**JavaWeb知识点**

1. **Http协议**

HTTP(Hyper Text Transfer Protocol)超文本传输协议

\*传输协议：定义了客户端和服务器端通信时，发送数据的格式,是应用层协议

\*特点：

1. 基于TCP/IP的高级协议
2. 默认端口号：80
3. 基于请求/响应模型：一次请求对应一次响应
4. 无状态到的：每次请求之间相互独立，不能交互数据

Http协议由Http请求和Http响应组成，当用浏览器访问某个网站时，浏览器将请求封装成一个Http请求发送给服务器站点，服务器接收到请求后会组织响应数据封装成一个Http响应返回给浏览器。



**Http请求**

1）请求行

请求方式：POST、GET

请求地址：/DemoEE/form.html   地址

协议版本：HTTP/1.1

HTTP/1.0，发送请求，创建一次连接，获得一个web资源，连接断开。

HTTP/1.1，发送请求，创建一次连接，获得多个web资源，保持连接。

2）请求头

Referer 请求来源网站。直接访问 没有    防盗链（防止盗取链接）

If-Modified-Since 浏览器最后变更时间。与某响应头控制页面的缓存

Cookie 与会话有关技术，用于存放浏览器缓存的cookie信息。

User-Agent 浏览器通知服务器，客户端浏览器与操作系统相关信息

Connection 保持连接状态。Keep-Alive 连接中/close 已关闭

Host 请求的服务器主机名

Content-Length 请求体的长度

Content-Type : application/x-www-form-urlencoded  POST请求特有，请求内容使用url编码

Accept： 浏览器可支持的MIME类型。MIME格式：大类型/小类型[;参数]

例如：

   text/html ，html文件

   text/css，css文件

   text/javascript，js文件

   image/\*，所有图片文件

Accept-Encoding 浏览器通知服务器，浏览器支持的数据压缩格式Accept-Language 浏览器通知服务器，浏览器支持的语言。

3）请求体

username=zhangsan&password=123 post请求

get请求会拼接在url地址后面

http://localhost:8080...?username=zhangsan&password=123

**Http响应**



1）响应行

Http协议

状态码：

常用的状态码如下：

200 ：请求成功

302 ：请求重定向（访问服务器两次  地址栏发生变化）

304 ：请求资源没有改变，访问本地缓存。

404 ：请求资源不存在。用户路径编写错误/服务器资源已删除

500 ：服务器内部错误。程序抛异常。

状态信息：状态信息是根据状态码变化而变化的

2）响应头

Location： 重定向地址

Content-Type：响应内容的解码格式（MIME类型）text/html;charset=UTF-8

Content-Disposition 通过浏览器以下载方式解析正文

取值：attachment;filename=xx.zip

Set-Cookie 与会话相关技术。服务器向浏览器写入cookie

Content-Encoding 服务器使用的压缩格式

Content-length 响应正文的长度

Refresh： 定时刷新，格式：秒数;url=路径。url可省略，默认值为当前页。

取值：3;url=www.itcast.cn    //三秒刷新页面到www.itcast.cn

Server 指的是服务器名称，默认值：Apache-Coyote/1.1。

通过server.xml配置进行修改。<Connector server="itcast"/>

Last-Modified 服务器通知浏览器，文件的最后修改时间。与If-Modified-Since一起使用。

3）响应体

响应体是服务器回写给客户端的页面正文，浏览器将正文加载到内存，然后解析渲染 显示页面内容

1. **GET请求和POST的请求区别**

\*GET:

1. 请求参数在请求行中，在URL之后；
2. 请求的URL长度有限制；
3. 不太安全。

\*POST:

1. 请求参数在请求体中；
2. 请求的URL长度没有限制；
3. 相对安全。

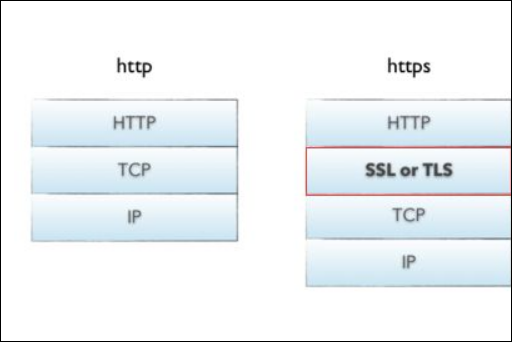
# **3**.http和https有什么区别?

http协议和https协议的区别：传输信息安全性不同、连接方式不同、端口不同、证书申请方式不同。

一、传输信息安全性不同

1、http协议：是超文本传输协议，信息是明文传输。如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息。

2、https协议：是具有安全性的ssl加密传输协议，为浏览器和服务器之间的通信加密，确保数据传输的安全。



二、连接方式不同

1、http协议：http的连接很简单，是无状态的。

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。

三、端口不同

1、http协议：使用的端口是80。

2、https协议：使用的端口是443．

四、证书申请方式不同

1、http协议：免费申请。

2、https协议：需要到ca申请证书，一般免费证书很少，需要交费。

Https采用的是对称加密+非对称加密+SSL(由ca机构颁发)

