Servlet

1. Servlet的配置，每个部分的作用：

<servlet>

<description>描述内容</description>\* <display-name> 显示名</display-name>\*

<servlet-name> </servlet-name>

<servlet-class> </servlet-class>或<jsp-file> </jsp-file>

<init-param> 0或1个元素， 时，可以用方法引用该参数。

<param-name> </param-name> <param-value> </param-value>

</init-param>

<load-on-startup>一个整形值</load-on-startup> 0或1个元素，优先级，当值为负或未定义，servlet容器将

<security-role-ref> 可以设定多个角色,对资源访问的身份认证, 也可以用过滤器技术。

<role-name> </role-name>

<role-link> </role-link>

</security-role-ref>

</servlet>

1. 过滤器是小型的 Web 组件，若服务器（如Tomcat）中有过滤器部署，则对于从客户端发送过来的请求，服务器首先 ，然后，或让 或 处理，或直接 （假如安全检查没通过）。
2. 如果系统中设置了多个过滤器（一般情况下，一个过滤器完成一项特定任务），则一组过滤器会形成一个 ，客户请求会 。
3. Java中的Filter并不是一个标准的 ，它不能 ，也不能 。主要用于对 进行预处理，也可以对 进行后处理，是个典型的处理链。
4. 过滤器在HttpServletRequest到达 之前，拦截客户的 。
5. 过滤器根据需要 HttpServletRequest，也可 HttpServletRequest头和数据。
6. 过滤器可以根据 ，决定页面迁移（返回、迁移到其它页面或到放行）
7. 常用的过滤器：

用户授权的Filter：Filter负责检查 ，根据请求过滤 。

日志Filter： 某些特殊的用户请求。

负责解码的Filter：包括对非标准编码的请求 。

1. 实现了javax.servlet.Filter接口的类被称为过滤器类，主要方法：

init（FileterConfig fileterconfig）：初始化，由 （tomcat）调用，

doFilter(ServletRequest request，ServletResponse response，FilterChain chain)

这个方法完成实际的 操作。servlet容器将先 。FilterChain参数用于 。

destroy()：servlet容器在 调用该方法。

1. 过滤器的生命周期：
2. 启动服务器时加载过滤器的实例，并调用 方法来初始化实例；
3. 每一次请求时都只调用 方法进行处理；
4. 停止服务器时调用 方法，销毁实例。
5. 过滤器配置文件各部分值

通过web.xml

<filter>

<filter-name> </filter-name>

<filter-class> </filter-class>

<init-param>

<param-name> </param-name>

<param-value> </param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name> </filter-name>

<url-pattern> </url-pattern>

</filter-mapping>

过滤范围：1、单个资源,如

2、指定目录：如全部：

1. Servlet3.0标准以后，过滤器也提供了注解方法，方法、作用与意义类同类Servlet的注解。如：

@WebFilter(filterName="SecurityCheck",value="/securitycheck/\*")

public class SecurityCheck implements Filter {。。。}

filterName：相当于

value：相当于

1. 若有多个过滤器，服务器会按照项目中过滤器定义的先后循序组装成一条链，然后依次序执行其中的 。如：有两个过滤器，EncodingFilter负责设置编码，SecurityFilter负责控制权限。
2. 安全过滤器的开发的设计思路：把 ，如把个人中心、支付中心等页面放在securitycheck/下。
3. 安全过滤器核心代码

@WebFilter(filterName="SecurityCheck",value="/securitycheck/\*")

public class SecurityCheck implements Filter {

。。。。

public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {

HttpServletRequest request =(HttpServletRequest)req;

HttpServletResponse response =(HttpServletResponse)res;

HttpSession session = request.getSession();

String username = (String) session.getAttribute("name");

//条件成立时，不需要过滤,进入下个过滤或到servlet

if(username!=null) {

//System.out.println("filter suc");

chain.doFilter(req,res); }

else response.sendRedirect("/Filter/bookmain.html");

