实验 11　Spring MVC 配置请求映射方式

一、　Spring MVC 配置请求映射方式

需求 1：配置基于 ant 通配符风格的 URL 路径映射。

?通配符：匹配任何单字符;

***通配符：**匹配 0 或者任意数量的字符;

****通配符：**匹配 0 或者多级目录

开发需求: 需要匹配如下五种情形的 ant 路径映射

通配符	URL路径
?	/ant1?
*	/ant2/*.do
*	/*/ant3
**	/**/ant4
**	/ant5/**

需求 2：配置基于 RESTful 风格的 URL 路径映射。

开发需求: 需要开发包括如下四种 HTTP 请求的 RESTful 服务

1. GET 用于获取资源

2. POST 用于新建资源

3. PUT 用于更新资源

4. DELETE 用于删除资源

二、 测试工具

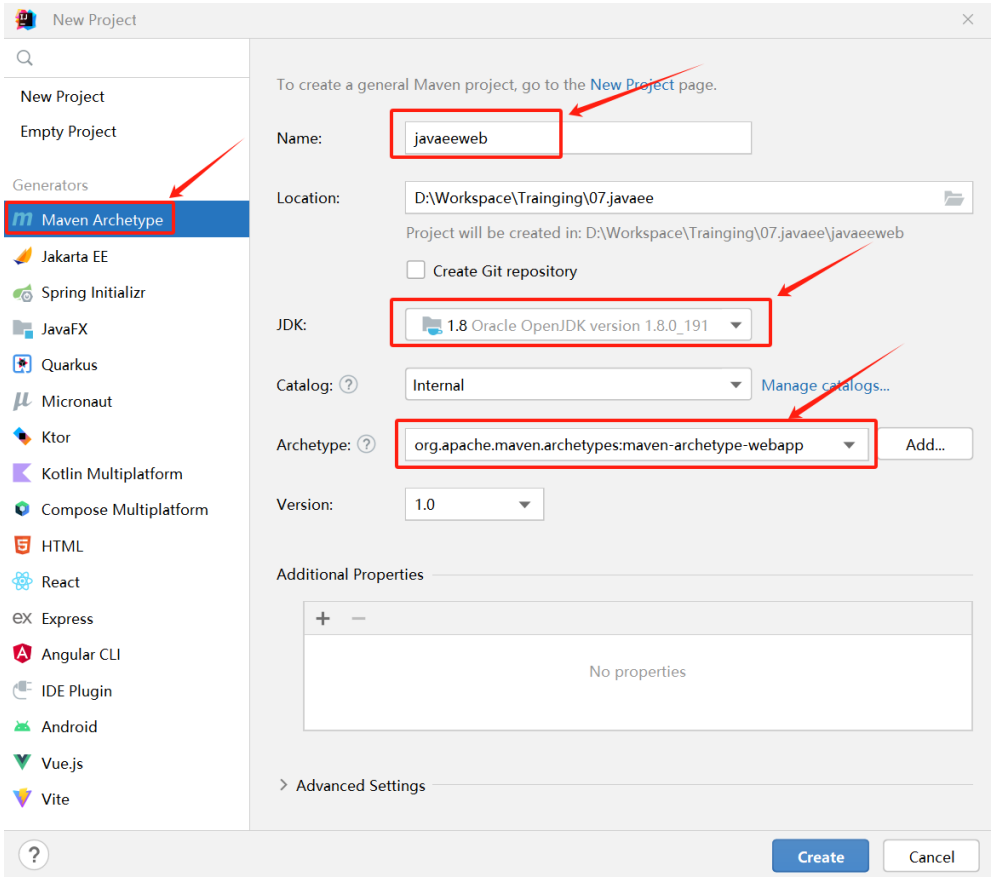
1、Postman11.2.0.0

三、 开发准备

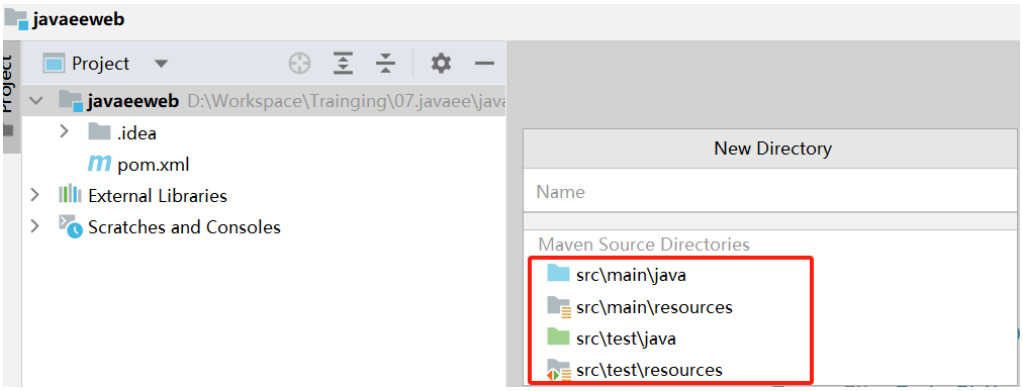
1、创建 Maven Web 项目

