

# Java 最常见的 208 道面试题：第六模块答案

Java团长 2019-03-17

## Java Web

### 64. jsp 和 servlet 有什么区别？

- jsp经编译后就变成了Servlet. (JSP的本质就是Servlet, JVM只能识别java的类, 不能识别JSP的代码, Web容器将JSP的代码编译成JVM能够识别的java类)
- jsp更擅长表现于页面显示, servlet更擅长于逻辑控制。
- Servlet中没有内置对象, Jsp中的内置对象都是必须通过HttpServletRequest对象, HttpServletResponse对象以及HttpServlet对象得到。
- Jsp是Servlet的一种简化, 使用Jsp只需要完成程序员需要输出到客户端的内容, Jsp中的Java脚本如何镶嵌到一个类中, 由Jsp容器完成。而Servlet则是个完整的Java类, 这个类的Service方法用于生成对客户端的响应。

### 65. jsp 有哪些内置对象？作用分别是什么？

JSP有9个内置对象：

- request：封装客户端的请求，其中包含来自GET或POST请求的参数；
- response：封装服务器对客户端的响应；
- pageContext：通过该对象可以获取其他对象；
- session：封装用户会话的对象；
- application：封装服务器运行环境的对象；
- out：输出服务器响应的输出流对象；

- config: Web应用的配置对象;
- page: JSP页面本身 (相当于Java程序中的this) ;
- exception: 封装页面抛出异常的对象。

## 66. 说一下 jsp 的 4 种作用域?

JSP中的四种作用域包括page、request、session和application, 具体来说:

- **page**代表与一个页面相关的对象和属性。
- **request**代表与Web客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面, 涉及多个Web组件; 需要在页面显示的临时数据可以置于此作用域。
- **session**代表与某个用户与服务器建立的一次会话相关的对象和属性。跟某个用户相关的数据应该放在用户自己的session中。
- **application**代表与整个Web应用程序相关的对象和属性, 它实质上是跨越整个Web应用程序, 包括多个页面、请求和会话的一个全局作用域。

## 67. session 和 cookie 有什么区别?

- 由于HTTP协议是无状态的协议, 所以服务端需要记录用户的状态时, 就需要用某种机制来识具体的用户, 这个机制就是Session.典型的场景比如购物车, 当你点击下单按钮时, 由于HTTP协议无状态, 所以并不知道是哪个用户操作的, 所以服务端要为特定的用户创建了特定的Session, 用于标识这个用户, 并且跟踪用户, 这样才知道购物车里面有几本书。这个Session是保存在服务端的, 有一个唯一标识。在服务端保存Session的方法很多, 内存、数据库、文件都有。集群的时候也要考虑Session的转移, 在大型的网站, 一般会有专门的Session服务器集群, 用来保存用户会话, 这个时候 Session 信息都是放在内存的, 使用一些缓存服务比如Memcached之类的来放 Session。
- 思考一下服务端如何识别特定的客户? 这个时候Cookie就登场了。每次HTTP请求的时候, 客户端都会发送相应的Cookie信息到服务端。实际上大多数的应用都是用 Cookie 来实现Session跟踪的, 第一次创建Session的时候, 服务端会在HTTP协议中告诉客户端, 需要在 Cookie 里面记录一个Session ID, 以后每次请求把这个会话ID发送到服务

器，我就知道你是谁了。有人问，如果客户端的浏览器禁用了 Cookie 怎么办？一般这种情况下，会使用一种叫做 URL 重写的技术来进行会话跟踪，即每次 HTTP 交互，URL 后面都会被附加上一个诸如 `sid=xxxxx` 这样的参数，服务端据此来识别用户。

