



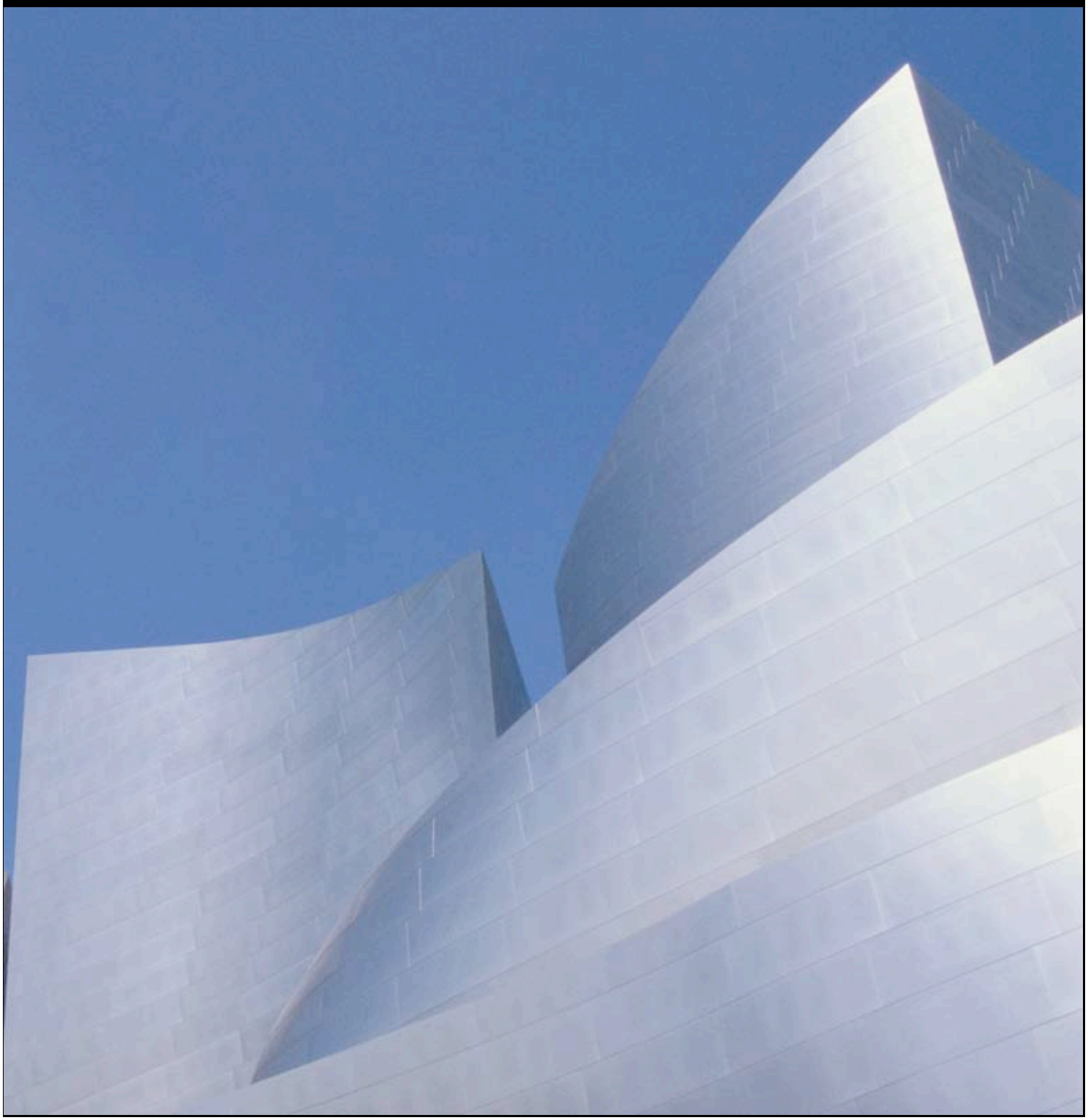
http://www.easyitedu.com

旺旺老师 JavaWeb 课程教程

# 02 Servlet 详解

**旺旺老师简介**

刘红旺，82 年生于山西，02 年来西安求学，一年后辍学从事软件开发工作，三年后投身软件教育至今。至此，旺旺拥有七年工作经验，四年软件教育经验。旺旺兴趣广泛，近期一直学习经济学与心理学。旺旺老师课程特点：通俗易懂，风趣幽默。



