# 注解式开发

**旧的web.xml文件的配置方式的缺点**：

1. 每个servlet都配置一套servlet标签，会导致web.xml文件过大；
2. 每创建一个Servlet都要配置一次web.xml会导致开发效率降低；
3. 在web.xml中使用到的配置是很少用到的

**能不能将这些配置信息写到Java类中呢**？

可以。在Servlet3.0版本之后，推出了各种基于Servlet的注解式开发。

好处：开发效率高；不需要编写大量的web.xml配置信息；可以在Java类上使用注解标注；web.xml文件的体积变小了。

*注意：不是说有了注解就不需要web.xml文件了，有一些需要变化的信息还是要配置到web.xml文件中，一般都是注解+配置文件的开发新式*。

一些不会经常变化的信息建议使用注解，可能会被修改的内容推荐写到web.xml文件中。

## 使用方式

在类上面标记·@WebServlet·表示这个类是一个Servlet。（这个注解只能应用在一个类上面）

## 设置注解的属性

name：用于指定Servlet的名字，等同于<Servlet-name>

urlPatterns、value：用于指定隐射路径。一个字符串数组，也就是说可以用多个url-pattern代表这个servlet。

这两个属性实际上的功能一样，如果属性名是value的话，可以省略属性名QAQ。

loadOnStartup：用于指定服务器启动阶段是否加载该Servlet。等同于web.xml中的<load-on-startup>。

initParams：初始化参数

对于属性是一个数组的属性，如果只需要一个属性值，可以不写花括号。

下面是一个带有注解的Servlet的展示：

package com.sleepy\_tian.servlet;  
  
import jakarta.servlet.ServletException;  
import jakarta.servlet.annotation.WebInitParam;  
import jakarta.servlet.annotation.WebServlet; // 包含了注解的类。  
import jakarta.servlet.http.HttpServlet;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.util.Enumeration;  
  
// 添加注解：  
// 用注解标注这个类，说明这是一个Servlet。  
@WebServlet( // 下面的配置需要哪些写哪些就可以，不是每一个都必须写。  
 name = "helloServlet", // servlet名字。  
 urlPatterns = {"/servlet/hello","/servlet/world"}, // url模式  
 loadOnStartup = 0, // 启动时加载。  
 initParams = { @WebInitParam(name = "paramName1", value = "paramVal1"), @WebInitParam(name = "paramName2", value = "paramVal2")} // 初始化参数。  
)  
  
  
public class HelloServlet extends HttpServlet {  
  
 public HelloServlet () {  
 System.out.println("Hello Servlet Constructor.");  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 resp.setContentType("text/html;UTF-8");  
 PrintWriter writer = resp.getWriter();  
 // 输出servlet名字。  
 String servletName = getServletName();  
 writer.print("Servlet name = " + servletName + "<br>");  
 // 输出servlet path  
 String servletPath = req.getServletPath();  
 writer.print("servlet path = " + servletPath + "<br>");  
 // 获取初始化参数。  
 Enumeration<String> initParameterNames = this.getInitParameterNames();  
 while (initParameterNames.hasMoreElements()) {  
 String parameterName = initParameterNames.nextElement();  
 String initParameter = this.getInitParameter(parameterName);  
 writer.print("init parameter: " + initParameter + "<br>");  
 }  
  
 }  
}