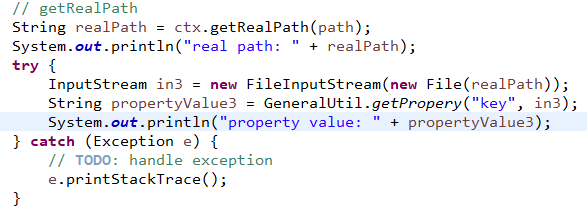
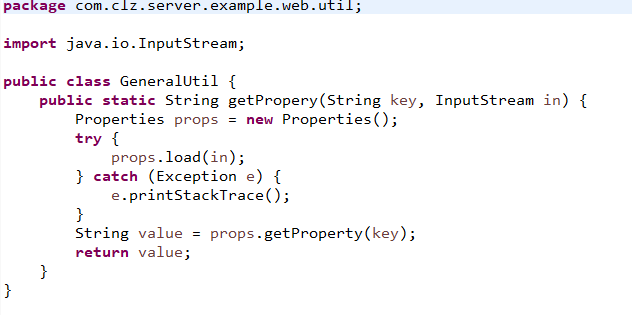
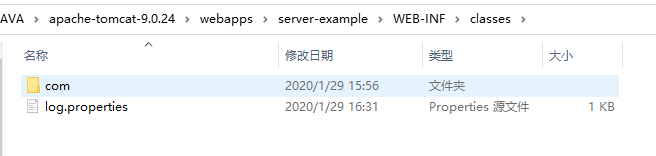
# ServletConfig对象

1. 单个Servlet配置信息
   1. web.xml中配置初始化参数
   2. Servlet获取参数
      1. 获取ServletConfig对象
         1. this.getServletConfig()
      2. 获取参数值
         1. Config.getInitParameter(<parameter-name>)

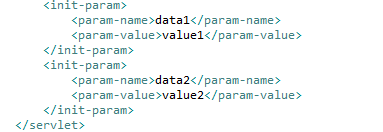
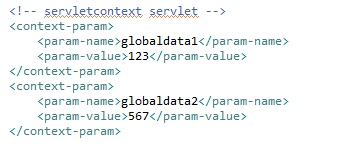
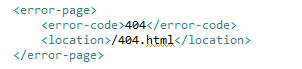
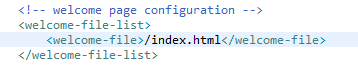
# ServletContext对象

1. 不同Servlet共享配置信息
   1. web.xml配置
   2. Serlvet获取参数

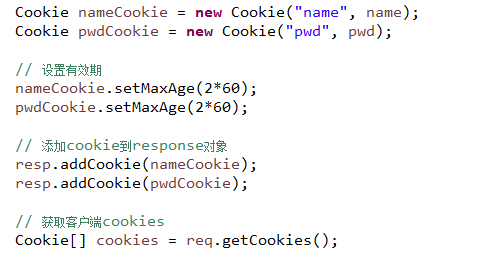


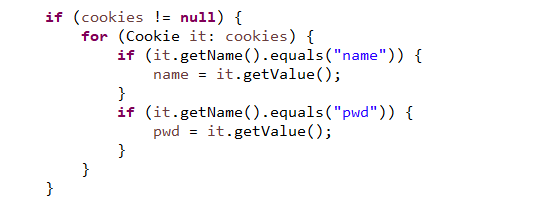
1. Servlet间通信
   1. 第一个Servlet设置全局属性
   2. 第二个Servlet获取属性
2. 读取外部资源配置文件信息
   1. getResource
   2. getResourceAsStream
   3. getRealPath
   4. 其中
      1. GeneralUtil类为自定义的类
      2. 为什么路径是这个
         1. 在tomcat中文件的实际路径