1. 对称加密

对称加密指的就是加密和解密使用同一个秘钥，所以叫做对称加密。对称加密只有一个秘钥，作为私钥。

常见的对称加密算法：DES，AES，3DES等等。

2. 非对称加密

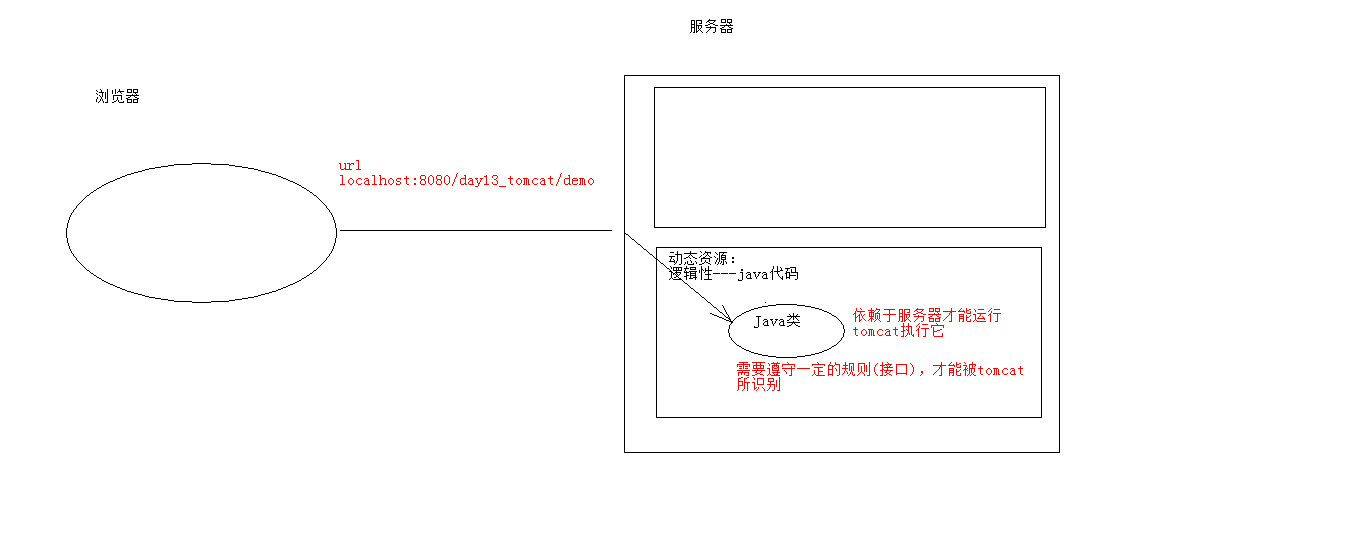
非对称加密指的是：加密和解密使用不同的秘钥，一把作为公开的公钥，另一把作为私钥。公钥加密的信息，只有私钥才能解密。私钥加密的信息，只有公钥才能解密。

常见的非对称加密算法：RSA，ECC

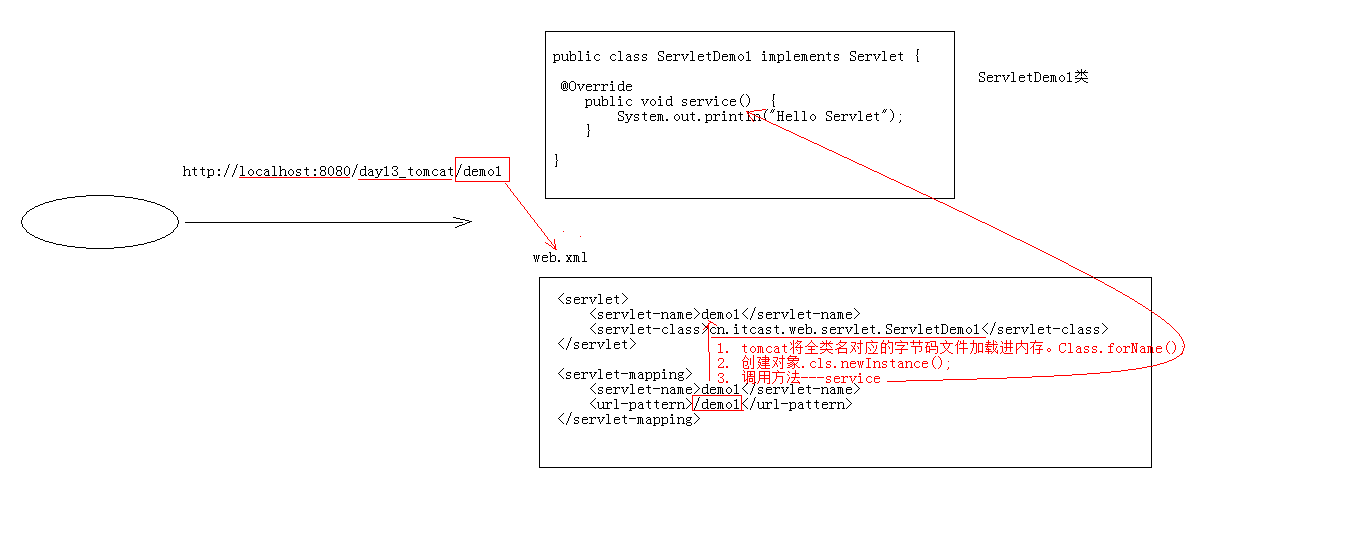
**4.Servlet**

## web服务器软件：  
    \* 服务器：安装了服务器软件的计算机  
    \* 服务器软件：接收用户的请求，处理请求，做出响应  
    \* web服务器软件：接收用户的请求，处理请求，做出响应。  
        \* 在web服务器软件中，可以部署web项目，让用户通过浏览器来访问这些项目  
        \* web容器  
  