# 图书说明（javaSE）

城市：陕西，西安

QQ： 22713528

技术交流群：17951956

电子邮件：shudaizi@163.com

个人网站：

http://www.easyitedu.com

简单 IT 学习网，学习 IT 变简单

**说明：本教程配套视频会同期发布，欢迎大家访问旺旺老师的网站 http://www.easyitedu.com 了解详情。**

旺旺一直有逛书店的习惯，到 java 柜台总会停下来看看有什么基础书籍向学生推荐，但可惜十分满意的基本没有。于是有了下面的话：

现在市面上的 java 教材有两种：

一种是学院派老师编写的，他们是主流，你看到十本 java 书，有九本半是这样的。他们的作者拥有让人敬仰的称谓，如某某教授，某某专家；他们写的书大都是大部头，动辄上千上万；他们文风严谨，遣词专业；他们案例深奥，让人深思。总之，我很敬仰他们，因为我当初也是读着这样的书学 java 的。但现在看来，这样的书并不适合入门，情节大都是这样的，当我们怀着莫大的热情准备开始学习 java, 买了一本久仰的《java 编程宝典》（有无此书，无从得之，是旺旺杜撰的书名），它很厚很重很专业，你用毅力恒心支持看了几十页，才相信，java 的确是难学，后来，就不怎么看它了。直到过了很多天，也许你已从事 java 方面的开发工作，一天，从书堆里无意的见到它，拍拍灰尘看看，够厚，就作为工具书备用吧。

还有一种是像旺旺这样草根阶层（好听点叫实战派）编写的，这样的书凤毛麟角，因为大家都在忙于工作，不像旺旺这样打了鸡血精力充沛的无法发泄。他们的称谓一般都不匝地，不谈也罢；他们的书也不怎么厚，能上千页的更少之又少；他们文风朴实，贴近大众；他们案例生动，通俗易懂，风趣幽默；你可以当技术类书籍来学习它，甚至当消遣类散文来阅读它。OK, 如果你初学 java, 需要的就是一本这样的书，所以**《旺旺老师 JavaSE 教程》的目标人群是：初学 java 的读者**。

还有如果您已认真完善的学习过 java, 或已从事 java 方面的开发工作，那也可以看看本书的部分章节，旺旺一些幽默风趣的案例也许能给您带来一些惊喜, 一些实用技术的深入讲解能给您带来些许收获。

所以《旺旺老师 JavaSE 教程》的**理想目标是： 让新手当技术类书籍来学习它，让老手当消遣类散文来阅读它**。想让所有人，在快乐中学习，在快乐中成长，在快乐中工作。学习 java 可以是件很轻松的事情，让所谓高深的技术平民化，让所谓复杂的技术简单化，这是我的奋斗目标, 也会一直为此努力。

# 图书说明（javaWeb）

旺旺一直想写本关于 Java 技术的书籍，但由于此人过于懒惰，毅力也相当的差劲，所以总是想的比做的多，想的比做的好。直到半年前（2009 年十二月份）的某个月黑风高的晚上，像往常一样旺旺洗完澡躺在床上准备睡觉，但这次与往常不一样的是他竟然没有想苍井空小泽圆小泽玛利亚饭岛爱这些天使，而是关于写书的事情，也就是那么简简单单的一想，突然黑天一声霹雳，雷想了，雨下了，旺旺也突然醒了，那感觉，说的严重点是感觉找到了人生的真谛，生活的意义，那就是他终于打算愿意好好的写那本本早该两年前开始写的 Java 书籍了。

其后的两三个月，旺旺干的热火朝天，并乐此不疲。从找模板，考虑技术大纲，总结各种案例，到编写，后来又找视频录制软件，录制视频，旺旺激动万分，日子过的也是相当充实，也不感觉无聊了，同时腰也不酸了，腿也不停了，走路也有劲了，以前感觉大把大把的时间现在反而感觉不够用了，总之，旺旺很享受那两三个月的时光・・・・・・

遗憾的是好景不长，直到现在，半年多过去了，旺旺的 JavaSE 也只写了三分之二的内容，根本原因还是旺旺这两个月又懒惰了，虽然也断断续续的写，但速度明显减慢，旺旺很生气，有时候也在不停的骂自己，可时间又在我的漫骂声中悄悄过去了・・・・・・

当然这期间也写了部分 JavaWeb 的内容，这里案例的讲解等非常少，不是以书籍的形式编写，可以暂时作为提纲，供大家复习总结所用！最后希望自己一直在状态，少贪玩，快速把 JavaSe 写完出版，也希望 JavaWeb 的知识能坚持。

