JavaWEB 篇

一、JDBC 技术

1-1 说下原生 jdbc 操作数据库流程

第一步：Class.forName()加载数据库连接驱动；

第二步：DriverManager.getConnection()获取数据连接对象;

第三步：根据 SQL 获取 sql 会话对象，有 2 种方式 Statement、PreparedStatement ;

第四步：执行 SQL 处理结果集，执行 SQL 前如果有参数值就设置参数值 setXXX();

第五步：关闭结果集、关闭会话、关闭连接。

1-2 什么要使用 PreparedStatement?

1、 PreparedStatement 接口继承 Statement， PreparedStatement 实例包含已编译的 SQL 语句，所以其执行速度要快于 Statement 对象。

2、作为 Statement 的子类， PreparedStatement 继承了 Statement 的所有功能。三种方法 execute、 executeQuery 和 executeUpdate 已被更改以使之不再需要参数

3、在 JDBC 应用中,在任何时候都不要使用 Statement，原因如下：

一、代码的可读性和可维护性.Statement 需要不断地拼接，而 PreparedStatement 不会。

二、PreparedStatement 尽最大可能提高性能.DB 有缓存机制，相同的预编译语句再次被调用不会再次需要编译。

三、最重要的一点是极大地提高了安全性.Statement 容易被 SQL 注入，而 PreparedStatementc 传入的内容不会和 sql 语句发生任何匹配关系。

1-3：说说事务的概念，在 JDBC 编程中处理事务的步骤。

1 事务是作为单个逻辑工作单元执行的一系列操作。

2，一个逻辑工作单元必须有四个属性，称为原子性、一致性、隔离性和持久性 (ACID) 属性，只有这样才能成为一个事务

事务处理步骤：

3，conn.setAutoComit(false);设置提交方式为手工提交

4，conn.commit()提交事务

5，出现异常，回滚 conn.rollback();

1-4 JDBC 的脏读是什么？哪种数据库隔离级别能防止脏读?

当我们使用事务时，有可能会出现这样的情况，有一行数据刚更新，与此同时另一个查询读到了这个刚更新的值。这样就导致了脏读，因为更新的数据还没有进行持久化，更新这行数据的业务可能会进行回滚，这样这个数据就是无效的。数据库的 *TRANSACTIONREADCOMMITTED*, *TRANSACTIONREPEATABLEREAD*，和 *TRANSACTION_SERIALIZABLE*隔离级别可以防止脏读。

1-5 什么是幻读，哪种隔离级别可以防止幻读?

幻读是指一个事务多次执行一条查询返回的却是不同的值。假设一个事务正根据某个条件进行数据查询，然后另一个事务插入了一行满足这个查询条件的数据。之后这个事务再次执行了这条查询，返回的结果集中会包含刚插入的那条新数据。这行新数据被称为幻行，而这种现象就叫做幻读。

只有 *TRANSACTION_SERIALIZABLE* 隔离级别才能防止产生幻读。

二、关系数据库中连接池的机制是什么?

前提：为数据库连接建立一个缓冲池。

1：从连接池获取或创建可用连接

2：使用完毕之后，把连接返回给连接池

3：在系统关闭前，断开所有连接并释放连接占用的系统资源

4：能够处理无效连接，限制连接池中的连接总数不低于或者不超过某个限定值。

其中有几个概念需要大家理解：

最小连接数是连接池一直保持的数据连接。如果应用程序对数据库连接的使用量不大，将会有大量的数据库连接资源被浪费掉。

最大连接数是连接池能申请的最大连接数。如果数据连接请求超过此数，后面的数据连接请求将被加入到等待队列中，这会影响之后的数据库操作。

如果最小连接数与最大连接数相差太大，那么，最先的连接请求将会获利，之后超过最小连接数量的连接请求等价于建立一个新的数据库连接。不过，这些大于最小连接数的数据库连接在使用完不会马上被释放，它将被放到连接池中等待重复使用或是空闲超时后被释放。

上面的解释，可以这样理解：数据库池连接数量一直保持一个不少于最小连接数的数量，当数量不够时，数据库会创建一些连接，直到一个最大连接数，之后连接数据库就会等待。

2-1 数据库连接池的原理。为什么要使用连接池?

