# Web阶段重点及面试题汇总

## JDBC相关

### 如何通过JDBC访问数据库

JDBC（Java数据库连接）即利用Java程序操作数据库的一门技术

JDBC提供了连接数据库、执行sql语句及获取结果数据的各种访问数据库的方法。并为各种不同的数据库提供统一的访问接口。通过JDBC访问数据库一般需要如下步骤：

1.注册数据库驱动

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

2.获取数据库连接

Connection conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/jt\_db",

"root",

"root");

3.获取传输器对象(Statement或PreparedStatement)

Statement stat = conn.createStatement();

4.利用传输器发送sql到数据库执行, 返回执行结果

String sql = "select \* from account";

ResultSet rs = stat.executeQuery(sql);

5.处理结果

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

double money = rs.getDouble("money");

System.out.println(

id+" : "+name+" : "+money);

}

6.释放资源(依次将ResultSet、Statement(或PreparedStatement)、Connection对象关闭)

rs.close();

stat.close();

conn.close();

问：为什么要这么做呢？(1)这些对象使用完后后面几乎不会再用到，如果不释放会占用服务器的内存空间，容易造成内存溢出。(2)其中的Connection连接数是有限的，如不及时释放，别的用户可能会获取不到连接，将无法访问数据库。

### Statement对象和PreparedStatement对象的区别

1、Statement传输器对象用于执行不带参数（提前将sql语句和参数拼接）的简单sql语句。如果每次执行的sql语句不同（即使sql骨架相同），数据库都要编译该sql语句。例如：

Statement stat = conn.createStatement();

stat.executeQuery("select \* from user where username='zhansan' and password=123");

2、PreparedStatement对象表示预编译的sql语句的sql语句的对象，用于执行带参数的预编译sql语句：

PreparedStatement ps = conn.prepareStatement("select \* from user where username=? and password=?");

ps.setString("zhangsan");

ps.setString("123");

ps.executeQuery();

3、虽然Statement对象和PreparedStatement对象可以完成相同的功能。但是相比之下，PreparedStatement对象具有如下优点：

(1)防止sql注入攻击

PreparedStatement对象在执行sql时，提前将sql语句的骨架（骨干）发送给数据库编译并确定下来，由于sql语句的骨架已经确定，所以后面再传入的就只能是参数的值，如果参数值中包含了影响sql语句的关键字或者特殊字符，也只会当作普通文本来处理。因此不会产生sql注入攻击问题。

(2)代码可读性好并且省去了拼接sql语句的麻烦。（见上↑）

(3)可以提高程序执行的效率。

不管使用哪一个传输器发送sql语句到数据库执行，都会先编译再执行，如果下次执行的sql语句和之前执行的完全相同，则会直接使用之前编译好的，将不会再次编译。

在同一种操作中（例如根据用户名和密码查询用户），PreparedStatement对象执行的sql骨架每次都是相同的，因此只需要编译一次。效率较高。

而Statement对象在执行sql时，每次都是将sql骨架和参数拼接好，只要参数不同，整条sql语句也就不同，因此每次都需要编译。

### 什么是数据库连接池

所谓的池就是程序中用来共享资源的容器，比如线程池，常量池。

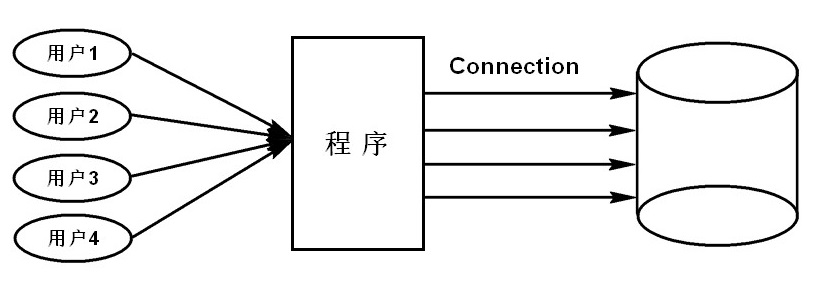
而数据库连接池就是用来存放连接的容器，在开发JDBC程序时，可以将数据库连接存放在池子（其实就是一个容器，比如数组或者集合等）中，用于在整个程序中共享连接。

