**Servlet**

Servlet：运行在WEB服务器中的小型java程序，通过HTTP协议接受和响应WEB客户端的请求

文件：

java文件：继承servlet接口

public class ServletDemo1 implements Servlet{

@Override

public void destroy() {

// TODO Auto-generated method stub

}

@Override

public ServletConfig getServletConfig() {

// TODO Auto-generated method stub

return null;

}

@Override

public String getServletInfo() {

// TODO Auto-generated method stub

return null;

}

@Override

public void init(ServletConfig arg0) throws ServletException {

// TODO Auto-generated method stub

}

@Override

public void service(ServletRequest arg0, ServletResponse arg1) throws ServletException, IOException {

// TODO Auto-generated method stub

System.out.println("hello,my servlet!");

}

}

XML文件：配置信息

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" id="WebApp\_ID" version="3.0">

<display-name>MyServlet</display-name>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<!-- 创建一个Servlet实例 -->

<servlet>

<servlet-name>servletDemo1</servlet-name>

<servlet-class>servlet.ServletDemo1</servlet-class>

</servlet>

<!-- 给servlet提供（映射）一个可供客户端访问的URL -->

<servlet-mapping>

<servlet-name>servletDemo1</servlet-name>

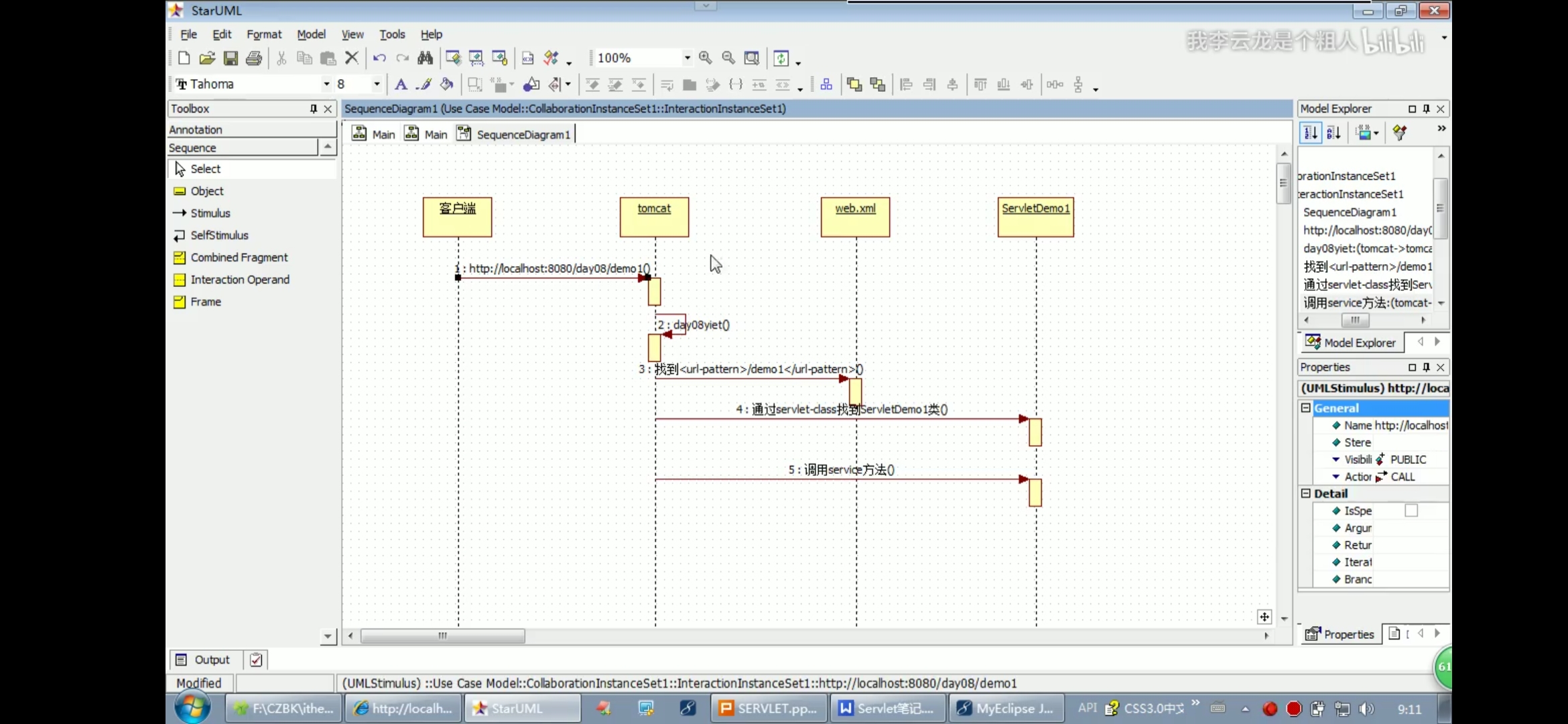
<!-- 相当于http://localhost:8080/MyServlet/demo1 -->