1，数据库连接是一件费时的操作，连接池可以使多个操作共享一个连接。

2，数据库连接池的基本思想就是为数据库连接建立一个“缓冲池”。预先在缓冲池中放入一定数量的连接，当需要建立数据库连接时，只需从“缓冲池”中取出一个，使用完毕之后再放回去。我们可以通过设定连接池最大连接数来防止系统无尽的与数据库连接。更为重要的是我们可以通过连接池的管理机制监视数据库的连接的数量、使用情况，为系统开发，测试及性能调整提供依据。

3，使用连接池是为了提高对数据库连接资源的管理

三、 Http 协议

3-1 Http 的长连接和短连接

HTTP 协议有 HTTP/1.0 版本和 HTTP/1.1 版本。HTTP1.1 默认保持长连接（HTTP persistent connection，也翻译为持久连接），数据传输完成了保持 TCP 连接不断开（不发 RST 包、不四次握手），等待在同域名下继续用这个通道传输数据；相反的就是短连接。

在 HTTP/1.0 中，默认使用的是短连接。也就是说，浏览器和服务器每进行一次 HTTP 操作，就建立一次连接，任务结束就中断连接。从 HTTP/1.1 起，默认使用的是长连接，用以保持连接特性。

3-2 Http 常见的状态码有哪些?

200 OK //客户端请求成功

301 Moved Permanently（永久移除），请求的 URL 已移走。Response 中应该包含一个 Location URL, 说明资源现在所处的位置

302 found 重定向

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

401 Unauthorized //请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，eg：输入了错误的 URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

3-3 GET 和 POST 的区别?

从表面现像上面看 GET 和 POST 的区别：

1. GET 请求的数据会附在 URL 之后（就是把数据放置在 HTTP 协议头中），以?分割 URL 和传输数据，参数之间以&相连，如：login.action?name=zhagnsan&password=123456。POST 把提交的数据则放置在是 HTTP 包的包体中。

2. GET 方式提交的数据最多只能是 1024 字节，理论上 POST 没有限制，可传较大量的数据。其实这样说是错误的，不准确的：

“GET 方式提交的数据最多只能是 1024 字节"，因为 GET 是通过 URL 提交数据，那么 GET 可提交的数据量就跟 URL 的长度有直接关系了。而实际上，URL 不存在参数上限的问题，HTTP 协议规范没有对 URL 长度进行限制。这个限制是特定的浏览器及服务器对它的限制。IE 对 URL 长度的限制是 2083 字节(2K+35)。对于其他浏览器，如 Netscape、FireFox 等，理论上没有长度限制，其限制取决于操作系统的支持。

3.POST 的安全性要比 GET 的安全性高。注意：这里所说的安全性和上面 GET 提到的“安全”不是同个概念。上面“安全”的含义仅仅是不作数据修改，而这里安全的含义是真正的 Security 的含义，比如：通过 GET 提交数据，用户名和密码将明文出现在 URL 上，因为(1)登录页面有

可能被浏览器缓存，(2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了，除此之外，使用 GET 提交数据还可能会造成 Cross-site request forgery 攻击。

Get 是向服务器发索取数据的一种请求，而 Post 是向服务器提交数据的一种请求，在 FORM（表单）中，Method 默认为"GET"，实质上，GET 和 POST 只是发送机制不同，并不是一个取一个发！

3-4 Http 中重定向和请求转发的区别?

本质区别：转发是服务器行为，重定向是客户端行为。

重定向特点：两次请求，浏览器地址发生变化，可以访问自己 web 之外的资源，传输的数据会丢失。

请求转发特点：一次强求，浏览器地址不变，访问的是自己本身的 web 资源，传输的数据不会丢失。

3-4 Http 与 Https 协议的区别

HTTP 协议传输的数据都是未加密的，也就是明文的，因此使用 HTTP 协议传输隐私信息非常不安全，为了保证这些隐私数据能加密传输，于是网景公司设计了 SSL（Secure Sockets Layer）协议用于对 HTTP 协议传输的数据进行加密，从而就诞生了 HTTPS。