实在也没什么好写的，就这样吧！（其他：其实苍空小泽圆小泽玛利亚饭岛这些人中旺旺只知道前些天来中国的苍井空，其它是现搜索的，由这些名字引出的您的一切行为后果均与旺旺无关，特此声明）。

写书宣言《大腕》版：出书,书本厚度至少要一千页；封面要复古，最好弄副古代的山水画，千万不能出现计算机英文单词；名字什么火就用什么，比如《爪哇那些事儿》《编程往事：爪哇风云 20 年》《爪哇升职计》《与爪哇同居的日子》《爪哇囧途》等；序言一定要由名人代写，不是大公司大总裁就是大出版社大编辑，总之一定要“大”，你要是随便找一朋友写都不好意思出版；内容要全，要广，上知天文，下知地理，把古代什么儒家啊道家啊法家啊墨家啊那些思想能加的全给他加上，读者一看，动不动就是孔子说老子说墨子说，倍有面子； 问：那这样读者还能看懂吗？答：看不懂？那就对了。你要了解现在读者的心态，翻下能看得懂的书是作者没水平，看不懂的才是权威，所以我们写书的口号就是：不求易懂，但求最难。

# 旺旺老师软件工程师实训

今天没课，早上陪从安徽来的姚伟同学找房子。他是第一次来西安，人生地不熟，在网上看了我的文章视频知道开培训班了所以来学习。因为还是学生，再加上就他一个人住，所以最后选在了一个离上课地点不远的民房，每月 300，室内独立卫生间，我们两个人都很满意，所以当场定下，搬家，入住。

这又勾起了自己当初上学时在村子住民房的经历，一个人住六楼的小房间，上个厕所还得跑到一楼，比他现在的房子差多少倍。而现在坐在自己的房子里写下这些，房子前年入住的，当初买房没要父母一分钱，这里我没有什么吹嘘的意思，只想向他还有其他同学说我经常说的那句话“人，应该为自己定个现实的目标，并以此为动力奋斗坚持。”

当初住在白庙村 76 号 6 层 4 号时，看到对面高层的万家灯火，我就告诉自己，一定要努力，要在这所城市安家落户，有自己的房子，06 年中旬，我实现了自己订立的目标。

可是当自己实现了当初的目标后，随后的两年，我迷茫了，迷失了，只有每月十五号工资卡上一大堆数字让我意识到自己的存在。期间虽然也有自己做培训的意愿，但惰性总是围绕着我，安逸的生活总腐蚀着我。直到去年冬天的某个晚上，我突然醒了，感觉自己不能这么生活下去，应该有新的目标并为此奋斗，从那一刻开始，感觉自己又找到了人生的真谛，生活的意义，以前大把大把的时间感觉也不够用了。

培训班没想到这么快能开起来，去年的计划是等至少两本书（《JavaSE 基础》与《JavaWeb》）写好后再开始，但有一位带过的学生的同学想学，又被各培训中心高昂的费用拒之门外，在他们的鼓励下很快就开始了，辞了原来的工作，拿着比原来低很多的收入，

匆忙但稳健的开始了。没想到在基本没做市场的情况下招生却是出奇的好，虽然也还没达到自己以前收入，但我却是很满意了，因为看好未来，并相信天道酬勤。

一般的招生简章都是课程体系什么的，我这不打算给您列出，只告诉您，该讲的都会讲到，并且在课程结束前一个月联系一些公司（旺旺老师的很多学生都已经是项目经理或技术骨干，负责招聘），专门培训他们需要的技术，只想说：我要长久做，一定给你们最好最适合的课程。

**培训时间：五个月，每周五个全天课。培训地点：陕西 西安。培训费用：6800 元。联系方式：13772086697 QQ:22713528 旺旺老师。**

其它：如果您在外地，如果您也没有在外地一个人独立生活经验，那至少两个人来，否则旺旺不能接受，这是对您负责也是对我自己负责。虽然我不排斥高中生，因为旺旺老师大学也没毕业，但请您是个努力勤奋的人。谢谢。