    \* 常见的java相关的web服务器软件：  
        \* webLogic：oracle公司，大型的JavaEE服务器，支持所有的JavaEE规范，收费的。  
        \* webSphere：IBM公司，大型的JavaEE服务器，支持所有的JavaEE规范，收费的。  
        \* JBOSS：JBOSS公司的，大型的JavaEE服务器，支持所有的JavaEE规范，收费的。  
        \* Tomcat：Apache基金组织，中小型的JavaEE服务器，仅仅支持少量的JavaEE规范servlet/jsp。开源的，免费的。  
  
    \* JavaEE：Java语言在企业级开发中使用的技术规范的总和，一共规定了13项大的规范

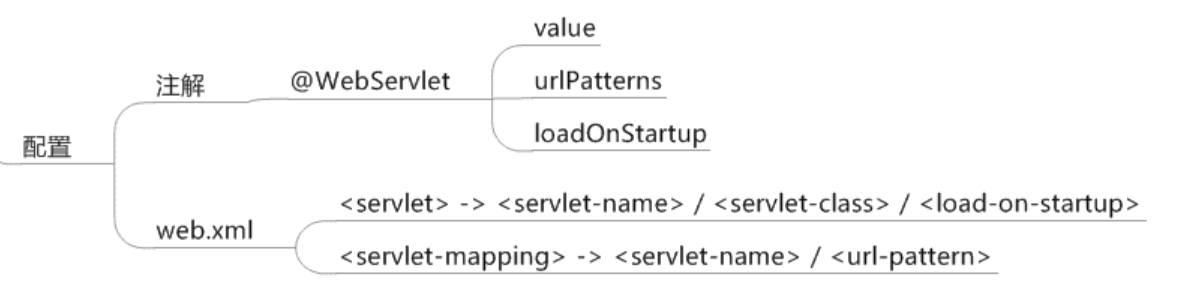
Servlet:\* Servlet就是一个接口，定义了Java类被浏览器访问到(tomcat识别) 的规则。  
    \* 将来我们自定义一个类，实现Servlet接口，复写方法。

****

\* 快速入门：  
        1. 创建JavaEE项目  
        2. 定义一个类，实现Servlet接口  
            \* public class ServletDemo1 implements Servlet  
        3. 实现接口中的抽象方法  
        4. 配置Servlet  
             在web.xml中配置：  
            <!--配置Servlet -->  
            <servlet>  
                <servlet-name>demo1</servlet-name>  
                <servlet-class>cn.itcast.web.servlet.ServletDemo1</servlet-class>  
            </servlet>  
          
            <servlet-mapping>  
                <servlet-name>demo1</servlet-name>  
                <url-pattern>/demo1</url-pattern>  
            </servlet-mapping>  
  
    \* 执行原理：  
        1. 当服务器接受到客户端浏览器的请求后，会解析请求URL路径，获取访问的Servlet的资源路径  
        2. 查找web.xml文件，是否有对应的<url-pattern>标签体内容。  
        3. 如果有，则在找到对应的<servlet-class>全类名  
        4. tomcat会将字节码文件加载进内存，并且创建其对象  
        5. 调用其方法