简单来说，HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，要比 http 协议安全。

HTTPS 和 HTTP 的区别主要如下：

1、https 协议需要到 ca 申请证书，一般免费证书较少，因而需要一定费用。

2、http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议。

3、http 和 https 使用的是完全不同的连接方式，用的端口也不一样，前者是 80，后者是 443。

4、http 的连接很简单，是无状态的；HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全。

HTTPS 的工作原理

我们都知道 HTTPS 能够加密信息，以免敏感信息被第三方获取，所以很多银行网站或电子邮箱等等安全级别较高的服务都会采用 HTTPS 协议。

1、客户端发起 HTTPS 请求

这个没什么好说的，就是用户在浏览器里输入一个 https 网址，然后连接到 server 的 443 端口。

2、服务端的配置

采用 HTTPS 协议的服务器必须要有一套数字证书，可以自己制作，也可以向组织申请，区别就是自己颁发的证书需要客户端验证通过，才可以继续访问，而使用受信任的公司申请的证书则不会弹出提示页面(startssl 就是个不错的选择，有 1 年的免费服务)。

这套证书其实就是一对公钥和私钥，如果对公钥和私钥不太理解，可以想象成一把钥匙和一个锁头，只是全世界只有你一个人有这把钥匙，你可以把锁头给别人，别人可以用这个锁把重要的东

西锁起来，然后发给你，因为只有你一个人有这把钥匙，所以只有你才能看到被这把锁锁起来的东西。

3、传送证书

这个证书其实就是公钥，只是包含了很多信息，如证书的颁发机构，过期时间等等。

4、客户端解析证书

这部分工作是有客户端的 TLS 来完成的，首先会验证公钥是否有效，比如颁发机构，过期时间等等，如果发现异常，则会弹出一个警告框，提示证书存在问题。

如果证书没有问题，那么就生成一个随机值，然后用证书对该随机值进行加密，就好像上面说的，把随机值用锁头锁起来，这样除非有钥匙，不然看不到被锁住的内容。

5、传送加密信息

这部分传送的是用证书加密后的随机值，目的就是让服务端得到这个随机值，以后客户端和服务端的通信就可以通过这个随机值来进行加密解密了。

6、服务段解密信息

服务端用私钥解密后，得到了客户端传过来的随机值(私钥)，然后把内容通过该值进行对称加密，所谓对称加密就是，将信息和私钥通过某种算法混合在一起，这样除非知道私钥，不然无法获取内容，而正好客户端和服务端都知道这个私钥，所以只要加密算法够彪悍，私钥够复杂，数据就够安全。

7、传输加密后的信息

这部分信息是服务段用私钥加密后的信息，可以在客户端被还原。

8、客户端解密信息

客户端用之前生成的私钥解密服务段传过来的信息，于是获取了解密后的内容，整个过程第三方即使监听到了数据，也束手无策。

HTTPS 的优点

正是由于 HTTPS 非常的安全，攻击者无法从中找到下手的地方，从站长的角度来说，HTTPS 的优点有以下 2 点：

1、SEO 方面

谷歌曾在 2014 年 8 月份调整搜索引擎算法，并称“比起同等 HTTP 网站，采用 HTTPS 加密的网站在搜索结果中的排名将会更高”。

2、安全性

尽管 HTTPS 并非绝对安全，掌握根证书的机构、掌握加密算法的组织同样可以进行中间人形式的攻击，但 HTTPS 仍是现行架构下最安全的解决方案，主要有以下几个好处：

（1）、使用 HTTPS 协议可认证用户和服务器，确保数据发送到正确的客户机和服务器；

（2）、HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，要比 http 协议安全，可防止数据在传输过程中不被窃取、改变，确保数据的完整性。

（3）、HTTPS 是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本。

HTTPS 的缺点

虽然说 HTTPS 有很大的优势，但其相对来说，还是有些不足之处的，具体来说，有以下 2 点：

1、SEO 方面

据 ACM CoNEXT 数据显示，使用 HTTPS 协议会使页面的加载时间延长近 50%，增加 10%到 20%的耗电，此外，HTTPS 协议还会影响缓存，增加数据开销和功耗，甚至已有安全措施也会受到影响也会因此而受到影响。