# 第二章：Servlet 详解

为

## 2.1：SERVLET 是什么

**宏观定义**：Servlet 是 SUN 制定的用 Java 开发 Web 应用程序的规范，技术，标准。现在使用的是 Servlet2.5。

**微观定义**：是一个部署在 WebServer 中可以被客户端访问调用（可以处理客户端请求）的 Java 类。

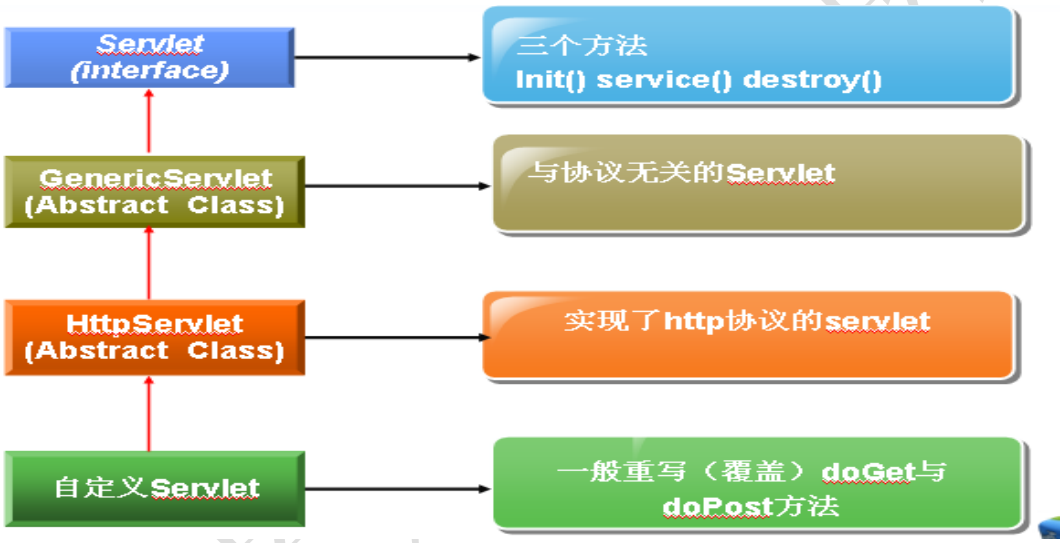
Servlet 类与普通 Java 类的区别：

| Servlet | 普通 Java 类 |
| --- | --- |
| 继承 HttpServlet | 一般继承 Object 类 |
| 必须在 WebServer 中才能运行 | 不需要 WebServer 支持也能运行 |
| 可以处理客户端请求 | 不能处理客户端请求 |

| | |
|---|---|
| 启动不需要 main 函数 | 启动需要 main 函数 |
| 由容器创建（new）实例 | 一般我们自己创建实例 |

## 2.2：SERVLET 的等级结构

一般我们继承 HttpServlet，其实我们可以继承 GenericServlet 或者实现 Servlet 接口，如果这样，就需要程序员做很多底层的工作，而这些工作 HttpServlet 已帮我们做好，所以 Sun 建议你继承 HttpServlet。




## 2.3：SERVLET 的生命周期

事物的生命周期指的是从事物的产生，生存，发展到消亡的过程。在编程中有变量的生命周期，对象（类的实例）的生命周期。

### 2.3.1：Servlet 接口中与生命周期有关的方法

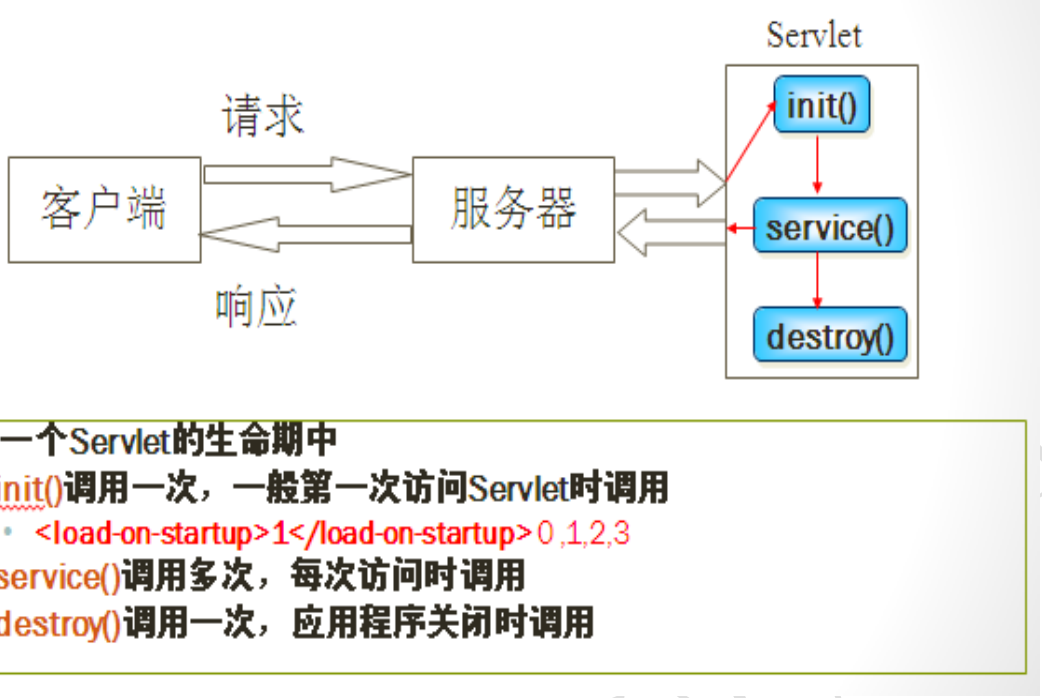
Servlet 接口中总共定义了五个方法。

```
package javax.servlet;

import java.io.IOException;

public interface Servlet {
    public void init(ServletConfig config) throws ServletException;
```

```
    public void service(ServletRequest req, ServletResponse res)
    throws ServletException, IOException;

    public void destroy();

    public ServletConfig getServletConfig();

    public String getServletInfo();

}
```