<url-pattern>/demo1</url-pattern>

</servlet-mapping>

</web-app>

# 执行过程



### serlet映射细节

* 同一个类可以对应多个映射地址

<servlet-mapping>

<servlet-name>servletDemo1</servlet-name>

<url-pattern>/demo1</url-pattern>

<url-pattern>/demo1.jsp</url-pattern>

<url-pattern>/1</url-pattern>

</servlet-mapping>

* 使用通配符

<servlet-mapping>

<servlet-name>servletDemo1</servlet-name>

<url-pattern>/demo1</url-pattern>

<url-pattern>/\*.demo</url-pattern>

<url-pattern>/demo/\*</url-pattern>

<url-pattern>/\*</url-pattern>

</servlet-mapping>

优先级：绝对匹配>/开头匹配>扩展名方式匹配

# 生命周期

实例化🡪初始化🡪服务🡪销毁

对应方法：构造器🡪init()🡪service()🡪destroy()

让servlet在服务器启动时就创建的方法：在web.xml中配置

<servlet>

<servlet-name>servletDemo1</servlet-name>

<servlet-class>servlet.ServletDemo1</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>

# 三种创建方式

### 实现javax.servlet.servlet接口

### 继承javax.servlet.GenericServlet类（适配器模式）

与方式一不同的是，方式二根据需要来重写方法，不用全部实现

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

（开发常用方式）

特点：根据需求重写doGet()和doPost()方法，不能重写service方法

# 线程安全

单实例：每次访问多线程

解决办法：不要写全局变量，写局部变量

# 获取配置信息

## ServletConfig

注意：使用方法二、三时不能重写init()方法，否则无法获取内置对象

public class ServletConfigDemo1 extends HttpServlet{

private static final long serialVersionUID = 1L;

// private ServletConfig config;

// @Override

// public void init(ServletConfig config) throws ServletException {

// // TODO Auto-generated method stub

// this.config = config;

// }

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

//方式一

// String encoding = config.getInitParameter("encoding");

//方式二

// String encoding = super.getInitParameter("encoding");

//方式三

String encoding = this.getServletConfig().getInitParameter("encoding");

System.out.println(encoding);

}

}

<servlet>

<servlet-name>ServletConfigDemo1</servlet-name>

<servlet-class>servlet.ServletConfigDemo1</servlet-class>

<init-param>

<param-name>encoding</param-name>

<param-value>GBK</param-value>

</init-param>

</servlet>

## ServletContext

ServletContext代表整个应用。一个应用只有一个ServletContext对象，单实例。

作用：

* 作为域对象，使多个Servlet共享数据
* 获取全局配置信息，包括多个servlet的配置信息
* 获取资源路径：getRealPath()

<context-param>

<param-name>sex</param-name>

<param-value>man</param-value>

</context-param>

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

ServletContext application = this.getServletContext();

if(application != null) {

//添加键值对

application.setAttribute("name", "Tom");

//获取属性

System.out.println(application.getAttribute("name"));//从域对象获取

System.out.println(application.getInitParameter("sex"));//从配置文件获取全局配置参数

}else {

System.out.println("application为空");

}

}

ServletContext实现请求转发

//实现请求转发

ServletContext application = this.getServletContext();

RequestDispatcher rd = application.getRequestDispatcher("/ServletConfigDemo1");

rd.forward(req, resp);

# 总结

servlet相关对象