而且 HTTPS 协议的加密范围也比较有限，在黑客攻击、拒绝服务攻击、服务器劫持等方面几乎起不到什么作用。

最关键的，SSL 证书的信用链体系并不安全，特别是在某些国家可以控制 CA 根证书的情况下，中间人攻击一样可行。

2、经济方面

（1）、SSL 证书需要钱，功能越强大的证书费用越高，个人网站、小网站没有必要一般不会用。

（2）、SSL 证书通常需要绑定 IP，不能在同一 IP 上绑定多个域名，IPv4 资源不可能支撑这个消耗（SSL 有扩展可以部分解决这个问题，但是比较麻烦，而且要求浏览器、操作系统支持，Windows XP 就不支持这个扩展，考虑到 XP 的装机量，这个特性几乎没用）。

（3）、HTTPS 连接缓存不如 HTTP 高效，大流量网站如非必要也不会采用，流量成本太高。

（4）、HTTPS 连接服务器端资源占用高很多，支持访客稍多的网站需要投入更大的成本，如果全部采用 HTTPS，基于大部分计算资源闲置的假设的 VPS 的平均成本会上去。

（5）、HTTPS 协议握手阶段比较费时，对网站的相应速度有负面影响，如非必要，没有理由牺牲用户体验。

四、 Cookie 和 Session

4-1 、Cookie 和 Session 的区别?

Cookie 是 web 服务器发送给浏览器的一块信息，浏览器会在本地一个文件中给每个 web 服务器存储 cookie。

以后浏览器再给特定的 web 服务器发送请求时，同时会发送所有为该服务器存储的 cookie。

Session 是存储在 web 服务器端的一块信息。session 对象存储特定用户会话所需的属性及配置信息。

当用户在应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。

Cookie 和 session 的不同点：

1、无论客户端做怎样的设置，session 都能够正常工作。当客户端禁用 cookie 时将无法使用 cookie。

2、在存储的数据量方面：session 能够存储任意的 java 对象，cookie 只能存储 String 类型的对象。

4-2 session 共享怎么做的（分布式如何实现 session 共享）？

问题描述：一个用户在登录成功以后会把用户信息存储在 session 当中，这时 session 所在服务器为 server1，那么用户在 session 失效之前如果再次使用 app，那么可能会被路由到 server2，这时问题来了，server 没有该用户的 session，所以需要用户重新登录，这时的用户体验会非常不好，

所以我们想如何实现多台 server 之间共享 session，让用户状态得以保存。

1、服务器实现的 session 复制或 session 共享，这类型的共享 session 是和服务器紧密相关的，比如 webSphere 或 JBOSS 在搭建集群时候可以配置实现 session 复制或 session 共享，但是这种方式有一个致命的缺点，就是不好扩展和移植，比如我们更换服务器，那么就要修改服务器配置。

2、利用成熟的技术做 session 复制，比如 12306 使用的 gemfire，比如常见的内存数据库如 redis 或 memorycache，这类方案虽然比较普适，但是严重依赖于第三方，这样当第三方服务器出现问题的时候，那么将是应用的灾难。

3、将 session 维护在客户端，很容易想到就是利用 cookie，但是客户端存在风险，数据不安全，而且可以存放的个增强，这里主要重写了几个 request 的方法，但是最重要的是重写了 request.getSession()，写到这里大家应该都明白为什么重写这个方法了吧，当然是为了获取 MySession，

于是这样就在 filter 中，偷偷的将原生的 request 换成 MyRequest 了，然后再将替换过的 request 传入 chan.doFilter()，这样 filter 时候的代码都使用的是 MyRequest 了，同时对业务代码是透明的，业务代码获取 session 的方法仍然是 request.getSession（），但其实获取到的已经是 MySession 了，这样对 session 的操作已经变成了对 redis 的操作。

这样实现的好处有两个，第一开发人员不需要对 session 共享做任何关注，session 共享对用户是透明的；第二，filter 是可配置的，通过 filter 的方式可以将 session 共享做成一项可插拔的功能，没有任何侵入性。

这个时候已经实现了一套可插拔的 session 共享的框架了，但是我们想到如果 redis 服务出了问题，这时我们该怎么办呢，于是我们延续 redis 的想法，想到可以将 session 维护在客户端内（加密的 cookie），当然实现方法还是一样的，我们重写 HttpSession 接口，实现其所有方法，比如 setAttribute 就是写入 cookie，getAttribute 就是读取 cookie，我们可以将重写的 session 称作 MySession2，这时怎么让开发人员透明的获取到 MySession2 呢，实现方法还是在 filter 内偷梁换柱，在 MyRequest 加一个判断，读取 sessionType 配置，如果 sessionType 是 redis 的，那么 getSession 的时候获取到的是 MySession，如果 sessionType 是 coolie 的，那么 getSession 的时候获取到的是 MySession2，以此类推，用同样的方法就可以获取到 MySession 3,4,5,6 等等。