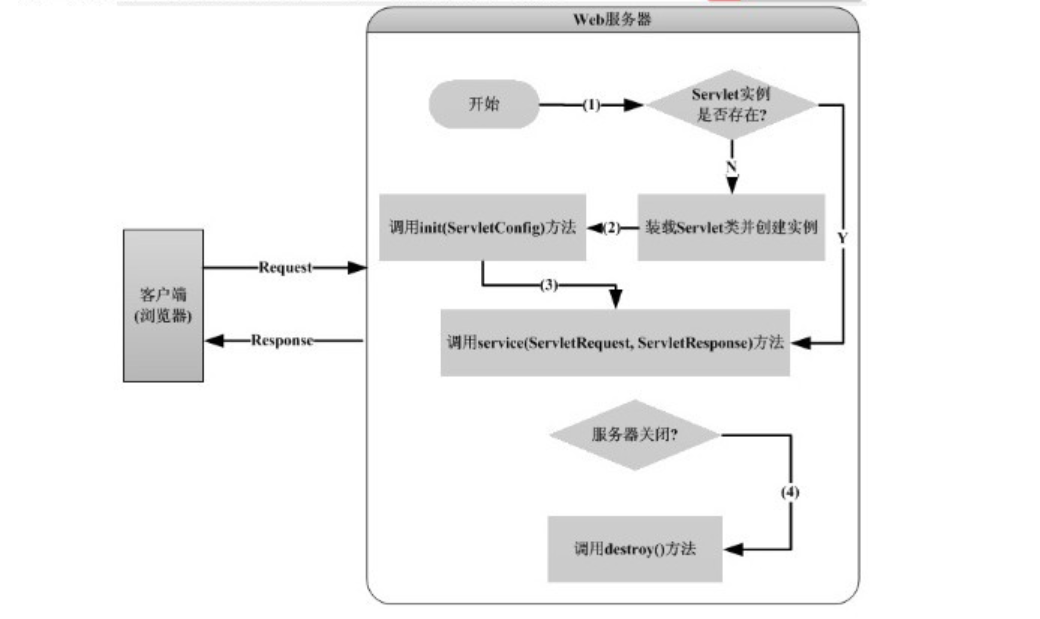
注解配置：



5.Servlet的生命周期？

Servlet的生命周期包含了下面4个阶段：

**1.加载和实例化；2.初始化 3.请求处理 4.服务终止**



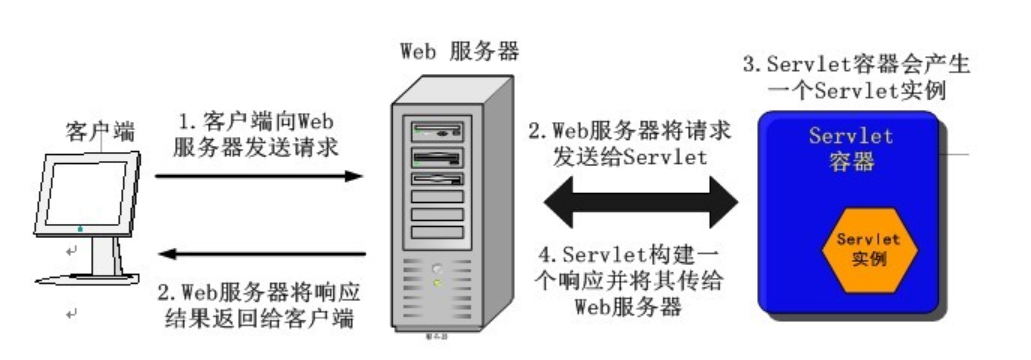
****6.Web服务器在与客户端交互时Servlet的工作过程是:****

1.   在客户端对web服务器发出请求

2.     web服务器接收到请求后将其发送给Servlet

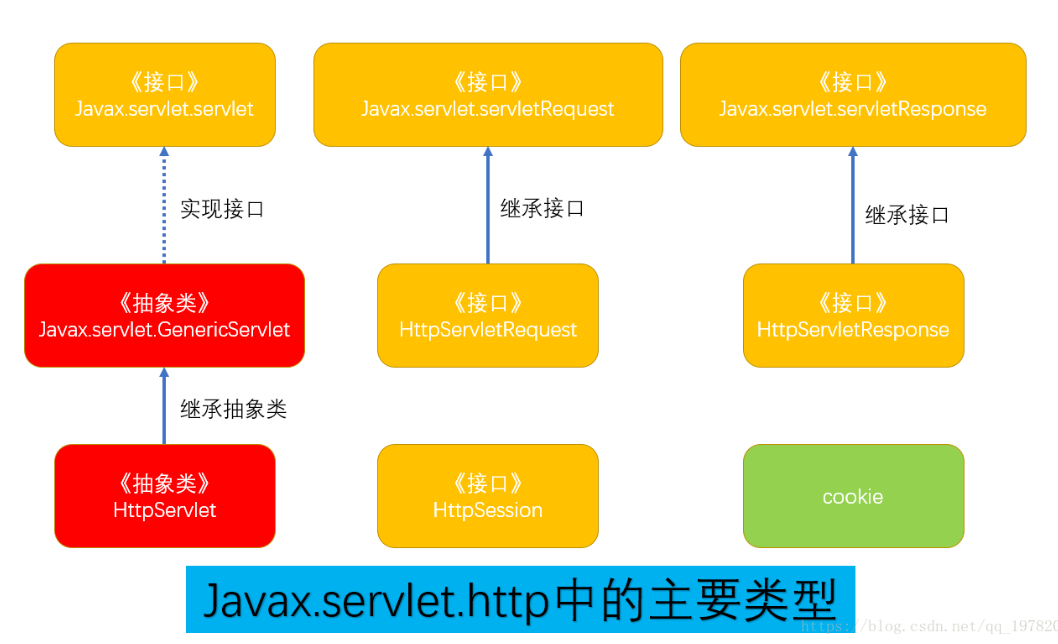
3.     Servlet容器为此产生一个实例对象并调用ServletAPI中相应的方法来对客户端 HTTP请求进行处理,然后将处理的响应结果返回给WEB服务器.

4.     web服务器将从Servlet实例对象中收到的响应结构发送回客户端.

