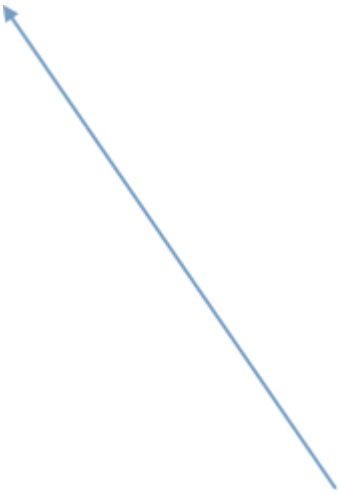
一、什么Servlet？

servlet 是运行在 Web 服务器中的小型 Java 程序（即：服务器端的小应用程序）。servlet 通常通过 HTTP（超文本传输协议）接收和响应来自 Web 客户端的请求。

1.1、编写一个servlet程序：

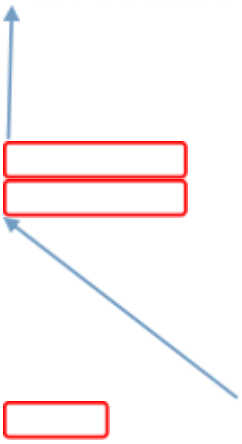
a、写一个java类，实现servlet接口

```
public class ServletDemo1 implements Servlet{


    //接收用户请求，并做出响应
    public void service(ServletRequest req, ServletResponse res)
            throws ServletException, IOException {
        res.getWriter().write("hello ServletDemo1");

    }
```



b、修改web.xml文件，给servlet提供一个可访问的URI地址

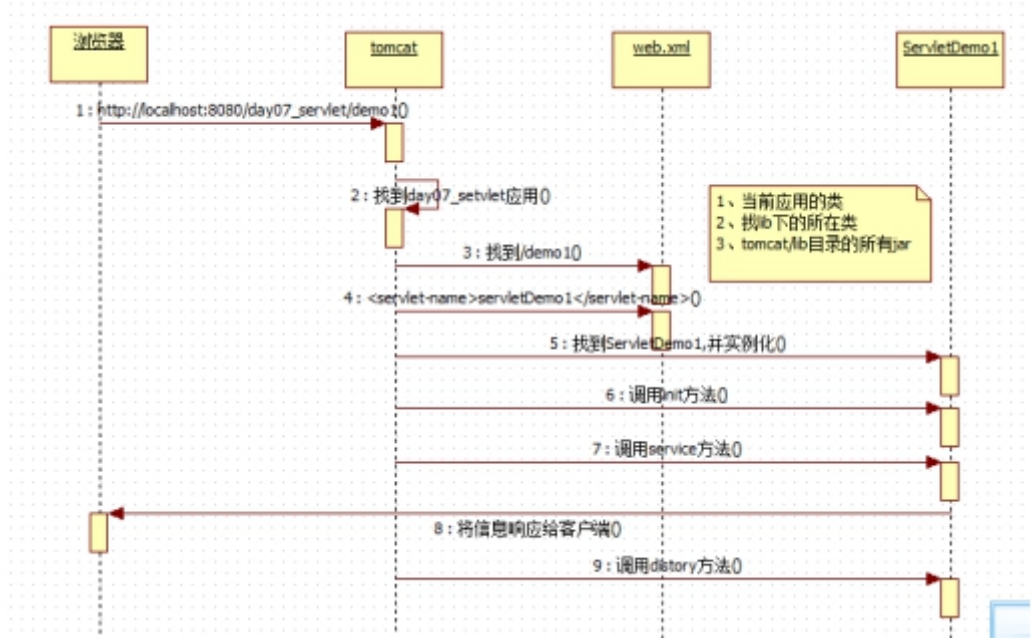
```
<!-- 创建一个servlet实例 -->
<servlet>
    <servlet-name>servletDemo1</servlet-name>
    <servlet-class>com.itcast.servlet.ServletDemo1</servlet-class>
</servlet>

<!-- 给servlet提供(映射)一个可访问的URI地址 -->
<servlet-mapping>
    <servlet-name>servletDemo1</servlet-name>
    <url-pattern>/demo1</url-pattern>
</servlet-mapping>
```



c、部署应用到tomcat服务器

D、测试：http://locahost:8080/day08_servlet/demo1

二、执行过程



三、Servlet生命周期（重要）

实例化-->初始化-->服务->销毁

出生：（实例化-->初始化）第一次访问Servlet就出生（默认情况下）

活着：（服务）应用活着，servlet就活着

死亡：（销毁）应用卸载了servlet就销毁。

```
//默认当第一次请求时，创建servet实例。应用存在实例就存在，当应用被卸载了，实例就销毁了。单实例
public class ServletDemo1 implements Servlet{
    //生命周期的方法：实例化对象
    //第一次被访问时调用
    public ServletDemo1(){
        System.out.println("********ServletDemo1被调用了********");
    }
    //生命周期的方法：初始化方法
    //第一次被访问时调用
    public void init(ServletConfig config) throws ServletException {
        System.out.println("********init被调用了********");
    }
```

```java
//生命周期的方法：服务方法
//接收用户请求，并做出响应
//每次请求都被调用
public void service(ServletRequest req, ServletResponse res)
        throws ServletException, IOException {
    System.out.println("********service被调用了********");
    //res.getWriter().write("hello ServletDemo1");
}
//生命周期的方法：销毁
//当应用卸载时调用
public void destroy() {
    System.out.println("********destroy被调用了********");
}
```