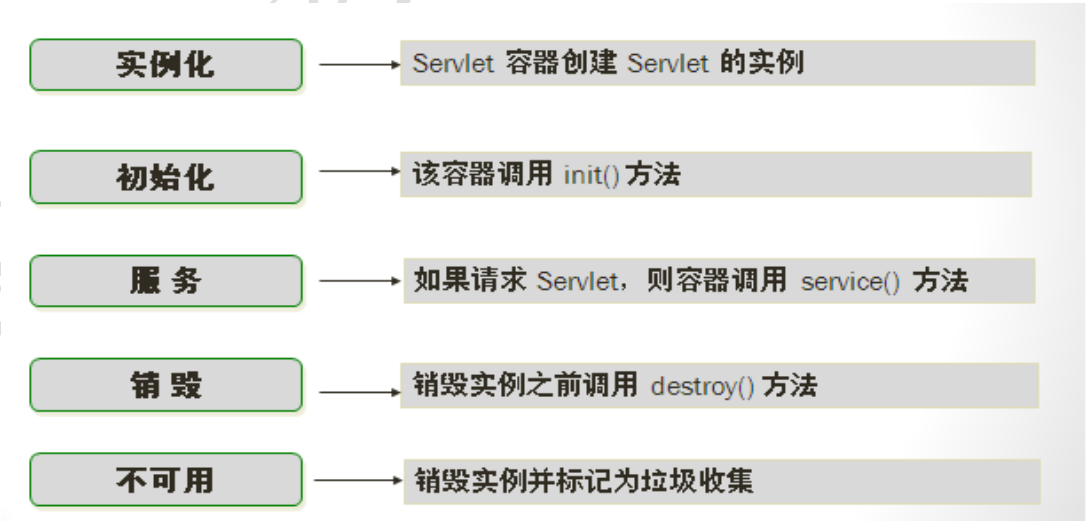
其中重要的方法有三个，案例代码：

```
package com.wangwang;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServlet;

@SuppressWarnings("serial")
public class SmzqServlet extends HttpServlet {
	public SmzqServlet() {
		System.out.println("-----------------SmzqServlet---------------");
	}

	@Override
	public void service(ServletRequest request, ServletResponse response)
		throws ServletException, IOException {
			System.out.println("----------------service--------------");
	}

	@Override
	public void destroy() {
		System.out.println("-----------------destroy-----------------");
	}

	@Override
	public void init(ServletConfig servletConfig) throws ServletException {
		System.out.println("-----------------init-----------------");
	}

}
```




- init()调用一次，一般第一次访问 Servlet 时调用。当加上<load-on-startup>1</load-on-startup>时，init 在 web 应用程序启动时调用，多个调用顺序 0,1,2,3。默认调用有参数的，除非只有无参数的。
- service()调用多次，每次访问时调用。也有可能调用 0 次。
- destroy()调用一次，应用程序关闭时调用（服务器关了，应用程序一定关，程序关闭，服务器不一定关）

