1.一个http请求包括

1. 方法-URI-协议/版本

POST /examples/test.jsp HTTP/1.1

其中HTTP有七种方法：GET、POST、HEAD、OPTIONS、PUT、DELETE、TRACE

1. 请求头信息

Accept:text/plain; text/html

Accept\_Language:en-gb

Connection:Keep-Alive

Host:localhost

User-Agent:mozilla/5.0(Windows NT 6.3; Win64; x64) AppleWebKit/537.36(KHTML, like Gecko) Chrome/37.0.2049.0 Safari/537.36

Content-Length:30

Content-Type:application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

1. 请求正文

LastName=xhsf&firstName=wu

2.一个http响应包括

1)协议-状态码-描述

HTTP/1.1 200 OK

2)响应头信息

Server: Apache-Coyote/1.1

Date: Thu, 8 Jan 2015 13:13:33 GMT

Content-Type: text/html

Last\_modified: Wed, 7 Jan 2015 13:13:12 GMT

Content-Length: 112

3)响应正文

<html>Hello World!</html>

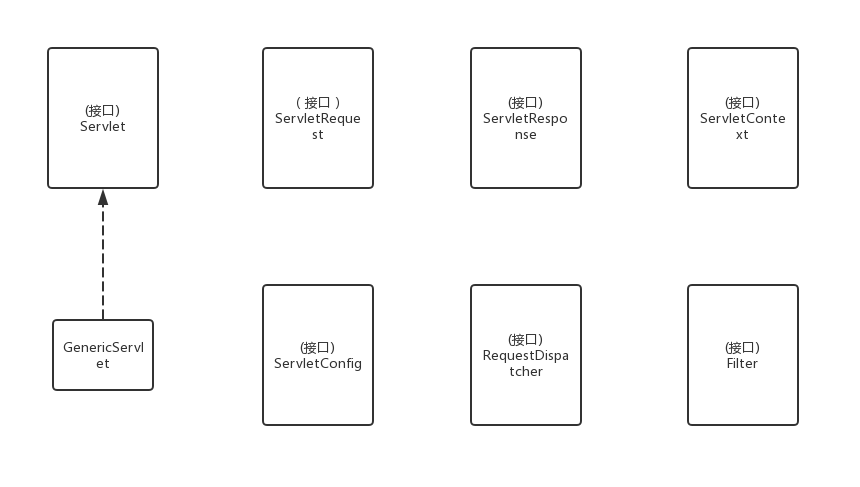
3.ServletAPI包括的4个java包

•javax.servlet 包含定义Servlet和Servlet容器之间契约的类和接口

1. init方法：当该Servlet第一次被请求时，Servlet容器会调用这个方法。

调用这个方法时，Servlet容器会传入一个ServletConfig。一般你会将ServletConfig赋给一个类级变量。

1. service方法:每当请求Servlet时，Servlet容器就会调用这个方法。
2. destroy方法：当要销毁Servlet时，Servlet容器就会调用这个方法。当Servlet容器要关闭时，一般会在这个方法中编写清除代码。
3. getServletInfo方法：会返回Servlet的描述。
4. getServletConfig方法：会返回由Servlet容器传给init方法的ServletConfig。



1. Servlet容器将Servlet类载入内存，并在Servlet实例上调用具体方法。
2. 在一个应用程序中，每种Servlet类型只能有一个实例。

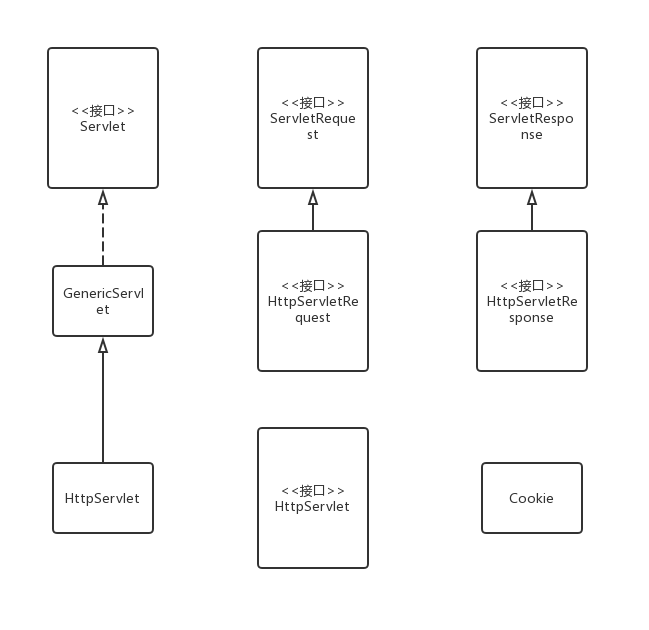
③用户请求使得Servlet容器调用Servlet的service方法，并传入一个ServletRequest实例和一个ServletResponse实例。ServletRequest中封装了当前的HTTP请求，ServletResponse表示当前用户的HTTP响应。