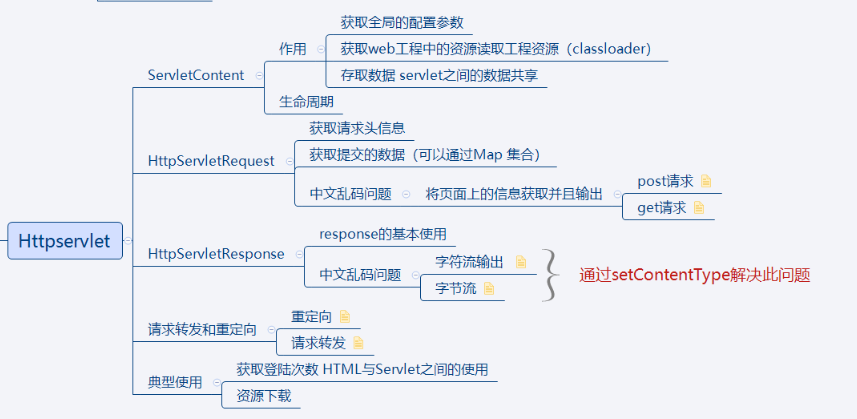


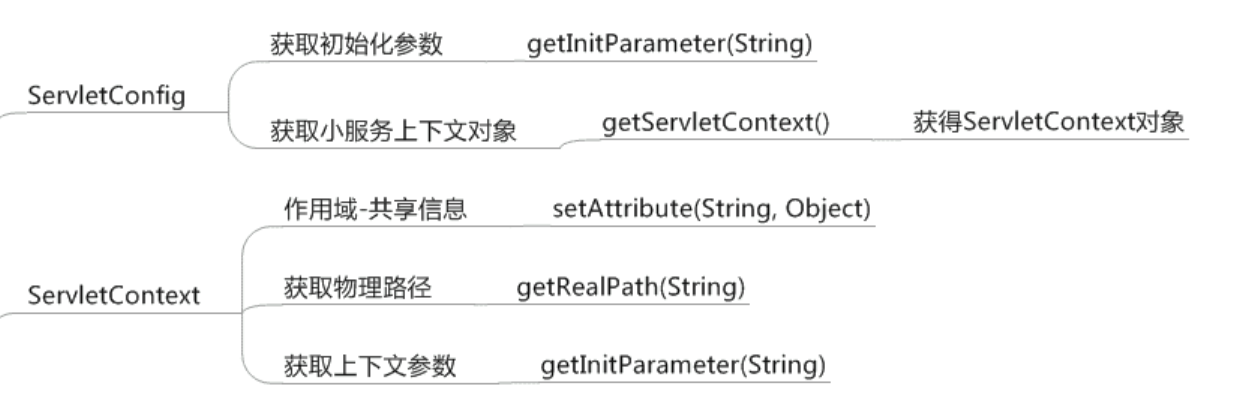
7.Servlet的体系结构

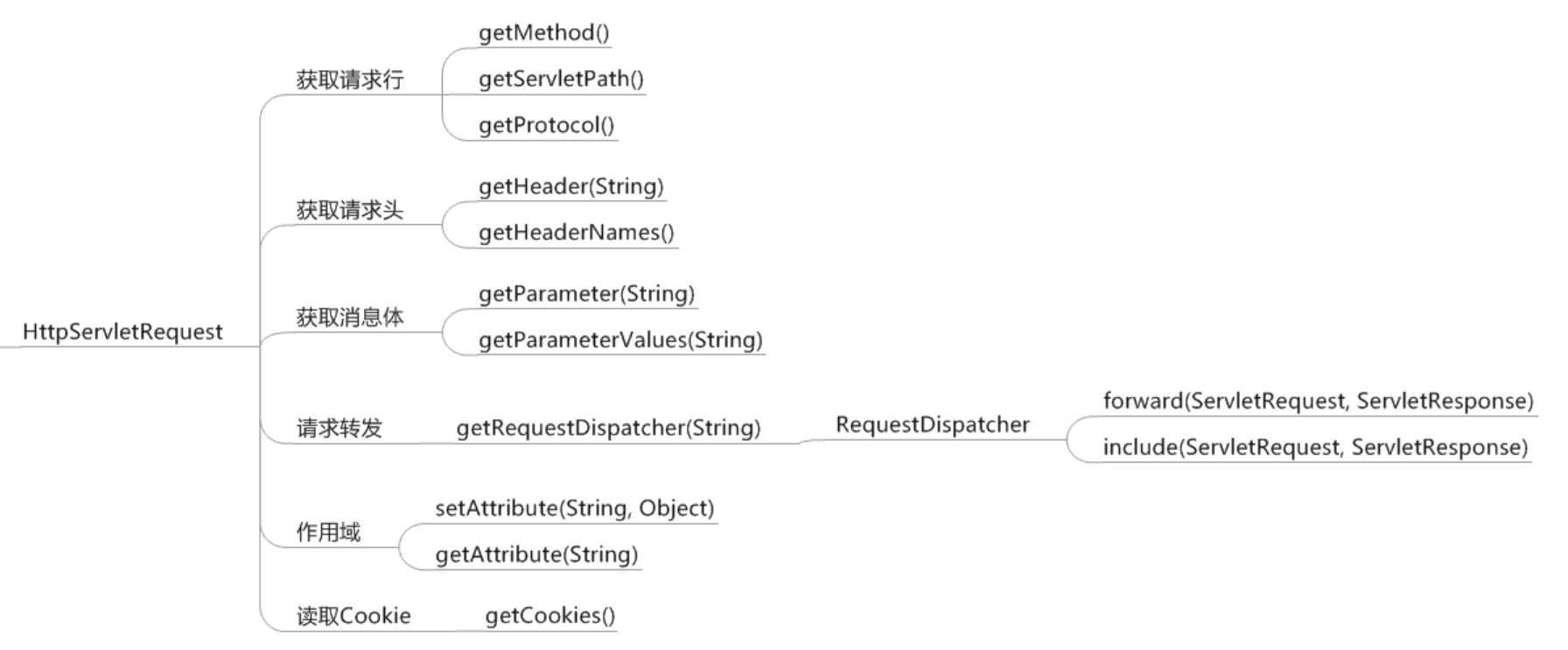
Servlet -- 接口  
            |  
        GenericServlet -- 抽象类  
            |  
        HttpServlet  -- 抽象类  
        \* GenericServlet：将Servlet接口中其他的方法做了默认空实现，只将service()方法作为抽象  
            \* 将来定义Servlet类时，可以继承GenericServlet，实现service()方法即可  
        \* HttpServlet：对http协议的一种封装，简化操作  
            1. 定义类继承HttpServlet  
            2. 复写doGet/doPost方法