## 2.3.2: Servlet 生命周期




1. 实例化，或称创建，new,分配内存空间。由容器完成。

2. 初始化，调用 init 方法，完成初始 servlet 要使用的资源，给变量赋初值等工作。

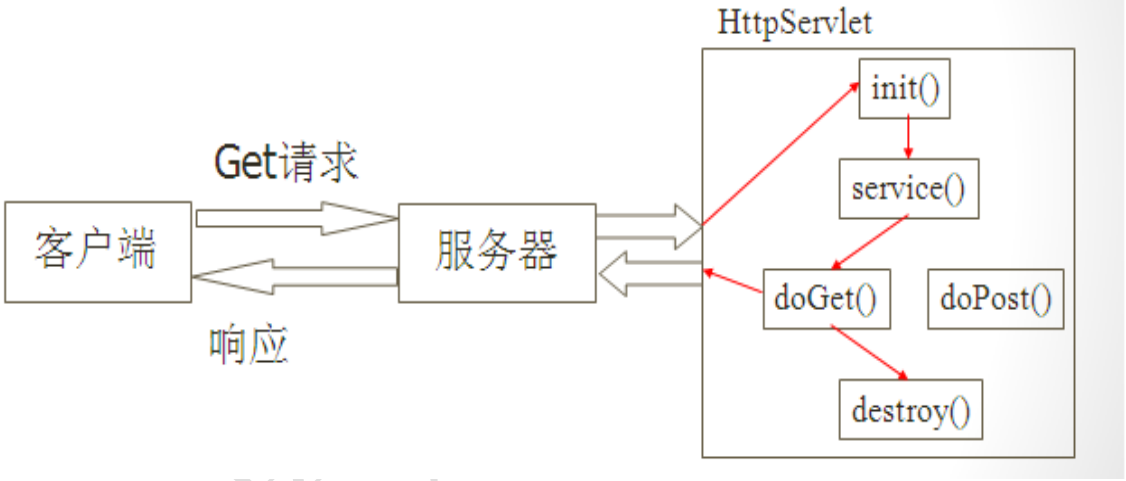
3. 服务，调用 service 方法

4. 销毁，destroy,释放初始化占用的资源

5. 不可用，垃圾回收车收集内存。

### 2.3.3: 为什么不重写 Services

一般我们编写的 Servlet 不覆盖 service 方法，而是重写 doGet 与 doPost 方法，因为 HttpServlet 中已经重写了 service 方法,并且根据请求类别调用相应的 doGet 与 doPost 方法。




```
protected void service(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException{
      String method = req.getMethod();

      if (method.equals(METHOD_GET)) {
            doGet(req, resp);

      } else if (method.equals(METHOD_POST)) {
            doPost(req, resp);
      }
}
```