### 连接池的工作原理（即为什么要使用连接池？）

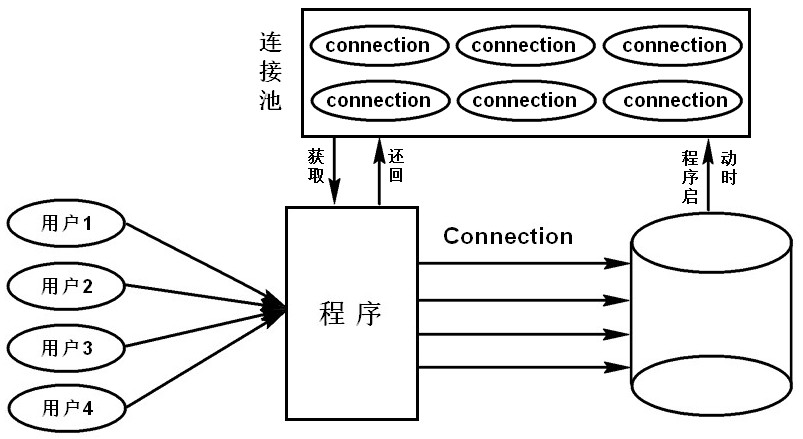
通过传统JDBC操作数据库，需要连接时就创建一个连接，用完后需要将连接关闭还给数据库。当并发访问的用户非常多时，这种频繁开关连接的操作将会十分的耗费资源，并且效率低下。

因此我们可以在程序中创建一个连接池，当程序启动时就初始化一批连接放在池中供整个程序共享。当需要连接时从连接池中获取一个连接进行使用，当用完连接后将连接还回连接池中。这样一来既可以减少连接开关的次数，也可以实现连接的复用，从而提高程序执行的效率。

传统JDBC访问数据库:



使用JDBC+数据库连接池访问数据库:



### 如何自己实现一个数据库连接池

SUN公司为实现连接池提供了一个接口: javax.sql.DataSource，要求所有的连接池必须实现这个接口

实现数据库连接池的步骤

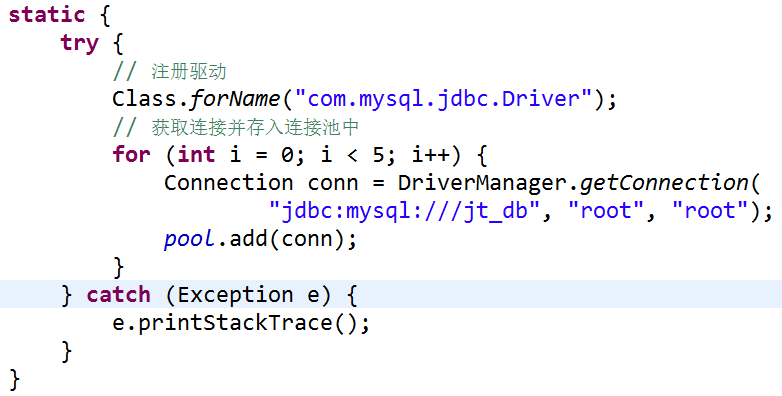
1、写一个类，实现 DataSource 接口



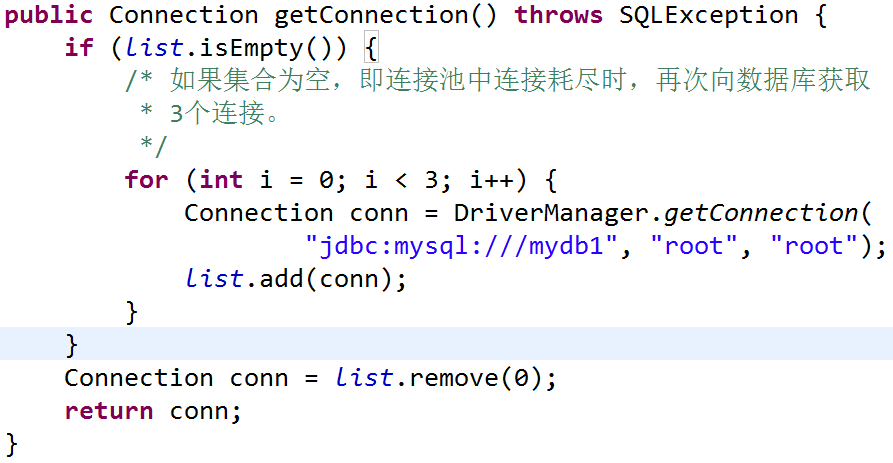
2、在实现类中，创建一个容器(LinkedList)，当作连接池使用



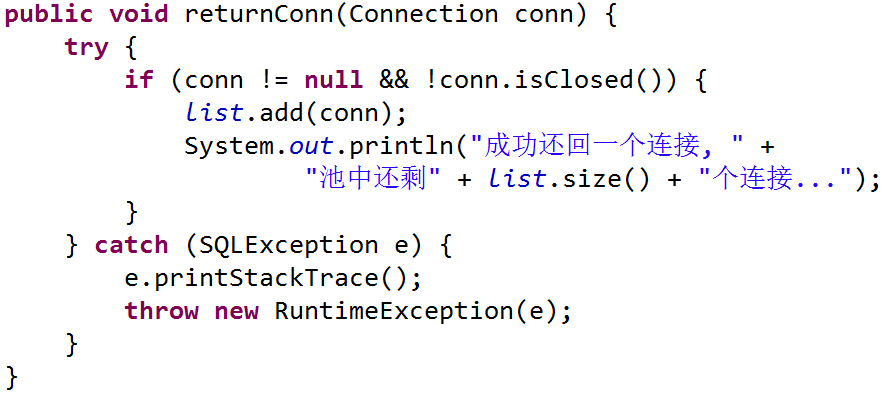
3、在静态代码块中，初始化一批连接放在连接池中



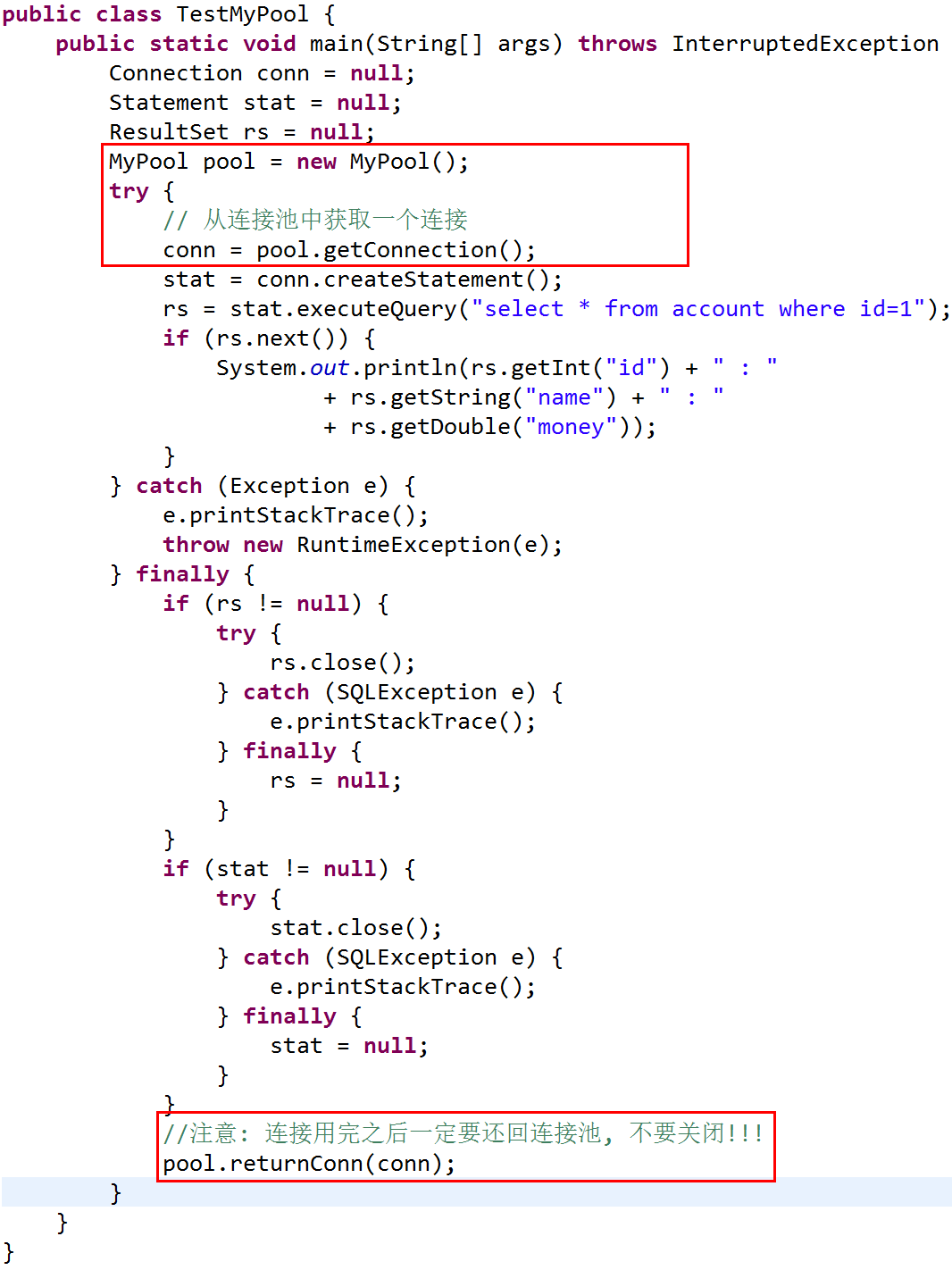
4、实现 getConnection 方法，方便获取连接



5、添加自定义方法 returnConn，用于将连接还回连接池中



6、写测试类测试自定义连接池



## Servlet、JSP

### 什么是Servlet？

HTML只能保存静态网页内容，而在实际开发中，往往需要给用户展示动态页面。Servlet就可以用来生成动态页面。

（所谓的动态页面是指：能够根据不同时间、不同用户而显示不同内容的页面，比如：留言板、订单页面、购物车页面等）

什么是Servlet呢？Servlet是SUN公司提供的动态Web资源开发技术，其本质上就是采用Java语言编写的服务器端程序。这段程序必须要放在Servlet容器中，由容器调用才可以执行。其主要作用是处理客户端的请求，并将处理结果响应给浏览器。例如：

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.getWriter().write("Hello..."+new Date().toString());

}

### JSP和Servlet有哪些相同点和不同点？他们之间的联系是什么？

**1.相同点**: JSP和Servlet都可以用于接收客户端请求、处理请求，并做出响应。

**2.不同点**:

(1) Servlet更适合处理逻辑，不适合作数据展示(即响应页面)