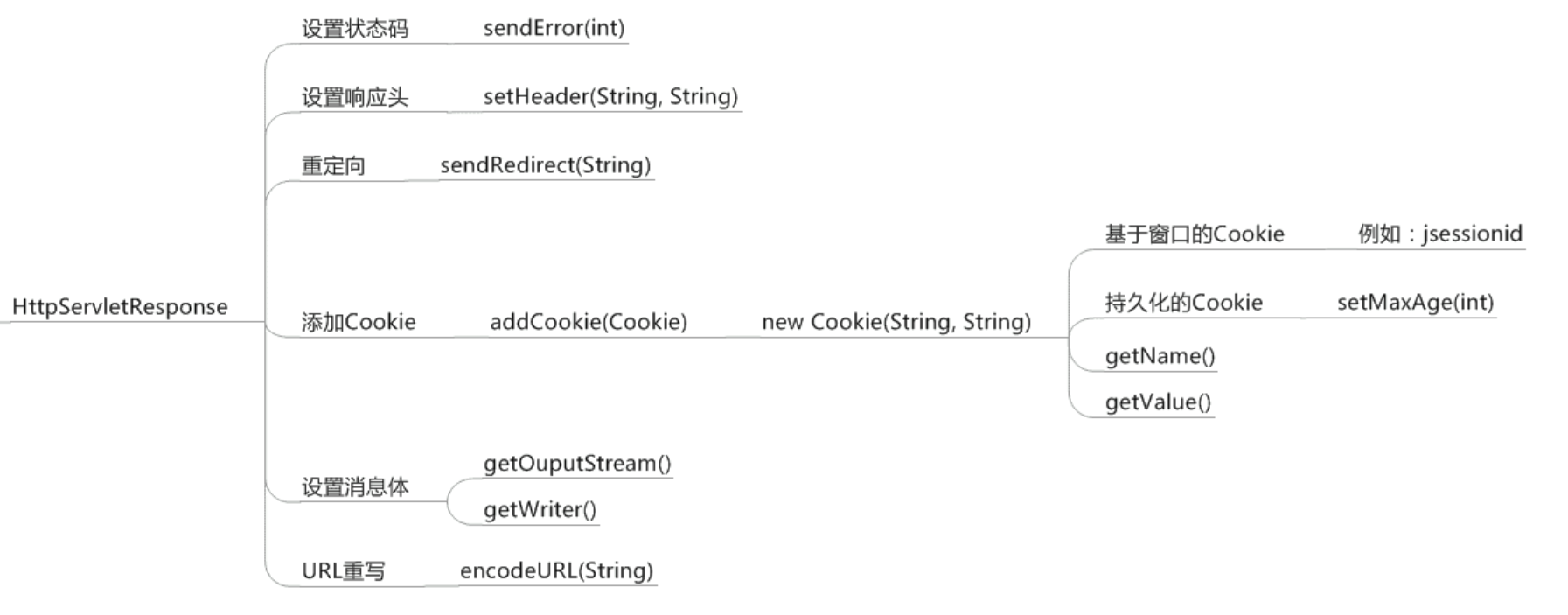


8.HttpServlet









**9.请求转发和重定向的区别？**

\*重定向：

1. 地址栏发生变化；
2. 重定向可以访问其他站点（服务器）的资源
3. 重定向是两次请求，不能使用request对象来共享数据

\*请求转发：

1. 转发地址栏路径不变；
2. 转发只能访问当前服务器下的资源；
3. 转发是一次请求，可以使用request对象来共享数据。

**10.cookie和session的区别？**

1.session存储数据在服务器端，cookie在客户端；

2.session没有数据大小限制，cookie有大小限制

3.session数据安全，cookie相对不安全

1. **JSP**

JSP和servlet

所有jsp文件都会被翻译成一个继承httpservlet的类，jsp最终也是servlet,对外提供服务，servlet主要是控制逻辑，jsp是视图。

1. 指令

\* 作用：用于配置JSP页面，导入资源文件

\* 格式：

<%@ 指令名称 属性名1=属性值1 属性名2=属性值2 ... %>

\* 分类：

1. page ： 配置JSP页面的

\* contentType：等同于response.setContentType()

1. 设置响应体的mime类型以及字符集

2. 设置当前jsp页面的编码（只能是高级的IDE才能生效，如果使用低级工具，则需要设置pageEncoding属性设置当前页面的字符集）

\* import：导包

\* errorPage：当前页面发生异常后，会自动跳转到指定的错误页面

\* isErrorPage：标识当前也是是否是错误页面。

\* true：是，可以使用内置对象exception

\* false：否。默认值。不可以使用内置对象exception

2. include ： 页面包含的。导入页面的资源文件

\* <%@include file="top.jsp"%>

3. taglib ： 导入资源

\* <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

\* prefix：前缀，自定义的

2. 注释:

1. html注释：

<!-- -->:只能注释html代码片段

2. jsp注释：推荐使用

<%-- --%>：可以注释所有

3. 内置对象

\* 在jsp页面中不需要创建，直接使用的对象

\* 一共有9个：

变量名 真实类型 作用

\* pageContext PageContext 当前页面共享数据，还可以获取其他八个内置对象

\* request HttpServletRequest 一次请求访问的多个资源(转发)