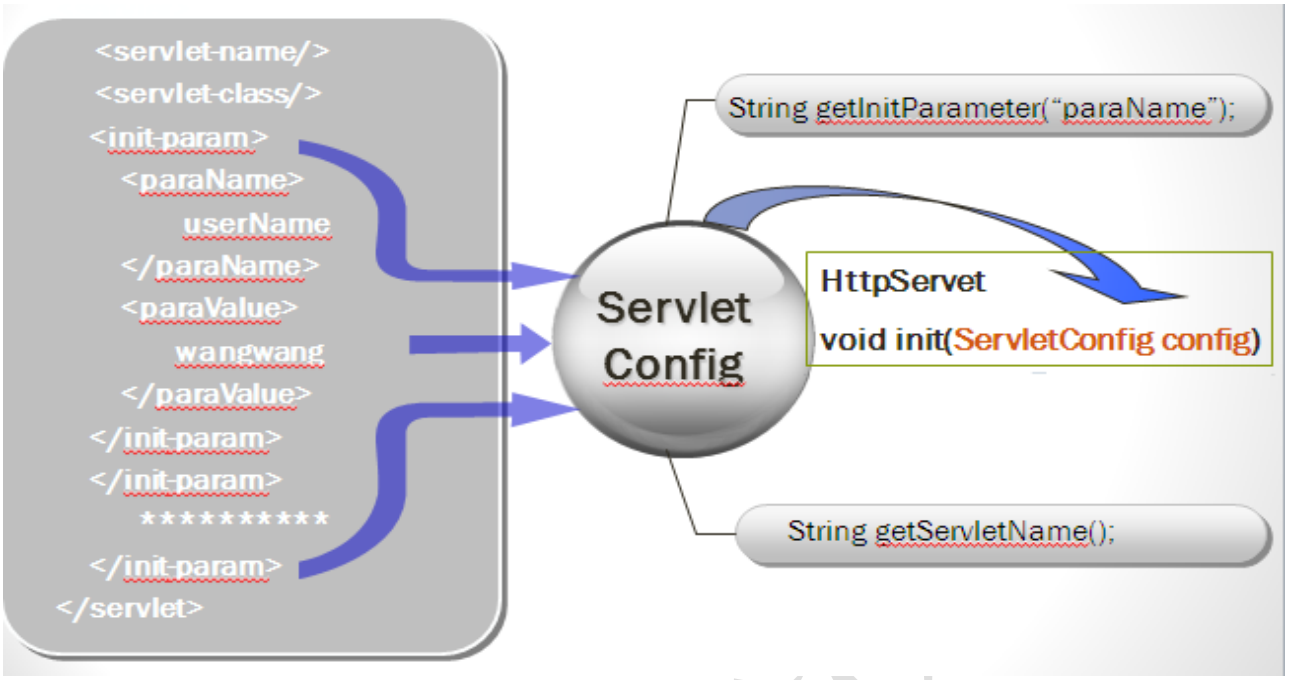
### 2.3.4: ServletConfig

封装一个<servlet>节点配置的所有信息，有三个重要方法：

servletConfig.getInitParameter("userName")：得到参数值

servletConfig.getServletName() ：得到 servletName

servletConfig.getServletContext():得到 ServletContext 类的实例 application 对象。




## 2.4： REQUEST 详解

Request 表示请求,其最重要的一个功能是封装客户端请求参数，它由 WebServer 生成（我们说 WebServer 对 WebApp 提供支持，比如生成 request），我们可以通过 request 的一些方法得到这些请求参数的数值。

### 2.4.1： 客户端如何发出请求

请求参数的几种表现：

1， form 表单内的控件,包括隐藏表单域

2， 使用 url 重写的参数名称与值语法 servletPath?参数名 1=参数值 1&参数名 2=参数值 2&参数名 3=参数值 3

### 2.4.2： 得到请求参数

当 form 表单是 get 请求时调用 doGet 方法，post 请求调用 doPost 方法。默认是 get 请求。

单个数值的请求：

```
String value = request.getParameter(“pName”);
```

多个数值的请求：

String[] values = request.getParameterValues(“pName”);

```
<input name="aihao" type="checkbox" value="chi" checked>吃
<input name="aihao" type="checkbox" value="he">喝
<input name="aihao" type="checkbox" value="piao">嫖
<input name="aihao" type="checkbox" value="du">赌
<input name="aihao" type="checkbox" value="chou">抽
<input name="aihao" type="checkbox" value="keng">坑
<input name="aihao" type="checkbox" value="meng">蒙
<input name="aihao" type="checkbox" value="guai">拐
<input name="aihao" type="checkbox" value="pian">骗
<input name="aihao" type="checkbox" value="tou">偷
```

服务器端代码：

```
String aihao[] = request.getParameterValues("aihao");
if (aihao == null) {
      System.out.println("悲哀，真是悲哀，你这人竟没爱好");
} else {
      System.out.print("您有[" + aihao.length + "]项爱好，包括");
      for (int i = 0; i < aihao.length; i++) {
          System.out.print(aihao[i] + ", ");
      }

}
```

### 2.4.3: Request 的其它重要方法

1. request.getContextPath()：得到 web 应用程序名称
2. request.getServletPath()：如果请求的是 servlet,那么返回的该 servlet 的 url-pattern,如果是 jsp,返回 jsp 文件名。
3. request.getRequestURI() = request.getContextPath() + request.getServletPath()
4. request.getRequestURL()：请求的全部路径，绝对路径
5. request.getMethod()：请求类别
6. request.getRemotePort()：客户端请求端口
7. request.getRemoteAddr()：得到客户端的 IP 地址
8. request.getRealPath(“/”):得到当前 Web 应用程序的绝对路径
9. request.getHeader("User-Agent"):得到客户端浏览器的信息
10. request.getRequestDispatcher(“<url-pattern>”):得到请求转发对象