# Servlet配置

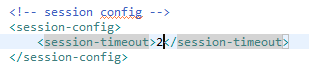
1. Web应用程序结构
   1. 公共资源
   2. META-INF目录
   3. WEB-INF目录
2. 部署描述符
   1. web.xml
   2. 设置web应用程序的组件部署信息
      1. Servlet声明
      2. ServletConfig配置
      3. ServletContext配置
   3. Servlet的url-mapping
      1. Servlet支持同一个servlet对应多个url-pattern
      2. url-pattern支持模糊匹配，如/hello/\*
      3. 匹配原则
         1. 精确路径匹配，完全匹配
         2. 最长路径匹配，最长前缀匹配
         3. 扩展名匹配
         4. Default servlet匹配或者放弃，/
   4. 需要在servlet启动时执行操作
      1. 负数或不设置
         1. Servlet会在第一次使用servlet时初始化
      2. 0或正数
         1. Servlet按数字大小先后启动初始化
   5. 配置自定义错误界面
      1. 更高级的做法还可以添加exception-type元素·捕获一个Java异常类型
   6. 欢迎页面
      1. <welcome-file-list>
      2. 指定多个欢迎页面<welcome-file>
      3. 没有找到时，顺序加载
   7. 浏览器 打开文件或下载文件
      1. MIME类型映射
         1. Multipurpose Internet Mail Extensions
      2. <mime-mapping>定义扩展文件名映射类型，包括两个子元素
         1. <extension>：扩展名
         2. <mime-type>：MIME标准

# Servlet session和Cookie配置

1. Session使用
   1. 使用场景
      1. 偏好记录
      2. 自动登录
      3. 浏览记录
   2. 原理
      1. 客户端或服务端保存用户数据
2. Cookie
   1. 会话数据
   2. 保存在客户端
   3. Key-Value数据
   4. 生命周期
      1. 默认会话结束后失效
      2. setMaxAge设置cookie有效期



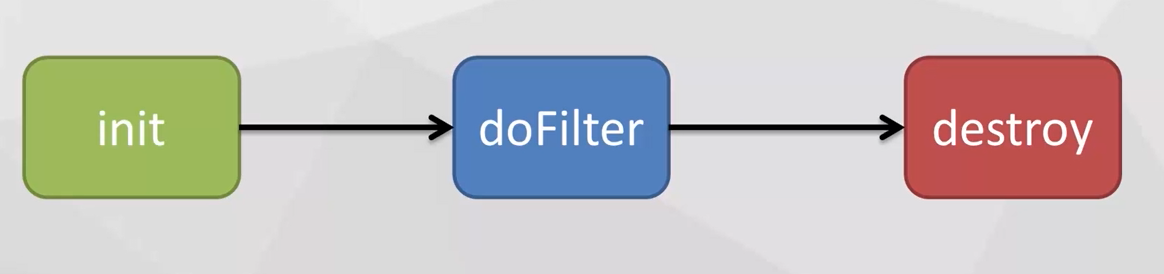
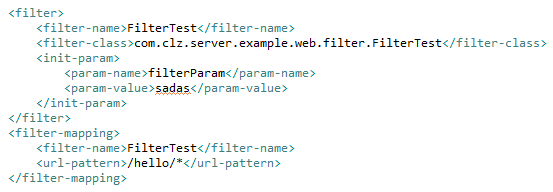
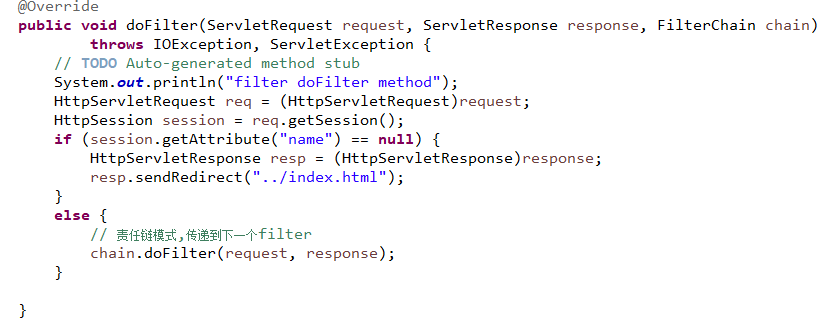
* 1. 缺陷
     1. 大小和数量的限制
     2. 数据安全性问题