④Servlet容器还会创建一个ServletContext实例，这个对象封装了上下文（应用程序）的环境详情。每个上下文只有一个ServletContext。

⑤每个Servlet实例也都有一个Servlet配置的ServletConfig。

⑥ 注：Servlet实例不议使用类级变量（static），因为Servlet实例会被应用程序中的所有用户共享，除非它们是只读的，或者是java.util.concurrent.atomic包的成员。

•javax.servlet.http 包含定义HTTPServlet和Servlet容器之间契约的类和接口



HttpServlet继承了GenericServlet。使用时使用HttpServletRequest和HttpServletResponse，分别继承自ServletRequest和ServletResponse。

使用HttpServlet可以直接覆盖doGet或doPost等方法，而不必再自己覆盖service方法。

javax.servlet.annotation 包含标注Servlet、Filter、Listener的标注。它还为被标注元件定义元数据。

javax.servlet.descriptor 包含提供程序化登录web应用程序的配置信息的类型。

4. @WebServlet(name = "MyServlet", urlPatterns = {"/my"})注解，声明一个Servlet

示例：

@WebServlet(name = "MyServlet", urlPatterns = {"/my"})  
public class MyServlet implements Servlet{  
 private transient ServletConfig servletConfig;  
 @Override  
 public void init(ServletConfig servletConfig) throws ServletException {  
 this.servletConfig = servletConfig;  
 }  
 @Override  
 public ServletConfig getServletConfig() {  
 return servletConfig;  
 }  
 @Override  
 public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws ServletException, IOException {  
 String servletName = servletConfig.getServletName();  
 servletResponse.setContentType("text/html");  
 PrintWriter writer = servletResponse.getWriter();  
 writer.print("<html>Hello word" + servletName + "</html>");  
 }  
 @Override  
 public String getServletInfo() {  
 return "My Servlet";  
 }  
 @Override  
 public void destroy() {  
 }  
}

扩展：

1. initParams = {name=””, value=””}可以设置初始化参数，可以通过ServletConfig的getInitParameter方法获取参数的值。

示例：

initParams = {  
 @WebInitParam(name = "admin", value = "xhsf"),  
 @WebInitParam(name = "email", value = "xhsf@qq.com")  
}

5.WEB-INF目录下的文件只有servlet可以访问，用户不可以

6.ServletRequest接口方法详解

1. int getContentLength: 返回请求主体的字节数。如果不知道字节长度，会返回-1.
2. String getContentType: 返回请求主体的MIME类型，如果不知道类型，返回null.
3. String getParameter(String name)：返回指定请求参数的值。如果参数不存在返回null。
4. String getProtocol: 返回这个HTTP请求的协议名称和版本。

7.ServletResponse接口方法详解

1. PrintWriter getWriter: 返回一个可以向客户端发送文本的PrintWriter。默认使用ISO-8859-1编码。
2. OutputStream getOutputStream: 同上，但是发送的是二进制数据。
3. setContentType(String):设置响应类型，如”text/html”

8.ServletConfig接口方法详解

可以通过@WebServlet的initParams或者部署描述符传给Servlet的配置信息封装ServletConfig。

1. String getInitParameter(String name) ：获取ServletConfig的初始化参数的值。
2. Enumeration<String> getInitParameterNames: 返回所有ServletConfig的初始化参数。
3. ServletConfig getServletConfig：返回ServletContext对象

9.ServletContext接口方法详解

ServletContext表示Servlet应用程序。每个Web应用程序只有一个上下文。在将一个应用程序同时部署到多个容器的分布式环境中，每台java虚拟机上的Web应用程序都会有一个ServletContext对象。

可以在ServletConfig中调用getServletContext方法获取ServletContext对象

保存在ServletContext内部的Map中的对象称为属性。

1. Object getAttribute(String name):获得一个属性
2. Enumeration<String> getAttributeNames:获得所有属性的名
3. setAttribute(String name, Object object):设置一个属性
4. removeAttribute(String name):移除一个属性

9.GenericServlet抽象类方法详解

GenericServlet实现了Servlet和ServletConfig接口，并完成：

•将init方法中的ServletConfig赋给一个类级变量，以便通过调用getSerlvetConfig来获取

•为Servlet接口中的所有方法提供默认实现。

•提供方法，包围Servlet中的方法。

1. 可以调用init：方法进行初始化，而不是通过调用Servlet中的init(ServletConfig)方法，因为在GenericServlet的init(ServletConfig)实现中调用了init()方法，所以只要覆盖init()方法就可以。
2. Enumeration<String> getAttributeNames:获得所有属性的名
3. setAttribute(String name, Object object):设置一个属性
4. removeAttribute(String name):移除一个属性