### 2.4.4：开发表示层框架常用到的 request 的一个方法

request.getParameterNames()：得到客户端所有请求参数名称的集合

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

     System.out.println("表示层框架底层常用的一个方法");
          Enumeration<String> em = request.getParameterNames();
          while (em.hasMoreElements()) {
                String paraName =  em.nextElement();
                String[] values = request.getParameterValues(paraName);
                if (values == null) {
                      System.out.println("请求参数" + paraName + "没有值");
                } else {
                      System.out.print("请求参数" + paraName + " = ");
                      for (int i = 0; i < values.length; i++) {
                            String value = values[i];
                            System.out.print( value + "  ");
                      }
                      System.out.println();
                }
          }
}
```

## 2.5：请求转发与重定向

现在我们开发的程序大多客户端发来了请求，由一个 Servlet 处理完之后马上给客户端应答，但真正的开发中，一个 Servlet 往往处理不了所有业务，我们需要多个 Servlet 协同工作处理，那么这里就有两个问题：第一，如何由一个 Servlet 到另外一个 Servlet;第二，这两个 Servlet 如何传递数据，或者说两个 Servlet 之间如何通信。由此，我们引出在开发 web 应用时非常重要的两个概念：请求转发与重定向。

### 2.5.1:请求转发

特点：可以传递 request 中的数据（包含 parameter 与 attribute 两方面数据），或者说 request 中的数据不会丢失；只能在一个 webApp 内转发。

第一个 Servlet 代码

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
          System.out.println("白大夫得到客户端的请求参数:");
          System.out.println("name = " + request.getParameter("name"));
          System.out.println("money = " + request.getParameter("money"));
```

```
            System.out.println("开始看病");
            request.setAttribute("bm", "禽流感");
            //出入下一个Servlet的<url-pattern>
            RequestDispatcher rd =
                    request.getRequestDispatcher("/madaifu");

            rd.forward(request, response);
      }
```

第二个 Servlet 代码：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
      System.out.println("马大夫 得到客户端的请求参数:");
      System.out.println("name = " + request.getParameter("name"));
      System.out.println("money = " + request.getParameter("money"));
      System.out.println("开始看病");
      //带到前一个大夫的诊断结果
      String bm = request.getAttribute("bm").toString();
      System.out.println(bm);
      //可以修改
      request.setAttribute("bm", "h1n1");
      //也可以删除
      //request.removeAttribute("bm");
      //还可以接着请求转发到下一个servlet
      RequestDispatcher rd = request.getRequestDispatcher("/lidaifu");
      rd.forward(request, response);

}
```

### 2.5.2:重定向

特点：不能传递 request 中的数据（包含 parameter 与 attribute 两方面数据），或者说 request 中的数据会丢失；重定向可以定向到其它应用程序。

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
      System.out.println("白大夫得到客户端的请求参数:");
      System.out.println("name = " + request.getParameter("name"));
      System.out.println("money = " + request.getParameter("money"));
      System.out.println("开始看病");
      request.setAttribute("bm", "禽流感");
      //出入下一个Servlet的<url-pattern>
      //response.sendRedirect(request.getContextPath() + "/madaifu");
      response.sendRedirect("/fuke/huangdaifu");

}
```

### 2.5.3:请求转发与重定向的区别

```
RequestDispatcher disp= request.getRequestDispather("\servlet2");

disp.forward(request, response);

String path = request.getContextPath();

reponse.sendRedirect(path +“servlet2”);
```

1，请求转发可以传递数据（数据不会丢失），重定向不能传递数据（数据会丢失）。

2，请求转发地址栏 url 路径不变，重定向 url 路径改变。

3，请求转发不需要加 webApp 名称，重定向要么不加/,要么加上 webApp 名称。

4，请求转发只能在一个 web 应用程序内部转发，重定向可以到其他应用程序。

5，请求转发后面调用的方法类别与前面相同，而重定向全部会变为 get 请求，与之前的没关系。

### 2.5.4:再看 request 对象

WebServer 提供对 WebApp 的支持：帮我们管理 Servlet 的生命周期；帮我们生成 request 的实例。下面我们深入 request:

| parameter | | attribute | |
|---|---|---|---|
| 参数名（String） | 参数值（String） | 参数名（String） | 参数值（Object） |
| name | Zhangsan | bingming | h1n1 |
| money | 1000 | | |
| | | | |

request 中 parameter 与 attribute 的区别：

1,parameter 是客户端传过来的（client 请求参数）数据，attribute 是在 Server 端放入的数据。

2,parameter 数据是只读的，attribute 是可读写的，parameter 相关的方法有 request.getParamenter();与 attribute 相关的方法又 setAttribute(String,Object)，getAttribute(String)，removeAttribute(String)

3,getParameter 返回的是 String 类型，而 getAttribute 返回的是 Object 类型，需要强转。

## 2.8：复习

1. Servlet 类与普通 Java 类的区别
2. Servlet 的等级结构
3. Servlet 接口定义的方法
4. Servlet 生命周期
5. init 方法的参数 ServletConfig
6. 为什么重写 doGet 与 doPost
7. 得到客户端多个名称相同的请求参数的值
8. 得到客户端请求参数名
9. request 其它方法
10. requst 中 parameter 与 attribute 的区别
11. 请求转发与重定向



| 版本 | 修改内容 | 时间 |
| --- | --- | --- |
| V1.0 | 创建 | 2010－08－30 |