Servlet学习笔记

B/S & CS架构 B/S 流行的原因

请求访问模型

什么是Servlet

负责处理HTTP请求和响应的技术之一。

由专门的Servlet容器（Servlet引擎）管理，与平台无关（Java的特点）

Servlet功能：

获取请求数据（显示：表单数据（浏览器或其他HTTP客户端程序）、隐式：cookies,HTTP请求头数据）

处理响应：从数据库查询而来的数据，Web Service或其他数据来源 返回给客户端，包括显示和隐式数据

Servlet生命周期（由容器负责管理）

1. 开始
2. 请求到来，判断相应的Servlet实例是否存在，如果不存在，则新建一个
3. 新建Servlet实例：导入相关类，创建实例，调用Servlet的init方法初始化实例
4. 调用Service方法处理请求
5. 调用Destroy方法销毁实例

Servlet容器：

最常见：免费且开源的Tomcat GlassFish

HTTP头信息的理解

必须支持HTTP协议 HTTPS

一个Servlet请求响应标准流程

1. HTTP客户端发起请求
2. Servlet容器解析请求，如果不合法，返回响应错误码（参见RFC 2616）
3. 如果合法，处理请求
4. 进入Servlet生命周期，控制权交给Servlet
5. Servlet处理完毕，得到响应数据，控制权交给容器
6. 处理响应数据，返回结果给客户端

参考链接：RFC 2616（在[www.ietf.org输入RFC2616](http://www.ietf.org输入RFC2616) 大天朝的需要翻墙才能Google，因此不能提供直接链接）

搭建开发环境

Step1：安装JDK 设置环境变量 具体自行搜索，网上太多了

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Step2 下载Tomcat 具体安装自行搜索，网上也很多

<http://tomcat.apache.org/>

第二节：Servlet生命周期详解

Servlet生命周期的三个关键方法

Init 初始化一个Servlet

Service 处理客户端请求

Destroy 销毁一个Servlet实例

Init方法：从Servlet容器启动到结束，只会调用一次；

调用时机点位当Servlet第一次被调用，对于以后的任何请求都不会再重新初始化

每一个新的HTTP请求来到Servlet实例上，都会产生一个新的线程

Init方法实例化的数据在该Servlet整个生命周期内有效

Public void init(ServletConfig config) throws ServletException{

//Initialization code…

}

执行完init方法以后，就有一个相应的servlet实例。Servlet接口通过该方法默认初始化了ServletConfig接口。这样，servlet可以通过该引用访问Web容器的键值对。通常，一些初始化资源的工作都会在这个方法里完成。比如数据库连接的获取。如果发生异常，将会抛出ServletException错误并且不会产生相应的实例。

Service方法：请求处理的核心方法，通过Servlet容器调用来处理客户端的HTTP请求，然后控制权在Servlet，Serlet处理之后，得到响应数据。

每一个请求都会新起一个线程调用该方法处理请求和响应。但我们不需要考虑同步的问题。因为Servlet容器会保证，每一次只有一个Servlet线程执行。

Public void service(ServletRequest request, ServletResponse response) throws ServletException,IOException {

}

Service 方法可以通过调用相应的doGet,doPost,doPut,doDelete等方法来处理相应的请求。

请求就是ServletRequest对象。Servlet实例通过处理request对象，将结果实例化成ServletResponse对象。

当发生错误时，Servlet容器将会清理该Servlet实例并产生一个错误响应。（404或503）

Servlet接口

Servlet接口的核心

两个子接口 GenericServlet和HttpServlet

DoGet和doPost方法就是HttpServlet的方法

Destroy方法：当Servlet生命周期即将终结时调用该方法；一般用于关闭数据库连接，挂起线程，将Cookie数据写入磁盘等清理工作；当Servlet的Destroy方法被调用后，该实例就被标记为可回收。

