码农求职小助手：JavaWeb高频面试题

笔记本：	5-JavaWeb		
创建时间：	2019/9/7 21:37	**更新时间：**	2019/9/7 21:38
作者：	pc941206@163.com		

更多资料请关注微信公众号：**码农求职小助手**

一、JSP

1、Jsp 和 servlet 有什么区别?

1、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 JV M能够识别的 Java 类）；

2、Jsp 更擅长表现于页面显示，servlet 更擅长于逻辑控制；

3、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；

4、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。

2、Jsp 有哪些内置对象？作用分别是什么?

- Jsp 有 9 个内置对象：

1、request：封装客户端的请求，其中包含来自 GET 或 POST 请求的参数；

2、response：封装服务器对客户端的响应；

3、pageContext：通过该对象可以获取其他对象；

4、session：封装用户会话的对象；

5、application：封装服务器运行环境的对象；

6、out：输出服务器响应的输出流对象；

7、config：Web 应用的配置对象；

8、page：Jsp 页面本身（相当于 Java 程序中的 this）；

9、exception：封装页面抛出异常的对象。

3、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：

1、page 代表与一个页面相关的对象和属性；

2、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；

3、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；

4、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。

二、Servlet

1、Servlet 的生命周期

Servlet 生命周期可分为 5 个步骤：

1、**加载 Servlet**：当 Tomcat 第一次访问 Servlet 的时候，Tomcat 会负责创建 Servlet 的实例；

2、**初始化**：当 Servlet 被实例化后，Tomcat 会调用 init() 方法初始化这个对象；

3、**处理服务**：当浏览器访问 Servlet 的时候，Servlet 会调用 service() 方法处理请求；

4、**销毁**：当 Tomcat 关闭时或者检测到 Servlet 要从 Tomcat 删除的时候会自动调用 destroy() 方法，让该实例释放掉所占的资源。一个 Servlet 如果长时间不被使用的话，也会被 Tomcat 自动销毁；

5、**卸载**：当 Servlet 调用完 destroy() 方法后，等待垃圾回收。如果有需要再次使用这个 Servlet，会重新调用 init() 方法进行初始化操作。

简单总结：只要访问 Servlet，service() 就会被调用。init() 只有第一次访问 Servlet 的时候才会被调用。destroy() 只有在 Tomcat 关闭的时候才会被调用。

2、Servlet 相关的 API：

- **doGet 与 doPost 方法的两个参数是什么?**

1、**HttpServletRequest**：封装了与请求相关的信息；

2、**HttpServletResponse**：封装了与响应相关的信息。

- **获取页面的元素的值有几种方式，分别说一下?**

1、request.getParameter() 返回客户端的请求参数的值；

2、request.getParameterNames() 返回所有可用属性名的枚举；

3、request.getParameterValues() 返回包含参数的所有值的数组。

3、forward 和 redirect 的区别?

实际发生位置不同，地址栏不同。

- **转发**

转发是发生在服务器的，是由服务器进行跳转的。细心的朋友会发现，在转发的时候，浏览器的地址栏是没有发生变化的，在我访问 Servlet111 的时候，即使跳转到了 Servlet222 的页面，浏览器的地址还是 Servlet111 的。也就是说浏览器是不知道该跳转的动作，**转发是对浏览器透明的。**实现转发只是一次的 http 请求，一次转发中 request 和 response 对象都是同一个。这也解释了，为什么可以使用 request 作为域对象进行 Servlet 之间的通讯。

- **重定向**

重定向是发生在浏览器上的。**重定向是由浏览器进行跳转的**，进行重定向跳转的时候，**浏览器的地址会发生变化的。**实现重定向的原理是由 response 的状态码和 Location 头组合而实现的。**这时由浏览器进行的页面跳转**实现重定向**会发出两个 http 请求**，**request 域对象是无效的，因为它不是同一个 request 对象**

- **转发和重定向的区别**

1、能够去往的 URL 的范围不一样：转发是服务器跳转，只能去往当前 web 应用的资源。而重定向是浏览器跳转，可以去往任何的资源；

2、传递数据的类型不一样：转发的 request 对象可以传递各种类型的数据，包括对象。而重定向只能传递字符串；

3、跳转的时间不同：转发时，执行到跳转语句时就会立刻跳转。而重定向时，整个页面执行完之后才执行跳转。

4、Tomcat 容器是如何创建 Servlet 类实例？用到了什么原理?