小知识：

如何让servlet在服务器启动时就创建。

```xml
<servlet>
  <servlet-name>servletDemo2</servlet-name>
  <servlet-class>cn.itcast.servlet.ServletDemo2</servlet-class>
  <load-on-startup>2</load-on-startup>
</servlet>
```

使当前Servlet在服务器启动时创建

四、Servlet的三种创建方式

4.1、实现javax.servlet.Servlet接口（参见：编写一个servlet程序：）

4.2、继承javax.servet.GenericServlet类(适配器模式)

```java
public class ServletDemo2 extends GenericServlet{

    public void service(ServletRequest req, ServletResponse res)
            throws ServletException, IOException {
        System.out.println("servletDemo2");
    }

}
```

4.3、继承javax.servlet.http.HttpServlet类（模板方法设计模式）

(开发中常用方式)

```java
//不要重写父类的service方法。
public class ServletDemo3 extends HttpServlet{
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        System.out.println("servletDemo3--get请求方法");
    }
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {

    }
```

Servlet --> GenericServlet --> HttpServlet (继承HttpServlet)

曾祖父 爷爷 爸爸

孙子

小技巧：使生成的servlet更清新一些

找到：MyEclipse\Common\plugins目录

把com.genuitec.eclipse.wizards_9.0.0.me201108091322.jar复制到上面目录

servet映射细节：

servet映射细节1：

```xml
  <!-- 配置多个映射路径 -->
<servlet-mapping>
  <servlet-name>ServletDmo4</servlet-name>
  <url-pattern>/ServletDmo4</url-pattern>
</servlet-mapping>

  <servlet-mapping>
  <servlet-name>ServletDmo4</servlet-name>
  <url-pattern>/ServletDmo44</url-pattern>
</servlet-mapping>
    <servlet-mapping>
  <servlet-name>ServletDmo4</servlet-name>
  <url-pattern>/admin/ServletDmo44</url-pattern>
</servlet-mapping>
```

servet映射细节2:通配符* 代表任意字符串

url-pattern: *.do 以*.字符串的请求都可以访问 注：不要加/

url-pattern: /* 任意字符串都可以访问

url-pattern： /action/* 以/action开头的请求都可以访问

匹配规则：

优先级：从高到低

绝对匹配--> /开头匹配 --> 扩展名方式匹配

如果url-pattern的值是/，表示执行默认映射。所有资源都是servlet

五、Servlet的线程安全

单实例：每次访问多线程

解决线程安全问题的最佳办法，不要写全局变量，而写局部变量。

六、Servlet获取配置信息

ServletConfig的使用

作用1：可以获取**servlet**配置信息

方式1：

```
private ServletConfig config;
//使用初始化方法回复到ServletConfig对象，此对象由服务器创建
public void init(ServletConfig config) throws ServletException {
    this.config = config;
}

public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    /*String encoding = "UTF-8";
    System.out.println(encoding);*/
    String value = config.getInitParameter("encoding");//根据配置文件中的名，得到值
    System.out.println(value);
```

方式2：

```
//通过使用继承父类的方法得到 ServletConfig对象
String value = this.getServletConfig().getInitParameter("encoding");
System.out.println(value);
```

方式3：

```
String value = this.getInitParameter("encoding");
System.out.println(value);
```