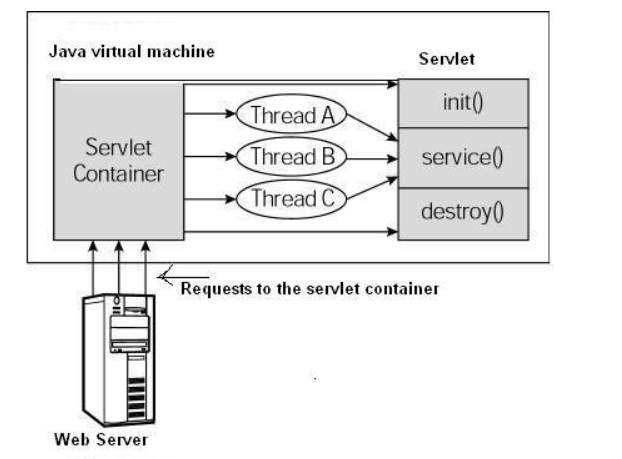
Public void destroy() {

//清理工作

}

Servlet可以通过注解或配置两种方式来描述。Servlet容器根据这些配置来实例化Servlet。一般情况下，只要不是分布式环境，servlet只会产生一个相应的servlet实例。（分布式Servlet容器的情况例外，如果一个servlet实现了SingleThreadModel接口，servlet会实例化多个实例来负载均衡请求。但每一个JVM里仍然只有一个实例）

示例图：



Demo程序：

@WebServlet("/HelloWorld")

**public** **class** HelloWorld **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** String message;

@Override

**public** **void** init() **throws** ServletException {

//该方法重载自GenericServlet抽象类 资源初始化

message = "Hello World...";

}

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

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<h1>"+ message +"</h1>");

}

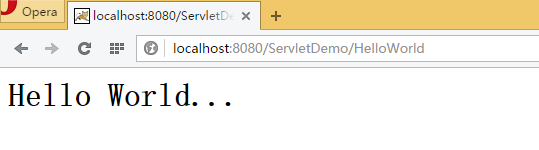
@Override

**public** **void** destroy() {

//该方法重载自GenericServlet 销毁资源

}

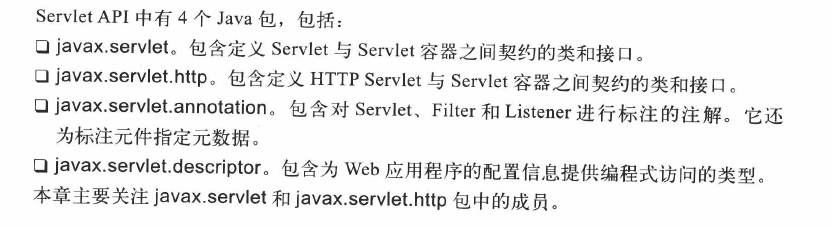
}





Servlet从2.5版本开始支持注解

Servlet API概述



第三节：表单数据

HTTP客户端的数据要提交到Servlet服务器的方式有很多种。在B/S模式下，比较常用的有Get方法和Post方法。

Get方法：

Get方法是显示的发送请求数据到服务器，一般格式是

<http://domainname/pagename?key1=value1&key2=value2> 请求发送到服务器后，HttpServlet实例将会调用doGet方法处理。