方式一：直接创建 trainingweb 项目（具体过程省略，只列出关键步骤）

IDEA 2022 创建新项目时（**其他版本根据实际**），选择创建 **Maven Archetype** 项目类型即可，如下



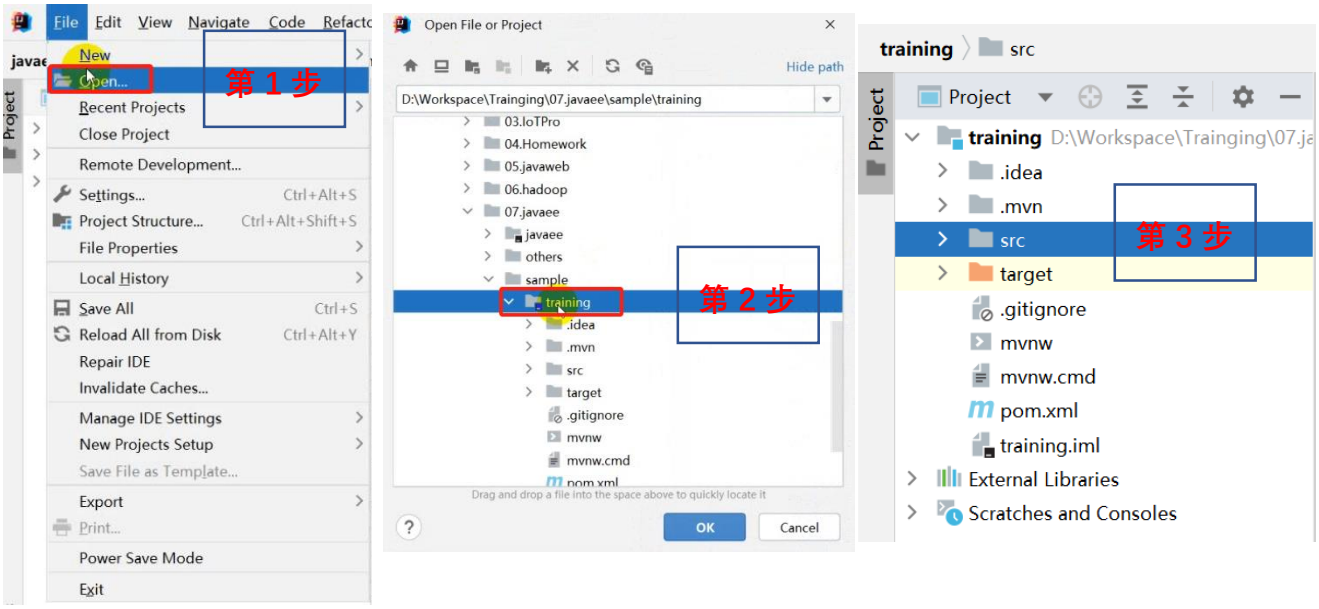
注意，上一步创建后，**需要等待项目的自动构建。**最终会自动创建 webapp 目录以及相应的 web 相关的配置文件。**其他目录需要手工创建目录，如下：**



方式二：导入 Maven Web 项目（推荐）

Step1，下载并解压 trainingweb 工程。从【学习通】→【章节】→本章上机实验中找到 **trainingweb.zip** 压缩包，下载到本地，并解压成 trainingweb 目录。

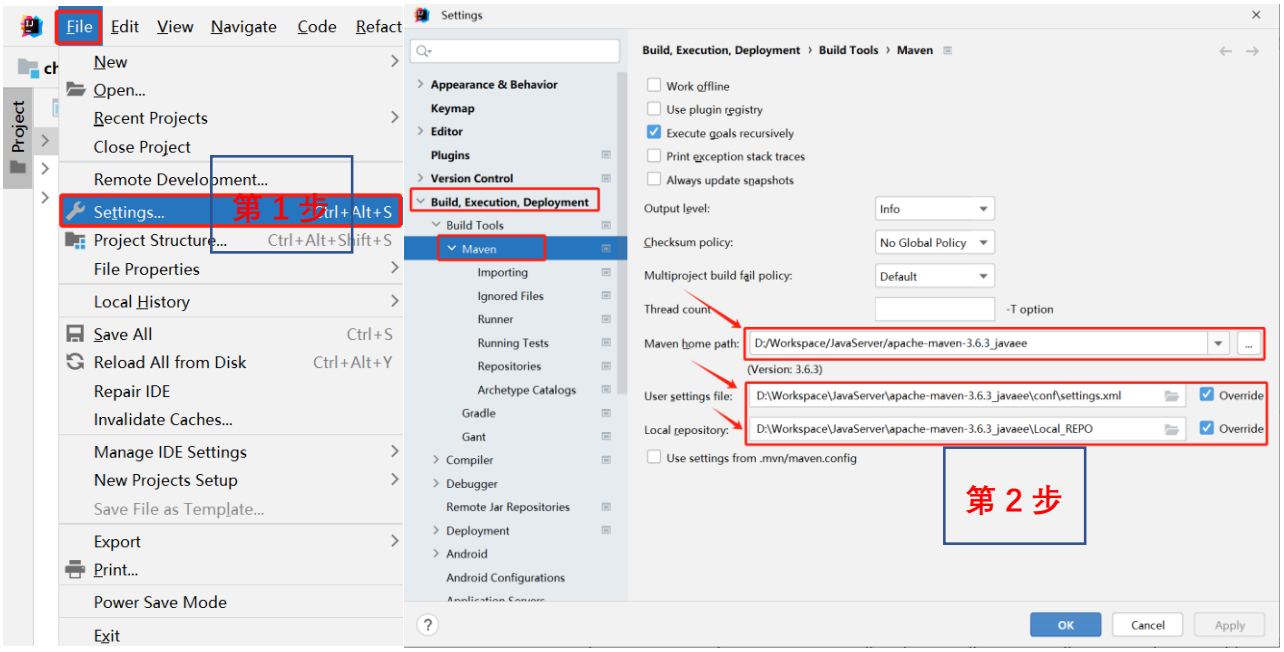
Step2，导入 trainingweb 工程。打开 IDEA 工具，点击【File】→【Open…】→选中上一步解压的 trainingweb 工程，如下图所示。。。确定即可导入。