\* session HttpSession 一次会话的多个请求间

\* application ServletContext 所有用户间共享数据

\* response HttpServletResponse 响应对象

\* page Object 当前页面(Servlet)的对象 this

\* out JspWriter 输出对象，数据输出到页面上

\* config ServletConfig Servlet的配置对象

\* exception Throwable 异常对象

1. **MVC模式**

1. M：Model，模型。JavaBean

\* 完成具体的业务操作，如：查询数据库，封装对象

2. V：View，视图。JSP

\* 展示数据

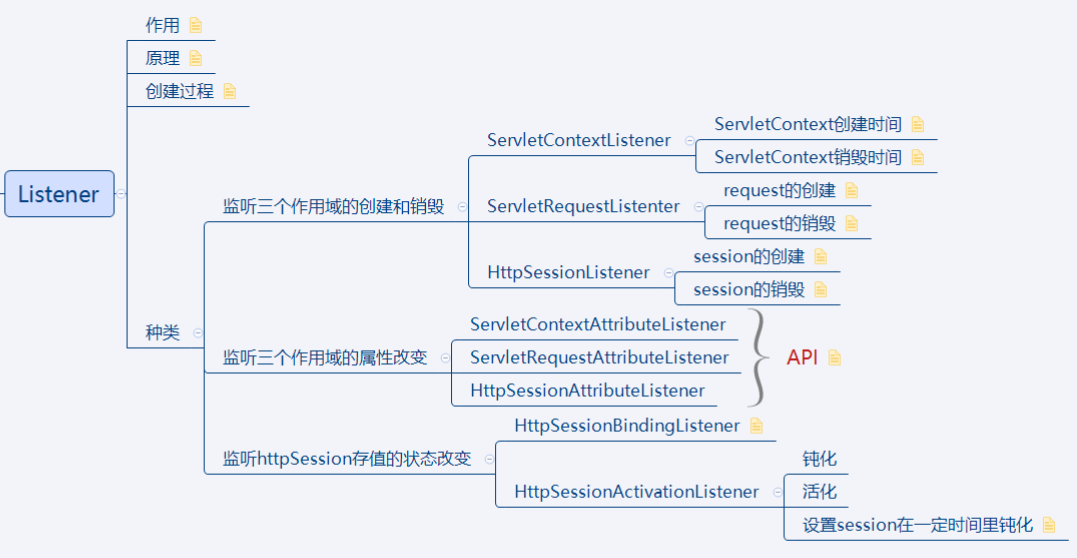
3. C：Controller，控制器。Servlet

\* 获取用户的输入

\* 调用模型

\* 将数据交给视图进行展示

1. **监听器**



**监听器**

　　1、什么是监听器？

　　　servlet规范中定义的一种特殊的组件，用来监听容器产生的事件并进行相应的处理。容器会为每个监听器只创建一个实例。

 　　　　　  容器一般会产生什么事件？

　　　　　　　　绑定数据相关事件

　　　　　　　　　　容器调用了session，request，servletContext的setAttribute， removeAttribute产生的事件

　　　　　　　　生命周期相关事件

　　　　　　　　　　容器创建销毁session，request，servletContext等。

　　2、监听器相关概念：

　　　　事件源：被监听的对象。 三个域对象 request  session  servletContext

　　　　监听器：监听事件源对象  事件源对象的状态的变化都会触发监听器

　　3、 监听器的分类：

　　　　　　第一维度：

　　　　　　　　　　按照被监听的对象分：ServletRequest域   HttpSession域   ServletContext域

　　　　　　第二维度：

　　　　　　　　　　按照监听的内容分： 监听域对象的创建与销毁的    监听域对象的属性变化的

　　　　总结如图：



　　　　　　ServletContextListener监听器的主要作用：

　　　　　　　　　1 初始化的工作：初始化对象 初始化数据 ---- 加载数据库驱动  连 接池的初始化

　　　　　　　　　2 加载一些初始化的配置文件 --- spring的配置文件

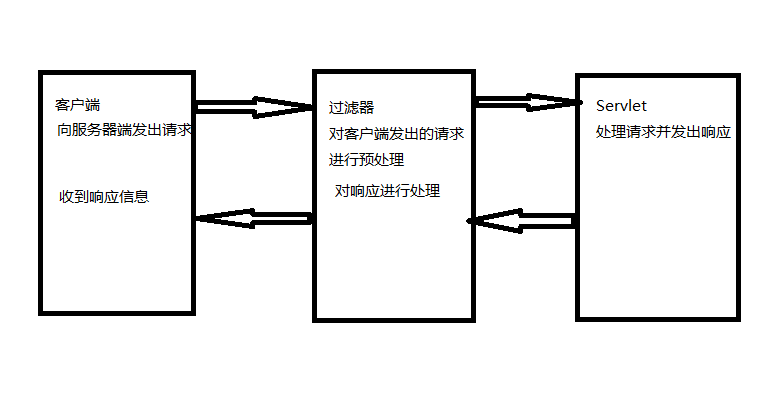
　　　　　　　　　3 任务调度：定时器----Timer/TimerTask

1. **过滤器**

**Filter，过滤器，是处于客户端与服务器资源文件之间的一道过滤网，在访问资源文件之前，通过一系列的过滤器对请求进行修改、判断等，把不符合规则的请求在中途拦截或修改。也可以对响应进行过滤，拦截或修改响应。**

**1.概念**

　 　过滤作用，对从客户端向服务器端发送的请求进行过滤，也可以对服务器端返回的响应进行处理。它使用户可以改变一个request和修改一个 response.。Filter 不是一个servlet，它不能产生一个response，但是它能够在一个request到达servlet之前预处理request，也可以在 response离开servlet时处理response。换句话说，filter其实是客户端与servlet中间的一个传递者，并且它可以对要传递 的东西进行修改。