作用2：可以获得**ServletContext**对象

七、ServletContext（重要）

ServletContext：代表的是整个应用。一个应用只有一个ServletContext对象。单实例。

作用：

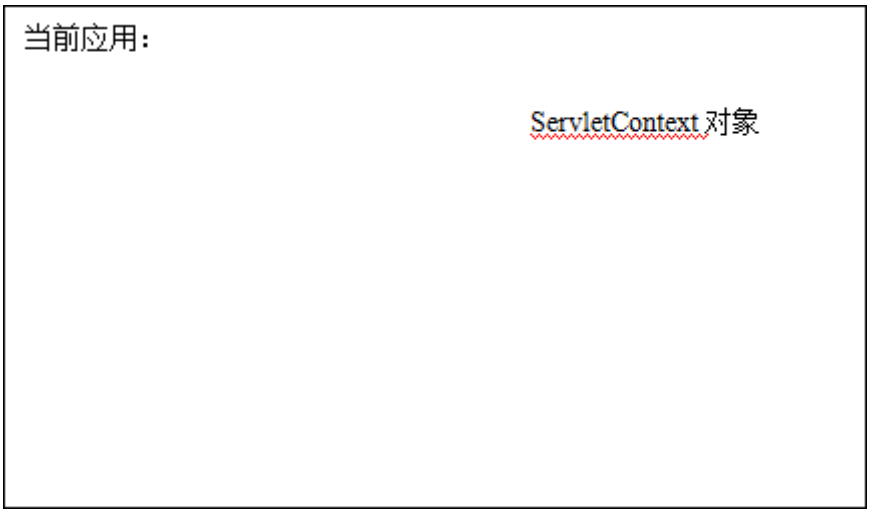
域对象：在一定范围内（当前应用），使多个Servlet共享数据。

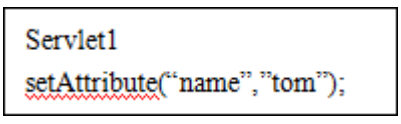
常用方法：

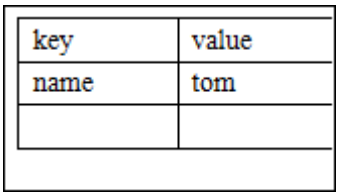
void setAttribute(String name,object value);//向ServletContext对象的map中添加数据

Object getAttribute(String name);//从ServletContext对象的map中取数据

void rmoveAttribute(String name);//根据name去移除数据







key	value
name	tom

Servlet2
getAttribute("name");

获取全局配置信息：

修改web.xml文件：

```
<!-- 配置当前应用的全局信息 -->
<context-param>
  <param-name>encoding</param-name>
  <param-value>UTF-8</param-value>
</context-param>
```

String	**getInitParameter**(String name) //根据配置文件中的key得到value

```
//获取全局配置信息
String encoding = sc.getInitParameter("encoding");
System.out.println(encoding);
```

获取资源路径：

String getRealPath(String path);//根据资源名称得到资源的绝对路径.

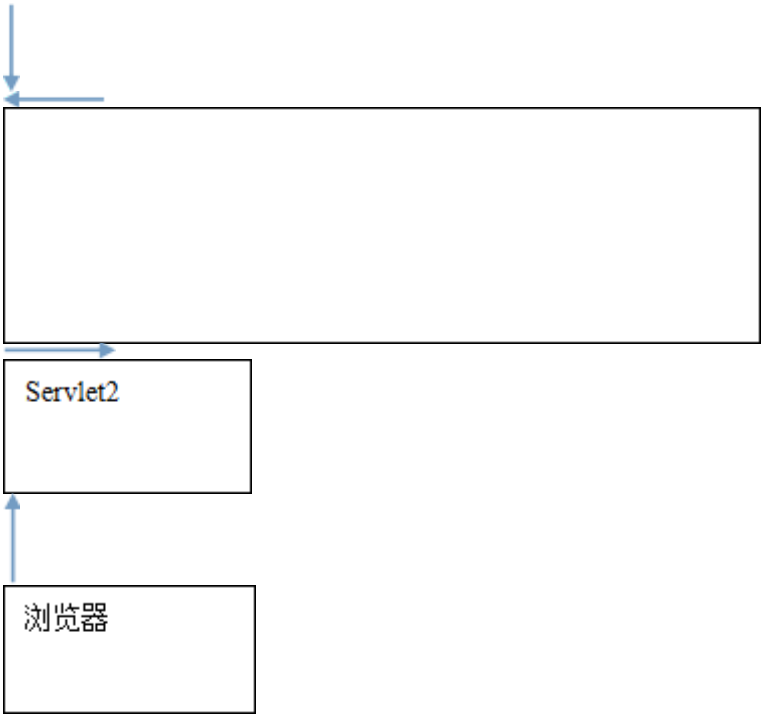
可以得到当前应用任何位置的任何资源。

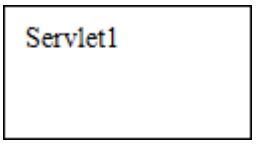
```java
private void test3() throws IOException, FileNotFoundException {
    ServletContext sc = this.getServletContext();
    //得到a.properties
    String path = sc.getRealPath("/WEB-INF/classes/com/itcast/servletContext/c.properties");//要以/开头，
    Properties prop = new Properties();
    prop.load(new FileInputStream(path));
    System.out.println(prop.getProperty("key"));
}
//得到b.properties
private void test2() throws IOException, FileNotFoundException {
    ServletContext sc = this.getServletContext();
    //得到a.properties
    String path = sc.getRealPath("/WEB-INF/classes/b.properties");//要以/开头，/代表的是当前应用的名称
    Properties prop = new Properties();
    prop.load(new FileInputStream(path));
    System.out.println(prop.getProperty("key"));
}
//获取a.properties文件的数据
private void test1() throws IOException, FileNotFoundException {
    ServletContext sc = this.getServletContext();
    //得到a.properties
    String path = sc.getRealPath("/WEB-INF/a.properties");//要以/开头，/代表的是当前应用的名称
    Properties prop = new Properties();
```

实现Servlet的转发。

```java
//实现请求转发,目标是跳转到ServletContextDemo1
ServletContext sc = this.getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher("/ServletContextDemo1");
rd.forward(request, response);//将请求信息向下传递
```

RequestDispatcher getRequestDispatcher(String path) ;//参数表示要跳转到哪去




Servlet1