}

HttpServletRequest和ServletRequest都是接口，前者 后者，HttpServlet Request 比ServletRequest多了一些针对于Http的方法,若需要获得session等，需要 。

1. 监听器(Listener)用于 ，它是Servlet规范中的 ，也需要在web.xml文件中

才可以使用(除了两个例外)。它监听着 、 和 等对象的创建与销毁事件，以及这些域对象中 。当这些事件发生的时候，对应的监听器会立刻做出反应。

1. 当ServletContext作为监听器对象，ServletContextListenter和ServeltContextAttributeListeneter是 ，ServeletContextEvent和ServletContextAttributeEvevnt是 。
2. 当HttpSession作为监听器对象，HttpSessionListener、HttpSessionActivationListener、HttpSessionAttributeListener和HttpSessionBindingListener是 ，HttpSessionEvent和HttpSessionBindingEvent是 。
3. 当ServletRequest作为监听器对象，ServletRequestListener和ServletRequestAttributeL istener是 ，ServletRequestEvent和ServletRequestAttributeEvent是 。
4. 当有被监听对象被servlet容器创建，或者在生命周期结束被销毁的时候，3种监听器负责监听。

ServletContextListener：ServletContext对象在 的时候被创建,当ServletContext对象被创建时，激发 方法；Servlet Context对象在 时被销毁，当ServletContext对象被销毁时激发 方法。

HttpSessionListener：session在 被创建，当session被创建时，激发 方法；session 或调用 时被销毁，当session被销毁时，激发 方法。

ServletRequestListener：request对象在 创建，当request对象被创建时，激发

方法；request对象在 被销毁，当request对象被销毁时，会激发 方法。

1. 监听对象的属性修改事件的监听器

ServletContextAttributeListener：当servletContext对象的属性有上述操作时，会分别激发 ， ， 三个方法。

HttpsessionAttributeListener：当session对象的属性有上述操作时，会分别激发和上述方法名一样方法。

ServletRequestAttributeListener：当request对象的属性有上述操作时，同样激发和上述方法名一样的方法。

虽然在对象属性被操作时，三种监听器对应的实现类的方法名是一模一样的，但是其中监听的事件名是完全不一样的。

建议答案

1. servlet名、servlet类名、JSP文件名、初始化sevlet、参数名、参数值、延迟装载、角色名、角色的一个引用
2. 让过滤器先执行、客户请求的目的地页面、servlet、进行页面转发
3. 过滤链、在过滤链中逐步过滤执行
4. Servlet、处理用户请求、对客户端生成响应、HttpServletRequest、HttpServletResponse
5. Servlet、HttpServletRequest
6. 检查、修改
7. servlet上下文数据
8. 用户请求、用户非法请求、详细记录、解码
9. servlet容器、过滤、调用过滤器的doFilter方法、访问后续过滤器、销毁过滤器实例前
10. init()、doFilter()、destroy()
11. 过滤器名称、过滤器对应的类、参数名称1、参数值1、过滤器名称、过滤范围、XXX.jsp, xxx.html ,或servlet名称、/\*
12. <filter-name>过滤器名称</filter-name>、<url-pattern>过滤范围</url-pattern>
13. doFilter()方法
14. 需要过滤的服务器资源集中放在一个目录中
15. 继承自、转换
16. 监听Java Web程序中的各类事件、特殊类、注册配置、Servlet Context、HttpSession、ServletRequest、属性发生修改的事件
17. 监听接口、监听事件
18. 监听接口、监听事件
19. 监听接口、监听事件
20. 在web服务器被启动、contextInitialized()、web服务关闭、context Dtroyed()

浏览器和服务器的会话开始时、sessionCreated()、失效时、session .invalidate()、sessionDestroyed()

每次请求开始时、requestInitialized()、每次访问结束后、requestDtroyed()

1. attributeAdded、attributeReplaced、attributeRemoved