　　注意：过滤器是用来拦截请求和响应的，不能产生响应，而servlet是用来处理请求并产生响应的。

**2.适用场合**

　　实现URL级别的权限访问控制，过滤敏感词汇，压缩响应信息等。

**3.过滤器如何实现拦截**

* 当客户端发生请求后，在HttpServletRequest 到达Servlet 之前，过滤器拦截客户的HttpServletRequest 。
* 根据需要检查HttpServletRequest ，也可以修改HttpServletRequest 头和数据。
* 在过滤器中调用doFilter方法，对请求放行。请求到达Servlet后，对请求进行处理并产生HttpServletResponse发送给客户端。
* 在HttpServletResponse 到达客户端之前，过滤器拦截HttpServletResponse 。
* 根据需要检查HttpServletResponse ，可以修改HttpServletResponse 头和数据。
* 最后，HttpServletResponse到达客户端。

**4.Filter接口**

　　Servlet API提供了一个Filter接口，编写的过滤器必须实现该接口。

**5.Filter的生命周期**

**（1）Filter接口中有三个重要的方法。**

* init()方法：初始化参数，在创建Filter时自动调用。当我们需要设置初始化参数的时候，可以写到该方法中。
* doFilter()方法：拦截到要执行的请求时，doFilter就会执行。这里面写我们对请求和响应的预处理。
* destroy()方法：在销毁Filter时自动调用。

**（2）Filter的生命周期**

　　  Filter的创建和销毁由web服务器控制。

* 服务器启动的时候，web服务器创建Filter的实例对象，并调用其init方法，完成对象的初始化功能。filter对象只会创建一次，init方法也只会执行一次。
* 拦截到请求时，执行doFilter方法。可以执行多次。
* 服务器关闭时，web服务器销毁Filter的实例对象。

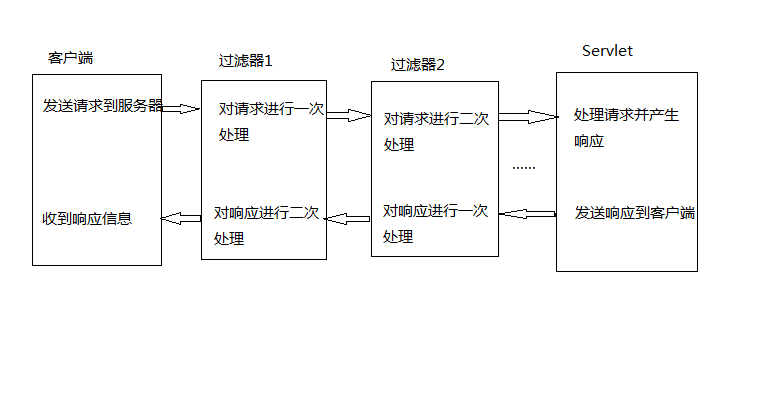
**6.Filter对象——FilterConfig**

　　用 户在配置filter时，可以使用<init-param>为filter配置一些初始化参数，当web容器实例化Filter对象，调用其 init方法时，会把封装了filter初始化参数的filterConfig对象传递进来。因此开发人员在编写filter时，通过 filterConfig对象的方法，就可获得：

* String getFilterName()：得到filter的名称。
* String getInitParameter(String name)： 返回在部署描述中指定名称的初始化参数的值。如果不存在返回null.
* Enumeration getInitParameterNames()：返回过滤器的所有初始化参数的名字的枚举集合。
* public ServletContext getServletContext()：返回Servlet上下文对象的引用。

**7.过滤器链——FilterChain**

　　一组过滤器对某些web资源进行拦截，那么这组过滤器就称为过滤器链。过滤器的执行顺序和<filter-mapping>有关（谁在前先执行谁）。



**15.AJAX**

JQeury实现方式

1. $.ajax()

\* 语法：$.ajax({键值对});

//使用$.ajax()发送异步请求

$.ajax({

url:"ajaxServlet1111" , // 请求路径

type:"POST" , //请求方式

//data: "username=jack&age=23",//请求参数

data:{"username":"jack","age":23},

success:function (data) {

alert(data);

},//响应成功后的回调函数

error:function () {

alert("出错啦...")

},//表示如果请求响应出现错误，会执行的回调函数

dataType:"text"//设置接受到的响应数据的格式

});

2. $.get()：发送get请求

\* 语法：$.get(url, [data], [callback], [type])

\* 参数：

\* url：请求路径

\* data：请求参数

\* callback：回调函数

\* type：响应结果的类型

3. $.post()：发送post请求

\* 语法：$.post(url, [data], [callback], [type])

\* 参数：

\* url：请求路径

\* data：请求参数

\* callback：回调函数

\* type：响应结果的类型

15.JDBC

步骤：

1. 导入驱动jar包 mysql-connector-java-5.1.37-bin.jar

2. 注册驱动Class.forName("com.mysql.jdbc.Driver");

3. 获取数据库连接对象 conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");

4. 定义sql

\* 注意：sql的参数使用？作为占位符。 如：select \* from user where username = ? and password = ?;

5. 获取执行sql语句的对象 PreparedStatement Connection.prepareStatement(String sql)

6. 给？赋值：

\* 方法： setXxx(参数1,参数2)

\* 参数1：？的位置编号 从1 开始

\* 参数2：？的值

7. 执行sql，接受返回结果，不需要传递sql语句

8. 处理结果

9. 释放资源