1、当容器启动时，会读取在 webapps 目录下所有的 web 应用中的 web.xml 文件，然后对 xml 文件进行解析，并读取 Servlet 注册信息。然后，将每个应用中注册的 servlet 类都进行加载，并通过反射的方式实例化。（有时候也是在第一次请求时实例化）

2、在 Servlet 注册时加上 <load-on-startup>1</load-on-startup> 如果为正数，则在一开始就实例化，如果不写或为负数，则第一次请求实例化。

5、Servlet 安全性问题

由于 Servlet 是单例的，当多个用户访问 Servlet 的时候，服务器会为每个用户创建一个线程。当多个用户并发访问 Servlet 共享资源的时候就会出现线程安全问题。

- **原则：**

1、如果一个变量需要多个用户共享，则应当在访问该变量的时候，加同步机制 synchronized (对象){}；

2、如果一个变量不需要共享，则直接在 doGet() 或者 doPost()中定义。这样不会存在线程安全问题。

三、Cookie 与 Session

1、Cookie 和 Session 有什么区别?

1、**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。

2、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。

3、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。**

- **Cookie 和 Session 有什么区别?**

1、从存储方式上比较：

Cookie 只能存储字符串，如果要存储非ASCII字符串还要对其编码。

Session可以存储任何类型的数据，可以把 Session 看成是一个容器。

2、从隐私安全上比较：

Cookie 存储在浏览器中，对客户端是可见的。信息容易泄露出去。如果使用 Cookie，最好将 Cookie 加密。

Session 存储在服务器上，对客户端是透明的。不存在敏感信息泄露问题。

3、从有效期上比较

Cookie 保存在硬盘中，只需要设置 maxAge 属性为比较大的正整数，即使关闭浏览器，Cookie 还是存在的。

Session 的保存在服务器中，设置 maxInactiveInterval 属性值来确定 Session 的有效期。并且 Session 依赖于名为 JSESSIONID 的 Cookie，该 Cookie 默认的 maxAge 属性为 -1。如果关闭了浏览器，该 Session 虽然没有从服务器中消亡，但也就失效了。

4、从对服务器的负担比较

Session 是保存在服务器的，每个用户都会产生一个 Session，如果是并发访问的用户非常多，是不能使用 Session 的，Session 会消耗大量的内存。

Cookie是保存在客户端的。不占用服务器的资源。像 baidu、Sina 这样的大型网站，一般都是使用 Cookie 来进行会话跟踪。

5、从浏览器的支持上比较

如果浏览器禁用了 Cookie，那么 Cookie 是无用的了。

如果浏览器禁用了 Session，Session 可以通过 URL 地址重写来进行会话跟踪。

6、从跨域名上比较

Cookie 可以设置 domain 属性来实现跨域名。

Session 只在当前的域名内有效，不可夸域名。

2、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格的文件。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。

3、如果客户端禁止 Cookie ， 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 sessionId 来确定当前对话所对应的服务器 Session，而 sessionId 是通过Cookie 来传递的，禁用 Cookie 相当于失去了sessionId，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：

1、设置 php.ini 配置文件中的“session.use_trans_sid = 1”，或者编译时打开了“--enable-trans-sid”选项，让 PHP 自动跨页传递 sessionId；

2、手动通过 URL 传值、隐藏表单传递 sessionId；

3、用文件、数据库等形式保存 sessionId，在跨页过程中手动调用。

四、JDBC

详解：https://segmentfault.com/a/1190000013312766

1、JDBC 操作数据库的步骤？

1、注册数据库驱动；

2、建立数据库连接；

3、创建一个 Statement；

4、执行 SQL 语句；

5、处理结果集；

6、关闭数据库连接。

```
Connection connection = null;
Statement statement = null;
ResultSet resultSet = null;
```

```
try {
    /*
    * 加载驱动有两种方式
    * 1：会导致驱动注册两次，过度依赖于mysql的api，脱离的mysql的开发包，程序则无法编译
    * 2：驱动只会加载一次，不需要依赖具体的驱动，灵活性高
    *
    * 我们一般都是使用第二种方式
    * */

    // 1.
    // DriverManager.registerDriver(new com.mysql.jdbc.Driver());

    // 2.
    Class.forName("com.mysql.jdbc.Driver");

    // 获取与数据库连接的对象-Connetcion
    connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/zhongfucheng",
"root", "root");

    // 获取执行sql语句的statement对象
    statement = connection.createStatement();

    // 执行sql语句，拿到结果集
    resultSet = statement.executeQuery("SELECT * FROM users");

    // 遍历结果集，得到数据
    while (resultSet.next()) {
        System.out.println(resultSet.getString(1));
        System.out.println(resultSet.getString(2));
    }
} catch (SQLException e) {
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
} finally {
    // 关闭资源，后调用的先关闭，关闭之前，要判断对象是否存在
    if (resultSet != null) {
        try {
            resultSet.close();
        } catch (SQLException e) {
            e.printStackTrace();
```

```
            }
        }
        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

2、JDBC 中的 Statement 和PreparedStatement，CallableStatement 的区别?

1、PreparedStatement 是预编译的 SQL 语句，效率高于 Statement；

2、PreparedStatement 支持？操作符，相对于 Statement 更加灵活；

3、PreparedStatement 可以防止 SQL 注入，安全性高于 Statement；

4、CallableStatement 适用于执行存储过程。