(2) JSP更适合展示数据，不适合处理逻辑

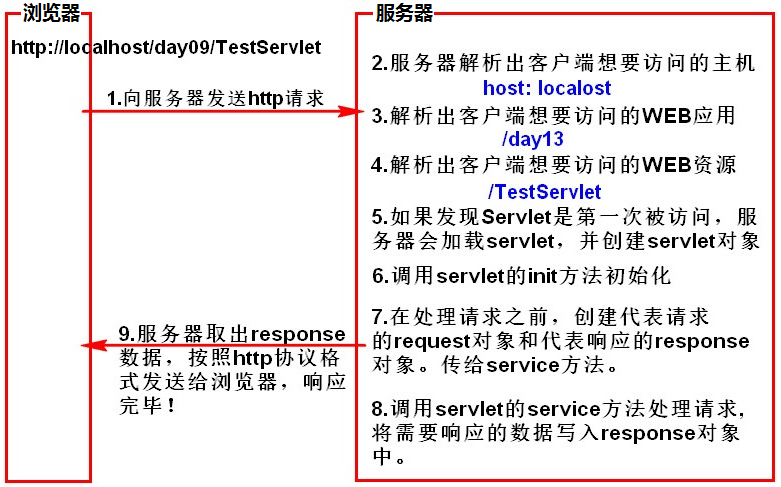
(3) Servlet没有内置对象、JSP中有内置对象

(4) 在MVC设计模式中，Servlet属于控制器(Controller)，JSP属于视图(View)

**3.两者之间的联系:**

JSP在一次被访问时，JSP引擎会对JSP进行编译，编译后就变成了Servlet，本质上就是一个Servlet。

### 描述Servlet调用过程



(1)浏览器向服务器发送请求（例如地址为：<http://localhost/day13/TestServlet>）

(2)服务端通过请求解析出浏览器访问的是哪一台主机（例如：localhost）

(3)服务端通过请求解析出浏览器访问的是哪一个WEB应用（例如：/day13）

(4)再通过请求解析出浏览器访问的是哪一个资源（例如：/TestServlet）

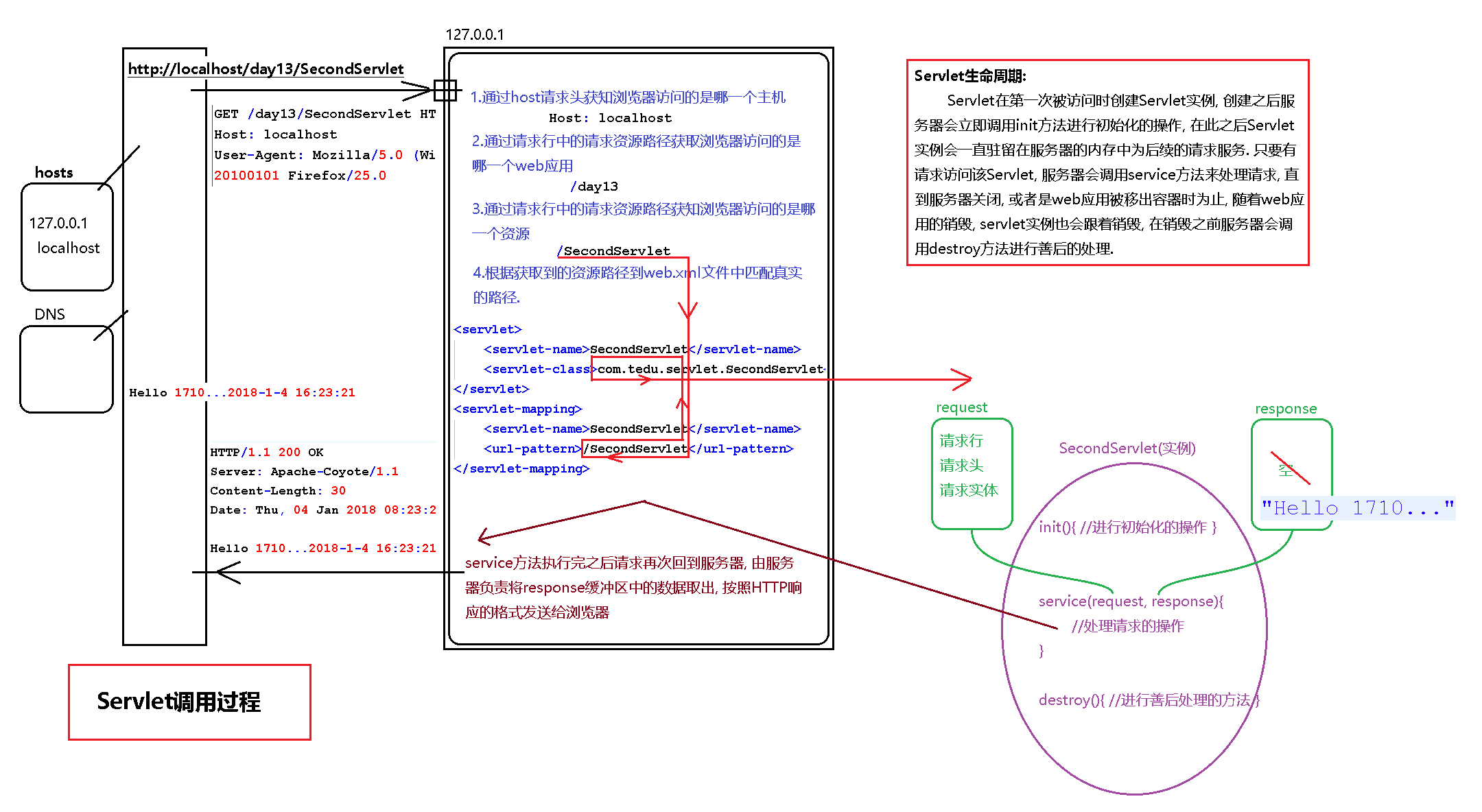
(5)如果该Servlet是第一次被访问，服务器会加载Servlet类，并创建该类的实例.