- Cookie 其实还可以用在一些方便用户的场景下，设想你某次登陆过一个网站，下次登录的时候不想再次输入账号了，怎么办？这个信息可以写到 Cookie 里面，访问网站的时候，网站页面的脚本可以读取这个信息，就自动帮你把用户名给填了，能够方便一下用户。这也是 Cookie 名称的由来，给用户的一点甜头。所以，总结一下：Session 是在服务端保存的一个数据结构，用来跟踪用户的状态，这个数据可以保存在集群、数据库、文件中；Cookie 是客户端保存用户信息的一种机制，用来记录用户的一些信息，也是实现 Session 的一种方式。

## 68. 说一下 session 的工作原理？

其实 session 是一个存在服务器上的类似于一个散列表的文件。里面存有我们需要的信息，在我们需要用的时候可以从里面取出来。类似于一个大号的 map 吧，里面的键存储的是用户的 sessionid，用户向服务器发送请求的时候会带上这个 sessionid。这时就可以从中取出对应的值了。

## 69. 如果客户端禁止 cookie 能实现 session 还能用吗？

Cookie 与 Session，一般认为是两个独立的东西，Session 采用的是在服务器端保持状态的方案，而 Cookie 采用的是在客户端保持状态的方案。但为什么禁用 Cookie 就不能得到 Session 呢？因为 Session 是用 Session ID 来确定当前对话所对应的服务器 Session，而 Session ID 是通过 Cookie 来传递的，禁用 Cookie 相当于失去了 Session ID，也就得不到 Session 了。

假定用户关闭 Cookie 的情况下使用 Session，其实现途径有以下几种：

- a. 设置 php.ini 配置文件中的 `"session.use_trans_sid = 1"`，或者编译时打开打开了 `"--enable-trans-sid"` 选项，让 PHP 自动跨页传递 Session ID。
- b. 手动通过 URL 传值、隐藏表单传递 Session ID。
- c. 用文件、数据库等形式保存 Session ID，在跨页过程中手动调用。

## 70. spring mvc 和 struts 的区别是什么？

- 拦截机制的不同

Struts2是类级别的拦截，每次请求就会创建一个Action，和Spring整合时Struts2的ActionBean注入作用域是原型模式prototype，然后通过setter，getter吧request数据注入到属性。Struts2中，一个Action对应一个request，response上下文，在接收参数时，可以通过属性接收，这说明属性参数是让多个方法共享的。Struts2中Action的一个方法可以对应一个url，而其类属性却被所有方法共享，这也就无法用注解或其他方式标识其所属方法了，只能设计为多例。

SpringMVC是方法级别的拦截，一个方法对应一个Request上下文，所以方法直接基本上是独立的，独享request，response数据。而每个方法同时又何一个url对应，参数的传递是直接注入到方法中的，是方法所独有的。处理结果通过ModeMap返回给框架。在Spring整合时，SpringMVC的Controller Bean默认单例模式Singleton，所以默认对所有的请求，只会创建一个Controller，有应为没有共享的属性，所以是线程安全的，如果要改变默认的作用域，需要添加@Scope注解修改。

Struts2有自己的拦截Interceptor机制，SpringMVC这是用的是独立的Aop方式，这样导致Struts2的配置文件量还是比SpringMVC大。

- 底层框架的不同

Struts2采用Filter（StrutsPrepareAndExecuteFilter）实现，SpringMVC（DispatcherServlet）则采用Servlet实现。Filter在容器启动之后即初始化；服务停止以后坠毁，晚于Servlet。Servlet在是在调用时初始化，先于Filter调用，服务停止后销毁。

- 性能方面

Struts2是类级别的拦截，每次请求对应实例一个新的Action，需要加载所有的属性值注入，SpringMVC实现了零配置，由于SpringMVC基于方法的拦截，有加载一次单例模式bean注入。所以，SpringMVC开发效率和性能高于Struts2。

- 配置方面

spring MVC和Spring是无缝的。从这个项目的管理和安全上也比Struts2高。

## 71. 如何避免 sql 注入？

- a. PreparedStatement（简单又有效的方法）
- b. 使用正则表达式过滤传入的参数
- c. 字符串过滤
- d. JSP中调用该函数检查是否包函非法字符
- e. JSP页面判断代码