这样两种方式都有了，那么我们怎实现两种 session 共享方式的快速切换呢，刚刚我提到一个 sessionType，这是用来决定获取到 session 的类型的，只要变换 sessionType 就能实现两种 session 共享方式的切换，但是 sessionType 必须对所有的服务器都是一致的，如果不一致那将会出现比较严重的问题，我们目前是将 sessionType 维护在环境变量里，如果要切换 sessionType 就要重启每一台服务器，完成 session 共享的转换，但是当服务器太多的时候将是一种灾难。而且重启服务意味着服务的中断，所以这样的方式只适合服务器规模比较小，而且用户量比较少的情况，当服务器太多的时候，务必需要一种协调技术，能够让服务器能够及时获取切换的通知。基于这样的原因，我们选用 zookeeper 作为配置平台，每一台服务器都会订阅 zookeeper 上的配置，当我们切换 sessionType 之后，所有服务器都会订阅到修改之后的配置，那么切换就会立即生效，当然可能会有短暂的时间延迟，但这是可以接受的。

4-3 在单点登录中，如果 cookie 被禁用了怎么办?

单点登录的原理是后端生成一个 session ID，然后设置到 cookie，后面的所有请求浏览器都会带上 cookie，然后服务端从 cookie 里获取 session ID，再查询到用户信息。所以，保持登录的关键不是 cookie，而是通过 cookie 保存和传输的 session ID，其本质是能获取用户信息的数据。除了 cookie，还通常使用 HTTP 请求头来传输。但是这个请求头浏览器不会像 cookie 一样自动携带，需要手工处理。

五、 jsp 技术

5-1 什么是 jsp，什么是 Servlet? jsp 和 Servlet 有什么区别?

jsp 本质上就是一个 Servlet，它是 Servlet 的一种特殊形式（由 SUN 公司推出），每个 jsp 页面都是一个 servlet 实例。

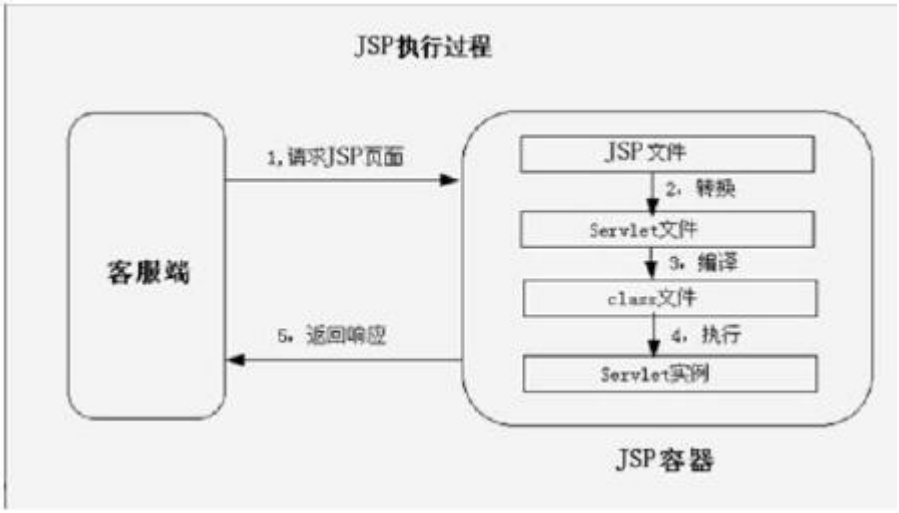
Servlet 是由 Java 提供用于开发 web 服务器应用程序的一个组件，运行在服务端，由 servlet 容器管理，用来生成动态内容。一个 servlet 实例是实现了特殊接口 Servlet 的 Java 类，所有自定义的 servlet 均必须实现 Servlet 接口。

区别:

jsp 是 html 页面中内嵌的 Java 代码，侧重页面显示;