(6)调用Servlet的init方法进行初始化操作.

(7)在处理请求之前，创建代表请求的request对象和代表响应的response对象，传递给service方法。

(8)调用Servlet的service方法处理请求。将需要响应的数据写入response对象中

(9)服务器取出response中保存的数据，按照Http协议的格式发送给浏览器，响应完毕!



### 描述Servlet生命周期

Servlet在第一次被访问时创建Servlet实例，创建之后服务器会立即调用init方法进行初始化的操作，创建之后，Servlet实例会一直驻留在服务器的内存中，为后续的请求服务. 只要有请求来访问这个Servlet，服务器就会调用service方法来处理请求，直到服务器关闭，或者WEB应用被移出容器，随着WEB应用的销毁，Servlet实例也会跟着销毁，在销毁之前服务器会调用destroy方法进行善后的处理。

### 转发和重定向的区别（即forward和redirect的区别）?

(1)转发是同一个WEB应用内部资源之间的跳转，而重定向既可以是同一个WEB应用内部资源的跳转，也可以是不同的WEB应用或者不同的虚拟主机中资源的跳转.

(2)转发是一次请求，而重定向是两次请求

(3)转发前后地址栏的地址不会发生变化，而重定向前后地址栏地址会发生变化.

(4)转发前后的request对象是同一个，而重定向前后的request对象不是同一个.

### 描述四大域对象的特征

**1、PageContext域**

(1)生命周期：开始访问JSP页面时创建PageContext对象，访问JSP页面结束时销毁PageContext对象

(2)作用范围：在整个JSP页面中

(3)主要功能：在整个JSP页面中实现数据的共享。

**2、request域**

(1)生命周期：一次请求开始时创建request对象，一次请求结束时销毁request对象。

(2)作用范围：在整个请求链中

(3)主要功能：在整个请求链中实现数据的共享

**3、session域**

(1)生命周期：

创建: 当第一次调用request.getSession()方法时将会创建session对象.

销毁：session分为三种情况:

a)超时销毁: 如果超过30分钟不操作session，session将会超时销毁.

b)自杀: 当调用session.invalidate()方法时会立即销毁session

c)意外身亡: 当服务器非正常关闭时，session会销毁! 当服务器正常关闭时，session将会以文件的形式保存在tomcat服务器work目录下

(2)作用范围：在整个会话范围内

(3)主要功能：在整个会话范围内实现数据的共享

**4、ServletContext域**

(1)生命周期：服务器启动WEB应用被加载之后立即创建ServletContext对象，服务器关闭或WEB应用被移出容器，销毁ServletContext对象。

(2)作用范围：在整个WEB应用中

(3)主要功能：在整个WEB应用范围内实现数据的共享

### JSP的九大隐式对象是哪九个？

page、requeset、response、application（ServletContext）、config（ServletConfig）、session（HttpSession）、exception、out、pageContext