## 72. 什么是 XSS 攻击，如何避免？

XSS攻击又称CSS,全称Cross Site Script（跨站脚本攻击），其原理是攻击者向有XSS漏洞的网站中输入恶意的 HTML 代码，当用户浏览该网站时，这段 HTML 代码会自动执行，从而达到攻击的目的。XSS 攻击类似于 SQL 注入攻击，SQL注入攻击中以SQL语句作为用户输入，从而达到查询/修改/删除数据的目的，而在xss攻击中，通过插入恶意脚本，实现对用户浏览器的控制，获取用户的一些信息。XSS是 Web 程序中常见的漏洞，XSS 属于被动式且用于客户端的攻击方式。

XSS防范的总体思路是：对输入(和URL参数)进行过滤，对输出进行编码。

## 73. 什么是 CSRF 攻击，如何避免？

CSRF (Cross-site request forgery) 也被称为 one-click attack或者 session riding, 中文全称是叫**跨站请求伪造**。一般来说, 攻击者通过伪造用户的浏览器的请求, 向访问一个用户自己曾经认证访问过的网站发送出去, 使目标网站接收并误以为是用户的真实操作而去执行命令。常用于盗取账号、转账、发送虚假消息等。攻击者利用网站对请求的验证漏洞而实现这样的攻击行为, 网站能够确认请求来源于用户的浏览器, 却不能验证请求是否源于用户的真实意愿下的操作行为。

如何避免:

### 1. 验证 HTTP Referer 字段

HTTP头中的Referer字段记录了该 HTTP 请求的来源地址。在通常情况下, 访问一个安全受限页面的请求来自于同一个网站, 而如果黑客要对其实施 CSRF 攻击, 他一般只能在他自己的网站构造请求。因此, 可以通过验证Referer值来防御CSRF 攻击。

### 2. 使用验证码

关键操作页面加上验证码, 后台收到请求后通过判断验证码可以防御CSRF。但这种方法对用户不太友好。

### 3. 在请求地址中添加token并验证

CSRF 攻击之所以能够成功, 是因为黑客可以完全伪造用户的请求, 该请求中所有的用户验证信息都是存在于cookie中, 因此黑客可以在不知道这些验证信息的情况下直接利用用户自己的cookie 来通过安全验证。要抵御 CSRF, 关键在于在请求中放入黑客所不能伪造的信息, 并且该信息不存在于 cookie 之中。可以在 HTTP 请求中以参数的形式加入一个随机产生的 token, 并在服务器端建立一个拦截器来验证这个 token, 如果请求中没有token或者 token 内容不正确, 则认为可能是 CSRF 攻击而拒绝该请求。这种方法要比检查 Referer 要安全一些, token 可以在用户登陆后产生并放于session之中, 然后在每次请求时把token 从 session 中拿出, 与请求中的 token 进行比对, 但这种方法的难点在于如何把 token 以参数的形式加入请求。

对于 GET 请求，token 将附在请求地址之后，这样 URL 就变成 `http://url?csrftoken=tokenvalue`。

而对于 POST 请求来说，要在 form 的最后加上 `<input type="hidden" name="csrftoken" value="tokenvalue"/>`，这样就把 token 以参数的形式加入请求了。

#### 4. 在 HTTP 头中自定义属性并验证

这种方法也是使用 token 并进行验证，和上一种方法不同的是，这里并不是把 token 以参数的形式置于 HTTP 请求之中，而是把它放到 HTTP 头中自定义的属性里。通过 `XMLHttpRequest` 这个类，可以一次性给所有该类请求加上 `csrftoken` 这个 HTTP 头属性，并把 token 值放入其中。这样解决了上种方法在请求中加入 token 的不便，同时，通过 `XMLHttpRequest` 请求的地址不会被记录到浏览器的地址栏，也不用担心 token 会透过 Referer 泄露到其他网站中去。

(完)