Servlet 是 html 代码和 Java 代码分离，侧重逻辑控制，mvc 设计思想中 jsp 位于视图层，servlet 位于控制层

Jsp 运行机制：如下图



JVM 只能识别 Java 类，并不能识别 jsp 代码！web 容器收到以.jsp 为扩展名的 url 请求时，会将访问请求交给 tomcat 中 jsp 引擎处理，每个 jsp 页面第一次被访问时，jsp 引擎将 jsp 代码解释为一个 servlet 源程序，接着编译
servlet 源程序生成.class 文件，再有 web 容器 servlet 引擎去装载执行 servlet 程序，实现页面交互。

5-2 jsp 有哪些域对象和内置对象及他们的作用

四大域对象：
（1）pageContext page 域-指当前页面，在当前 jsp 页面有效，跳到其它页面失效
（2）request request 域-指一次请求范围内有效，从 http 请求到服务器处理结束，返回响应的整个过程。
在这个过程中使用 forward（请求转发）方式跳转多个 jsp，在这些页面里你都可以使用这个变量
（3）session session 域-指当前会话有效范围，浏览器从打开到关闭过程中，转发、重定向均可以使用
（4）application context 域-指只能在同一个 web 中使用，服务器未关闭或者重启，数据就有效
九大内置对象：
JSP 中一共预先定义了 9 个这样的对象，分别为：request、response、session、application、out、pagecontext、config、page、exception

1、request 对象

request 对象是 javax.servlet.httpServletRequest 类型的对象。 该对象代表了客户端的请求信息，主要用于接受通过 HTTP 协议传送到服务器的数据。（包括头信息、系统信息、请求方式以及请求参数等）。request 对象的作用域为一次请求。

2、response 对象

response 代表的是对客户端的响应，主要是将 JSP 容器处理过的对象传回到客户端。response 对象也具有作用域，它只在 JSP 页面内有效。

3、session 对象

session 对象是由服务器自动创建的与用户请求相关的对象。服务器为每个用户都生成一个 session 对象，用于保存该用户的信息，跟踪用户的操作状态。session 对象内部使用 Map 类来保存数据，因此保存数据的格式为 “Key/value” 。 session 对象的 value 可以使复杂的对象类型，而不仅仅局限于字符串类型。

4、application 对象

application 对象可将信息保存在服务器中，直到服务器关闭，否则 application 对象中保存的信息会在整个应用中都有效。与 session 对象相比，application 对象生命周期更长，类似于系统的“全局变量”。

5、out 对象

out 对象用于在 Web 浏览器内输出信息，并且管理应用服务器上的输出缓冲区。在使用 out 对象输出数据时，可以对数据缓冲区进行操作，及时清除缓冲区中的残余数据，为其他的输出让出缓冲空间。待数据输出完毕后，要及时关闭输出流。

6、pageContext 对象

pageContext 对象的作用是取得任何范围的参数，通过它可以获取 JSP 页面的 out、request、reponse、session、application 等对象。pageContext 对象的创建和初始化都是由容器来完成的，在 JSP 页面中可以直接使用 pageContext 对象。

7、config 对象

config 对象的主要作用是取得服务器的配置信息。通过 pageConext 对象的 getServletConfig() 方法可以获取一个 config 对象。当一个 Servlet 初始化时，容器把某些信息通过 config 对象传递

给这个 Servlet。 开发者可以在 web.xml 文件中为应用程序环境中的 Servlet 程序和 JSP 页面提供初始化参数。

8、page 对象

page 对象代表 JSP 本身，只有在 JSP 页面内才是合法的。 page 隐含对象本质上包含当前 Servlet 接口引用的变量，类似于 Java 编程中的 this 指针。

9、exception 对象

exception 对象的作用是显示异常信息，只有在包含 isErrorPage="true" 的页面中才可以被使用，在一般的 JSP 页面中使用该对象将无法编译 JSP 文件。excepation 对象和 Java 的所有对象一样，都具有系统提供的继承结构。exception 对象几乎定义了所有异常情况。在 Java 程序中，可以使用 try/catch 关键字来处理异常情况； 如果在 JSP 页面中出现没有捕获到的异常，就会生成 exception 对象，并把 exception 对象传送到在 page 指令中设定的错误页面中，然后在错误页面中处理相应的 exception 对象