### 什么是过滤器及过滤器的作用

**1、什么是过滤器：**

过滤器也叫做Filter，是JavaWeb的三大组件之一。可以拦截用户对资源的访问。

所谓的拦截其实就是将代表请求的request对象和代表响应的response对象拦截下来,从而做一些相关处理。

**2、过滤器的作用：**

通过过滤器可以实现在用户访问某个资源之前或者之后，对访问的请求和响应进行拦截，从而做一些相关的处理。比如：

(1)根据用户登录与否控制是否允许访问

(2)全站乱码处理或异常处理

(3)权限控制

(4)自动登录功能实现

...

### 如何开发过滤器?

Servlet API中提供了一个Filter接口，开发web应用时，如果编写一个类实现了这个接口，则这个类就是一个过滤器。

(1)写一个类实现Filter接口，并实现其中的方法，例如：

public class FilterDemo1 implements Filter {

public void init(FilterConfig filterConfig)throws ServletException {}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)throws IOException,ServletException {

System.out.println(

"开始对拦截到的请求进行处理...demo1...");

chain.doFilter(request, response);

}

public void destroy() {}

}

(2)在web应用的web.xml中配置过滤器，例如：

<filter>

<filter-name>FilterDemo1</filter-name>

<filter-class>com.tedu.filter.FilterDemo1</filter-class>

</filter>

<filter-mapping>

<filter-name>FilterDemo1</filter-name>

<url-pattern>/ServletDemo1</url-pattern>

</filter-mapping>

### 什么是监听器及如何开发监听器?

**1、什么是监听器？**

监听器通常是一个实现特定接口的Java程序，该程序会监听另外一个Java对象的状态变化（例如对象的创建与销毁、对象中属性的更新等），一旦对象的状态发生变化，就会通知监听器，监听器会立即调用特定的方法来进行处理。

例如写一个类实现ServletContextListener接口，那么该类就是ServletContext生命周期监听器，当ServletContext对象创建时就会立即通知监听器，将会调用监听器中的contextInitialized方法来进行处理。而当ServletContext对象销毁时，也会立即通知监听器，将会调用监听器中的contextDestroyed方法来进行处理。

**2、如何开发监听器？**

(1)写一个类，实现监听器接口，并实现其中的方法, 例如:

public class MyServletContextListener

implements ServletContextListener {

//在ServletContext对象创建时调用

public void contextInitialized(

ServletContextEvent sce) {

System.out.println("我轻轻的来了...");

}

//在ServletContext对象销毁时调用

public void contextDestroyed(

ServletContextEvent sce) {

System.out.println("我轻轻的又走了...");

}

}

(2)在web.xml文件中注册监听器

<web-app>

<listener>

<listener-class>com.tedu.listener.MyServletContextListener</listener-class>

</listener>

</web-app>

## Cookie、Session

### 什么是会话？ 用自己的话描述Cookie和Session的工作原理

**1、什么是会话：**将浏览器和服务器之间的多次请求和响应加在一起就可以看作浏览器和服务器之间的一次会话。

**2、Cookie的工作原理：**



客户端向服务器发送请求需要保存数据时，服务器端获取到需要保存的数据，并将数据通过Set-Cookie响应头发送给浏览器，浏览器会以Cookie的形式保存起来（浏览器的内存中或者临时目录下）.

当客户端再次发送请求访问服务器，服务器可以通过Cookie请求头获取上次发送给浏览器的Cookie信息，通过这种方式可以保存会话中产生的数据.

