---
1 Servlet

一、Servlet概述
是由SUN公司提供的动态Web资源开发技术
Servlet本质上是一段Java程序

和之前的Java程序不同的是，Servlet程序中没有main函数，不能独立运行。

必须放在Servlet容器（容器就是服务器，比如tomcat）中运行，由容器调用才可以执行！！
所以Servlet是符合一定规范的Java程序
(要实现一个Servlet接口)

二、开发Servlet
1、开发Servlet程序的步骤
(1)写一个类，实现一个Servlet接口或者间接实现Servlet接口的子类
GenericServlet、HttpServlet
(2)将编译后的Servlet类,放在Web应用中,并在web.xml文件中配置Servlet对外访问的虚拟路径

2、使用Eclipse开发一个Servlet程序
2.1.创建一个Web项目(工程)

2.2 创建一个Servlet程序，并实现该程序

2.3 通过Eclipse启动tomcat服务器并运行Servlet

2.4.Servlet在web.xml文件中的配置

三、Servlet继承关系
Servlet接口 -- 顶层接口，声明了一个Servlet程序该具备哪些方法
|
|-- GenericServlet抽象类，实现其中大部分的方法，但是有一个方法没有实现，就是service方法，该方法需要开发人员自己来实现(因为该方法是处理请求的核心方法.)
|
|-- HttpServlet类，继承了GenericServlet类，继承了父类中的方法，并且实现了service方法，在service方法中判断请求方式，根据不同的请求方式调用不同的doXxx方法(如果是GET提交，调用doGet方法，如果是POST提交，调用doPost方法)

四、Servlet调用过程
1、Servlet调用过程
<<参见Servlet调用过程图解>>

2、Servlet生命周期
Servlet在第一次被访问时创建实例（对象），创建之后，会立即调用init方法进行初始化的操作。之后该Servlet实例会一直留在服务器的内存中，为后续的请求进行服务。
每次有请求访问Servlet，都会调用service方法进行处理。
直到服务器关闭，或者Web应用被移出了容器，随着Web应用的销毁，Servlet实例也会跟着销毁，在销毁之前，会调用destroy方法进行善后的处理。

五、修改Servlet模板
参考课件

---
2 Request

一、Request对象简介
request是代表Http请求信息的对象
在服务器调用service方法处理请求之前，会创建出代表请求信息的request对象，将所有的请求信息封装在request对象中，在service方法处理请求执行的期间，如果需要获取任何的请求信息（比如请求参数、请求头等信息）就可以通过request对象进行获取

```
String username = request.getParameter("username");
String password = request.getParameter("password");
String password = request.getParameter("password");
String host = request.getHeader("Host");
...
```

二、Request对象的继承关系

```
ServletRequest接口,定义了一个request对象所具备的功能
        |
        |-- HttpServletRequest接口,继承了ServletRequest接口,并且添加了部分和Http协议相关的方法
                    |
                    |-- Xxx类 request对象
```

三、Request对象的功能

1、获取客户端的基本信息、获取请求头信息

```
//通用的获取请求头信息的方法
String headerValue = request.getHeader(String headerName);

//获取客户机的IP地址
String ip = request.getRemoteAddr();
//获取请求方式(GET、POST)
String method = request.getMethod();
//获取Web应用的虚拟路径(/day08_ys)
String path = request.getContextPath();
```

2、获取请求参数!!!

```
//根据请求参数的名称获取对应的值
String value = request.getParamter(String name);

//根据请求参数的名称获取对应的多个值组成的数组
String[] vs = request.getParameterValues(String name);
```

!!!在获取中文参数时可能会出现乱码问题:
如果提交方式为POST提交,中文参数必然会出现乱码, 解决方式为:

```
request.setCharacterEncoding("utf-8");
```

注意: 这行代码必须放在任何获取参数的代码之前执行

如果提交方式为GET提交, tomcat8.0及以后的版本没有乱码问题, 但是如果是tomcat7.0及以下的版本, GET提交也会有乱码。解决方式为, 在服务器的server.xml文件中的Connector标签上添加一个 URIEncoding="UTF-8" 属性, 也可以解决GET提交的乱码!!

```
<Connector port="8080" protocol="HTTP/1.1"
       connectionTimeout="20000"
       redirectPort="8443"
       URIEncoding="UTF-8"
        />
```

3、实现请求转发和作为域对象使用!!!

请求转发和域对象经常配合着使用，实现将数据带到目的地！！

3.1.实现请求转发

请求转发：是同一个Web应用内部两个资源之间的跳转
实现转发:

```
request.getRequestDispatcher("/xx").forward(request, response);
```

转发的特点:
(1)转发前后浏览器地址栏地址不会发生变化
(因为转发是服务器内部的跳转,浏览器看不到)
(2)转发前后是同一个请求, request请求对象也是同一个。
(3)转发只能是同一个Web应用内部的资源跳转
不能是不同的Web应用
(4)在转发时, 转发的路径只需要写资源的路径(第三部分路径),前面的路径都不需要写,因为服务器会自动添加Web应用的访问路径

3.2.作为域对象使用

域对象：利用自身所带的Map集合，在指定的范围内共享数据的对象，就是域对象（request）

域对象提供的操作map集合的方法

```
//往域中添加一个域属性
request.setAttribute(String name, Object values);
//从域中获取一个属性值
```

```
request.getAttribute(String name);
//删除域中的一个属性
request.removeAttribute(String name);
```

域对象的三大特征:
(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: 在一次请求中
(3)主要功能: 带数据到目的地(比如将RequestDemo3中的个人信息通过转发+域对象带到JSP取出并显示在网页上)

---
3 Response

一、Response对象简介
代表Http响应的对象
在服务器调用service方法处理请求之前, 会创建出代表响应的response对象, 在service方法执行期间, 如果有任何数据要发送给浏览器, 可以将数据先写入response缓冲区中, 等方法执行结束, 服务器会从response中获取数据, 按照http协议的格式发送给浏览器!!

二、Response对象的继承关系

```
ServletResponse接口,定义了一个response对象所具备的功能
        |
        |-- HttpServletResponse接口,继承了ServletResponse接口,并且添加了部分和Http协议相关的方法
                    |
                    |-- Xxx类 response对象
```

三、Response对象常用的方法

```
response.getOutputStream();
response.getWriter();

response.sendRedirect("xxx");

response.setHeader("Refresh", "3;url=/xxx/index.html");
```

四、Response对象的功能
1、向客户端浏览器发送数据

```
response.getOutputStream();
response.getWriter();
```

通过字符流向浏览器发送中文数据可能会出现乱码问题:

```
//通知服务器使用utf-8发送数据, 同时也通知浏览器使用utf-8接收数据
response.setContentType("text/html;charset=utf-8");
```

2、实现重定向
重定向和请求转发都可以实现资源的跳转
重定向不限制两个资源是否是同一个Web应用

实现重定向:

```
response.sendRedirect("/xxx/xx");
```

重定向的特点:
(1)两次请求,两次响应
(2)地址栏地址会发生变化
(3)重定向前后的request对象不是同一个, 不能带数据过去
(4)重定向不限制两个资源是否是同一个Web应用
(5)重定向时, 服务器不会帮我们添加Web应用的虚拟路径。

!!转发和重定向都可以实现跳转, 什么时候该使用转发或重定向?
(1)如果跳转前后(A-->B)要带数据到目的地, 就需要使用转发(加上域对象).
(2)如果希望跳转前后的地址栏地址不要发生变化, 使用转发!
(3)如果希望跳转前后的地址栏地址会发生变化, 使用重定向!

3、实现定时刷新