1. Session
   1. 保存在服务端
   2. HttpSession
   3. 生命周期
      1. 默认有效期30分钟
      2. setMaxInactiveInterval设置有效期
      3. 部署描述符配置有效期
      4. Invalidate使Session失效
   4. 通过request对象获取
   5. 也可通过web.xml配置Session失效时间
      1. 单位分钟

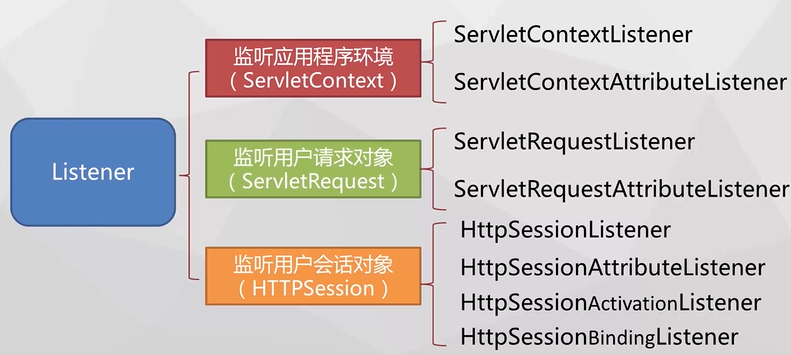
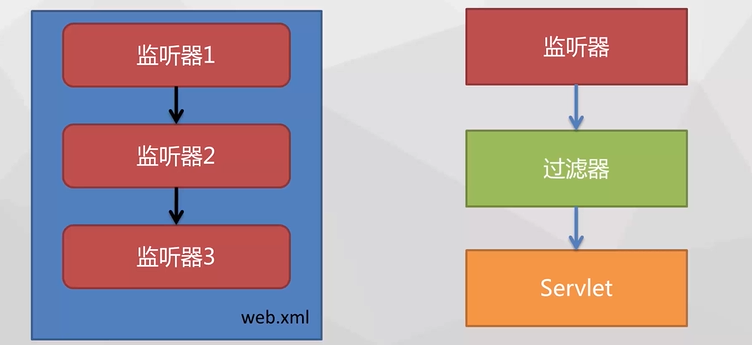
# Servlet转发与重定向

1. 请求转发
   1. Forward
   2. 将当前的request和respond对象交给指定的web组件处理
   3. 一次请求一次响应
   4. url不会变
   5. 转发对象
      1. ReqeustDispatcher
      2. 通过HttpServletRequest获取
         1. forward转发到forwardexample
      3. 通过ServletContext获取
2. 请求重定向
   1. sendRedirect
   2. 通过response对象发送给浏览器一个新url地址，让其重新请求
   3. 两次请求，两次响应
   4. url会变化

# Servlet filter配置

1. 过滤器
   1. 过滤请求与响应
   2. 自定义过滤规则
   3. 用于对用户请求进行预处理，和对请求响应进行后的处理的web应用组件
2. 应用场景
   1. 用户认证
   2. 编解码处理
   3. 数据压缩处理
3. 生命周期
   1. Web.xml配置
   2. 未登录用户跳转的实现
4. Filter链——责任链模式

# Sevlet listener配置

1. 监听器
   1. 观察者模式
2. 监听事件发生，在事件发生后能够做出相应处理的web应用组件
3. 监听器分类
4. 监听器的应用场景
   1. 应用统计
   2. 任务触发
   3. 业务需求
5. 监听器的启动顺序
6. web.xml配置比较简单

# Servlet并发处理

1. 单实例
2. 多线程
3. 线程不安全
4. 如何线程安全
   1. 变量的线程安全
      1. 参数变量本地化
      2. 使用同步块synchronized
   2. 属性的线程安全
      1. ServletContext线程不安全
      2. HttpSession理论上线程安全
      3. ServletRequest线程安全
   3. 避免在servlet中创建线程
   4. 多个Servlet访问外部对象加锁

# JSP简介

1. JSP是对Servlet的简化
2. JSP内置了诸如HttpRequest，ServletContext等对象
3. TOMCAT中的JSP容器会将JSP转化为Servlet的.java文件，再翻译为.class文件，最后执行