六、 XML 技术

6-1 什么是 xml，使用 xml 的优缺点，xml 的解析器有哪几种，分别有什么区别?

xml 是一种可扩展性标记语言，支持自定义标签（使用前必须预定义）使用 DTD 和 XML Schema 标准化 XML 结构。

优点：用于配置文件，格式统一，符合标准；用于在互不兼容的系统间交互数据，共享数据方便；

缺点：xml 文件格式复杂，数据传输占流量，服务端和客户端解析 xml 文件占用大量资源且不易维护

Xml 常用解析器有 2 种，分别是：DOM 和 SAX;

主要区别在于它们解析 xml 文档的方式不同。使用 DOM 解析，xml 文档以 DOM 树形结构加载入内存，而 SAX 采用的是事件模型

七、 Servlet 生命周期

1、Servlet 通过调用 init () 方法进行初始化。

2、Servlet 调用 service() 方法来处理客户端的请求。

3、Servlet 通过调用 destroy() 方法终止（结束）。

4、最后，Servlet 是由 JVM 的垃圾回收器进行垃圾回收的

八、 tomcat 容器是如何创建 servlet 类实例？用到了什么原理?

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）

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。

九、 Java servlet、filter、listener、interceptor 之间的区别和联系?

1.servlet：servlet 是一种运行服务器端的 java 应用程序

2.filter：filter 是一个可以复用的代码片段，可以用来转换 HTTP 请求、响应和头信息

3.listener：监听器，通过 listener 可以监听 web 服务器中某一个执行动作，并根据其要求作出相应的响应。

4.interceptor：是在面向切面编程的，就是在你的 service 或者一个方法，前调用一个方法，或者在方法后调用一个方法。比如动态代理就是拦截器的简单实现

servlet、filter、listener 是配置到 web.xml 中，interceptor 不配置到 web.xml 中，struts 的拦截器配置到 struts.xml 中。spring 的拦截器配置到 spring.xml 中。

5.加载顺序

web.xml 的加载顺序是：context- param -> listener -> filter -> servlet

十、 前端技术

9-1 谈谈你对 ajax 的认识?

Ajax 是一种创建交互式网页应用的的网页开发技术

Ajax 的最大特点：

可以实现局部刷新，在不更新整个页面的前提下维护数据，提升用户体验度。

9-2 jsonp 原理

jsonp 的最基本的原理是：动态添加一个<script>标签，使用 script 标签的 src 属性没有跨域的限制的特点实现跨域。首先在客户端注册一个 callback, 然后把 callback 的名字传给服务器。此时，服务器先生成 json 数据。 然后以 javascript 语法的方式，生成一个 function , function 名字就是传递上来的参数 jsonp。最后将 json 数据直接以入参的方式，放置到 function 中，这样就生成了一段 js 语法的文档，返回给客户端。

客户端浏览器，解析 script 标签，并执行返回的 javascript 文档，此时数据作为参数，传入到了客户端预先定义好的 callback 函数里。

9-3 跨域两种解决方案

Jsonp 和 CORS

CORS 是一个 W3C 标准，全称是"跨域资源共享"（Cross-origin resource sharing）。CORS 需要浏览器和服务器同时支持。目前，所有浏览器都支持该功能，IE 浏览器不能低于 IE10。它允许浏览器向跨源服务器，发出 XMLHttpRequest 请求，从而克服了 AJAX 只能同源使用的限制。整个 CORS 通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS 通信与同源的 AJAX 通信没有差别，代码完全一样。浏览器一旦发现 AJAX 请求跨源，就会自动添加一些附加的头信息，有时还会多出一次附加的请求，但用户不会有感觉。因此，实现 CORS 通信的关键是服务器。只要服务器实现了 CORS 接口，就可以跨源通信。CORS 请求默认不发送 Cookie 和 HTTP 认证信息。如果要把 Cookie 发到服务器，一方面要服务器同意，指定 Access-Control-Allow-Credentials 字段。另一方面，开发者必须在 AJAX 请求中打开 withCredentials 属性。否则，即使服务器同意发送 Cookie，浏览器也不会发送。或者，服务器要求设置 Cookie，浏览器也不会处理。

9-4 常见的前端框架有哪些

EasyUI LayUI Angularjs VUE.js