10.HttpServletRequest接口方法详解

新增自ServletRequest接口的方法有：

1. String getContextPath:返回表示请求上下文的请求URI部分。
2. Cookie[] getCookies:返回Cookie对象数组。
3. String getHeader(String name): 返回指定HTTP标题的值。
4. String getMethod: 返回生成这个请求的HTTP方法名称。
5. String getQueryString：返回请求URL中的查询字符串。
6. HttpSession getSession：返回与这个请求相关的会话对象。如果没有，将创建一个新的会话对象。
7. HttpSession getSession(Boolean create): 返回与这个请求相关的会话对象。如果没有，并且create参数为true，将创建一个新的会话对象。

11.HttpServletResponse接口方法详解

新增自ServletResponse接口的方法有：

1. addCookie(Cookie cookie):给这个响应对象添加一个cookie。
2. addHeader(String name, String value):给这个响应添加一个header。
3. sendRedirect(String location):发送一条响应码，将浏览器跳转到指定的位置。

12.HTTP的4种状态保持技术

•URL重写

其中?后面的每一项称为token,形式如：url?key1=value1&key2=value2

通过在html标签中的属性href中添加token,在doxxx方法中通过判断token参数达到重定向。

示例:

<a href=’?city=london’>London</a>

<a href=’?city=paris>Paris</a>

String city = request.getParameter(“city”);

If (city != null &&( city.equals(“london”) || city.equals(“paris”))) {}

else {}

1. URL重写需要再服务端上完成，所有的连接都必须带值。
2. 某些字符，如空格，与和问号等都必须用base64编码。
3. 所有的信息都是可见的。
4. 有最大长度限制。
5. 传递到下个资源需要将值插入到链接中。

•隐藏域

同URL重写类似，但部是把值放在URL上，而是放在表单里。

因为是通过post方法，所以调用的是doPost方法

<form>

<input type=’hidden’ name=’id’ value=’1’ />

</form>

Int id = request.getParameter(“id”);

•Cookies（要熟悉Cookie类和HttpServletRequest和HttpServletResponse）

Cookies是一个信息片段，可以自动地在浏览器和Web服务器间交互。

浏览器在访问同一Web服务器时，会将之前收到的cookie一并发送。

Cookies作为HTTP header的一部分。

浏览器通常支持每个网站20个cookies。

Cookie的name不能设置为comment、discard、domain、Expires、Max-Age、Path、secure、version、和起始字符为’$’等非法字符串（忽略大小写）。

1. 创建cookies：Cookie cookie = new Cookie(name, value);
2. 可以调用setxxxx方法设置domain、path、maxAge（决定cookie何时过期）属性。
3. 将cookie发送到浏览器，需要调用HttpServletResponse的addCookie方法。

示例：httpServletResponse.addCookie(cookie);

1. 可以通过HttpServletRequest的getCookies方法获取浏览器提交的cookie，返回的是一个Cookie数组，若没有cookie返回null。

示例：Cookie[] cookies = httpServletRequest.getCookies();

1. 因为没有直接删除cookie的方法，所以删除一个cookie必须用一下过程：

@Override  
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 Cookie cookie = new Cookie("name", "");  
 cookie.setMaxAge(0); //将持续时间设置为0  
 resp.addCookie(cookie);  
}

•HttpSession

1）一个用户有且仅有一个HttpSession，并且不会被其他用户访问到。

2）HttpSession对象在用户第一次访问网站的时候自动被创建。

3）HttpSession中的值是存放在内存中的，如果内存不够会放到二级存储上。（不要在HttpSession放入太多对象或大对象）

4）HttpSession的值是任意实现Serializable接口的java对象，因为Servlet会在有必要时将这些对象放入文件或数据库。如果不支持序列化的对象，在Servlet试图序列化时将会报错。

5）HttpSession的数据不会发送到客户端，Servlet容器会为每个HttpSession生成唯一的标识，并将该标识发送给浏览器，或创建一个名为JSESSIONID的cookie,或者在URL后面附加一个名未jsessionid的参数。在后续请求中，浏览器会将标识提交给服务器。

6）默认情况下HttpSession会在用户不活动一段时间后自动过期。

7）可以通过部署配置符session-timeout元素配置存活的时间，如30，表示30分钟。

1. 可以通过调用HttpServletRequest的getSession和getSession(Boolean create)方法获取Session对象。

示例：HttpSession session = req.getSession()

HttpSession session = req.getSession(false)

1. 可以通过HttpSession的setAttribute方法将值放入HttpSession。

如果name已经存在，则会覆盖旧值。

setAttribute(String name, Object value);

1. 可以通过HttpSession的getAttribute方法取回之前放入的对象。

Object getAttribute(String name);

1. 可以通过HttpSession的getAttibuteNames方法取回之前放入的所有对象的名。

Enumeration<String> getAttributeNames()

1. 可以通过HtppSession的getId方法获取cookie中标识Session的JSESSIONID的值

String getId()

1. 可以通过HttpSession的invalidate方法强制会话过期，并清空保存的对象。
2. 可以通过HttpSession的getMaxInactiveInterval方法来查看会话多久过期。
3. 可以通过HttpSession的setMaxInactiveInterval方法来设置会话多久过期。单位为秒。若为0则永不过期。默认为30分钟。