这里，？后面的键值对数据就是表单数据。之所以说这种方式是显示的，是因为这些数据在浏览器的地址栏都可以看到。



必应搜索

这种方式比较容易构造，也存在一定安全风险。（毕竟，我们不希望我们的密码在地址栏被显示出来）

因此另一种隐式的方式就是Post方法。

Post方法：

Post方法会把所有的请求数据都在浏览器后台传输过去，因此在浏览器地址栏我们是看不到表单数据的。

Servlet获取表单数据

当表单数据通过Servlet容器传递给Servlet的时候，Servlet可以通过如下方法获取：

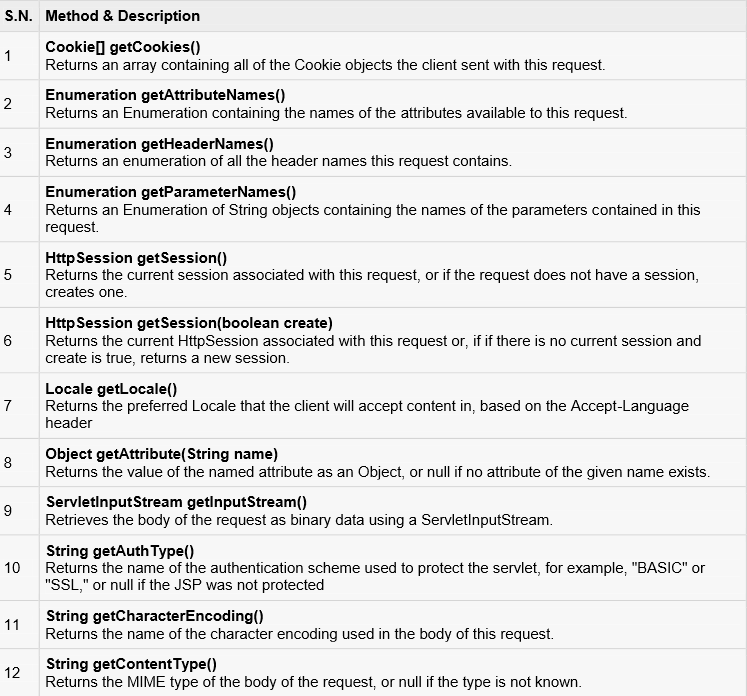
getParameter() 通过调用request.getParameter()方法获取表单key。

getParameterValues() 获取value或values，当表单参数值有多个时获得到的是一个数组。

getParameterNames() 获取keys，获取所有的key

客户端请求

当一个客户端请求到来的时候，我们可以通过HttpServletRequest的一些方法读取HTTP头



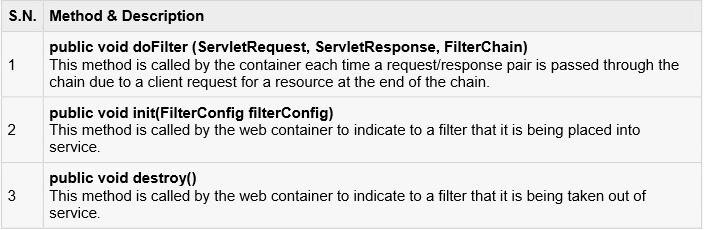
第四节：

过滤器的编写

Servlet过滤器作用：

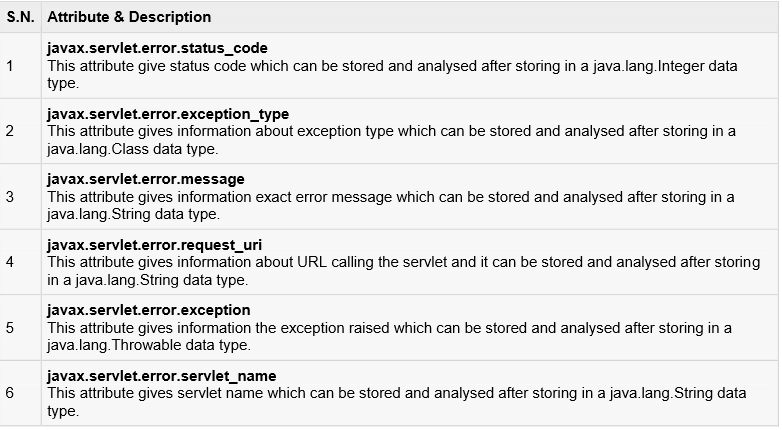
1. 拦截请求
2. 过滤响应

过滤器方法：



异常处理：





Cookie处理

Session处理

数据连接

文件上传