**3、Session的工作原理:**



浏览器向服务器发送请求需要保存数据时，在服务器端获取到需要保存的数据，并获取到session对象，接着将数据保存在Session中，在响应时将session的ID以Cookie的形式发送给浏览器保存

当浏览器再去访问服务器时，请求中会携带着上次保存的sessionID，服务器可以通过sessionID获取到之前的session，再从session中获取到之前为当前浏览器保存的数据，通过这种方式，可以来保存会话中产生的数据！

### Web开发中session的使用场景? 什么时候session失效?

**1、session的使用场景**:

实现购物车

保存用户登录状态

**2、Session失效:**

见上↑

### 描述Cookie和Session的作用、区别 和 各自的应用范围

**1.Cookie和Session的作用:**

Cookie和Session都是会话技术，都可以用于保存会话过程中产生的数据，比如作为购物车用于保存商品信息.

**2.Cookie和Session的区别**:

(1)Cookie是将会话中产生的数据保存在客户端，是客户端的技术

(2)Session是将会话中产生的数据保存在服务器端，是服务器端的技术

(3)Cookie保存的信息的时间比较长，但是安全性不高，可能随着用户的操作，Cookie会被清空，所以Cookie存储数据的稳定性比较差。因此Cookie适合存放要保存时间较长，但安全性要求不高的信息

(4)Session通常保存信息的时间比较有限，但安全性比较高，因为是保存在服务器端，不会随着用户的操作而导致Session意外丢失，因此session适合存放安全性要求比较高，但是不需要长时间保存的数据。

(5)在性能上，Cookie性能更高一些。Cookie是将数据保存在用户各自的客户端浏览器中，而Session是服务器端的对象，是将数据（在一定时间内）保存在服务器端，因此当访问量增多时，会降低服务器的性能。

(6)在数据量上，单个Cookie保存的数据量不能超过4KB，而很多浏览器都限制一个站点给浏览器最多发送20个Cookie。而Session不存在此问题。

**3.各自的应用范围:**

Cookie：(1)实现购物车 (2)实现"记住用户名"功能 (3)实现7/30天自动登陆 等

Session：(1)实现购物车 (2)保存用户登录状态 等

## HTTP协议、服务器

### 描述HTTP协议的作用及具体内容（请求和响应的结构）

**1、Http协议的作用：**用来规定浏览器客户端和服务器之间通信的格式。即规定了浏览器在发送请求时该符合什么样的格式，和服务器在做响应时该符合什么样的格式。

**2、HTTP协议的内容：**HTTP请求（请求行、若干请求头、请求实体内容）、HTTP响应（状态行、若干响应头、响应实体内容）

### 简述http请求报文的结构(HTTP请求信息的格式)

请求报文/请求信息中包含三部分内容, 分别为请求行、若干请求头、请求实体内容， 如下：

1.请求行:

GET /test/index.jsp HTTP/1.1

2.若干请求头:

host : localhost

xxx : xxx

...

(一个空行)

3.请求实体内容

如果请求方式为GET提交，请求实体中没有任何数据

如果请求方式为POST提交，并且请求中携带了数据，此时请求实体中才会有数据

### GET请求方式和POST请求方式的区别

在HTTP协议中定义了7中请求方式，但是其中最常用的就是GET和POST。

GET和POST方式都可以向服务器发送请求，并且都可以在发送的请求中向服务器提交数据。

而两种方式的区别主要体现在请求参数传输过程的不相同。

**1、GET提交方式**

**(1)通过地址栏URL提交数据, 相对不安全**

采用GET提交向服务器向服务器发送数据时，需要将数据拼接在URL的后面进行发送，例如：（其中URL和数据之间用 "?" 连接，多个参数之间用 "&" 连接）

http://localhost/test/LoginServlet**?**username=zhangsan&password=123

由于这种方式会将数据显示在地址栏上，数据被彻底 "暴露" 出来，因此非常不安全。

**(2)通过地址栏提交数据, 数据量不能太大。**

由于浏览器对URL长度有限制，因此采用GET方式提交的数据量非常小。通常在1KB左右(有的浏览器限制为4KB)。

**2、POST提交方式**

**(1)通过请求实体提交数据, 相对更安全。**

**(2)通过请求实体提交数据, 理论上没有数据量的限制。**