注意，导入工程后，对于 **IDEA 是 2023** 的版本，要点击左上角边的 文件夹小图标，即可查看到工程中的文件。

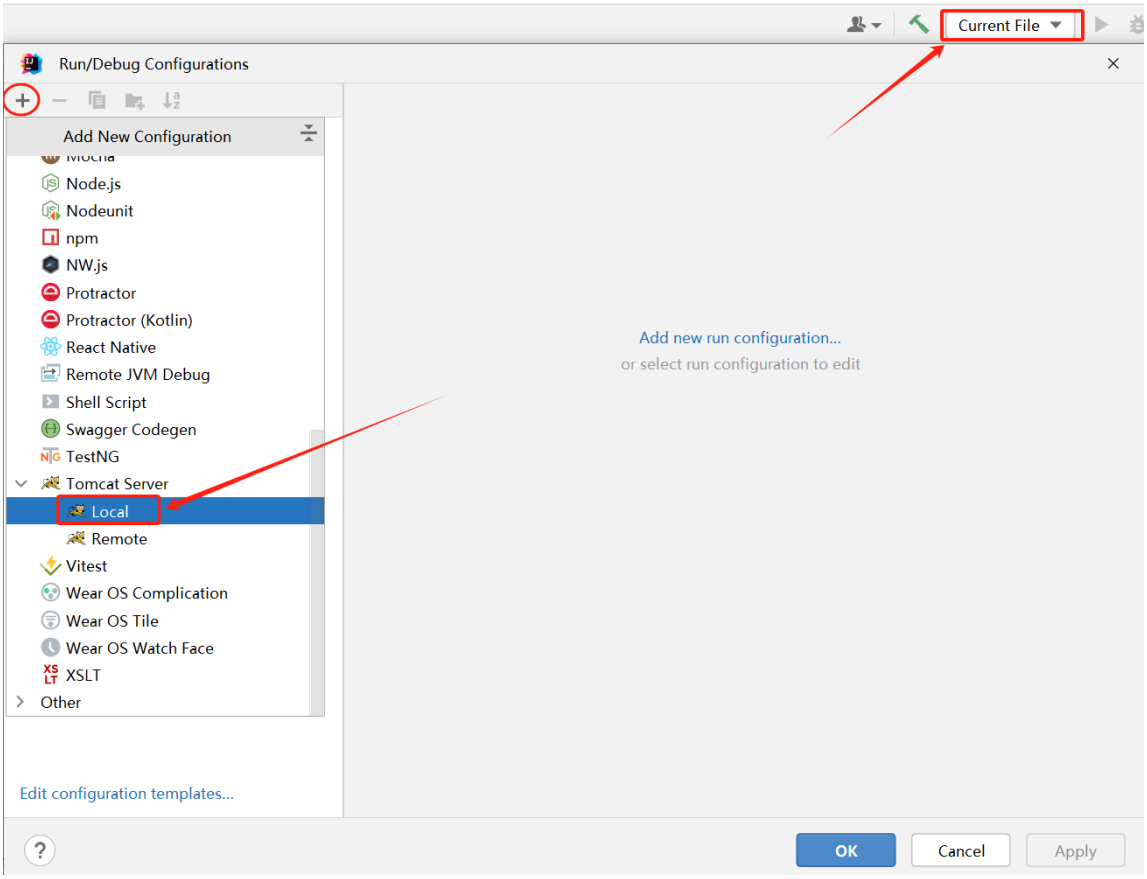
2、配置 Maven 构建工具

选择【File】→【Setting…】然后找到菜单位【Build, Execution, Deployment】→【Build Tools】→【Maven】，按如下图所示，将你的**本地 Maven 路径和配置文件**设置好即可。

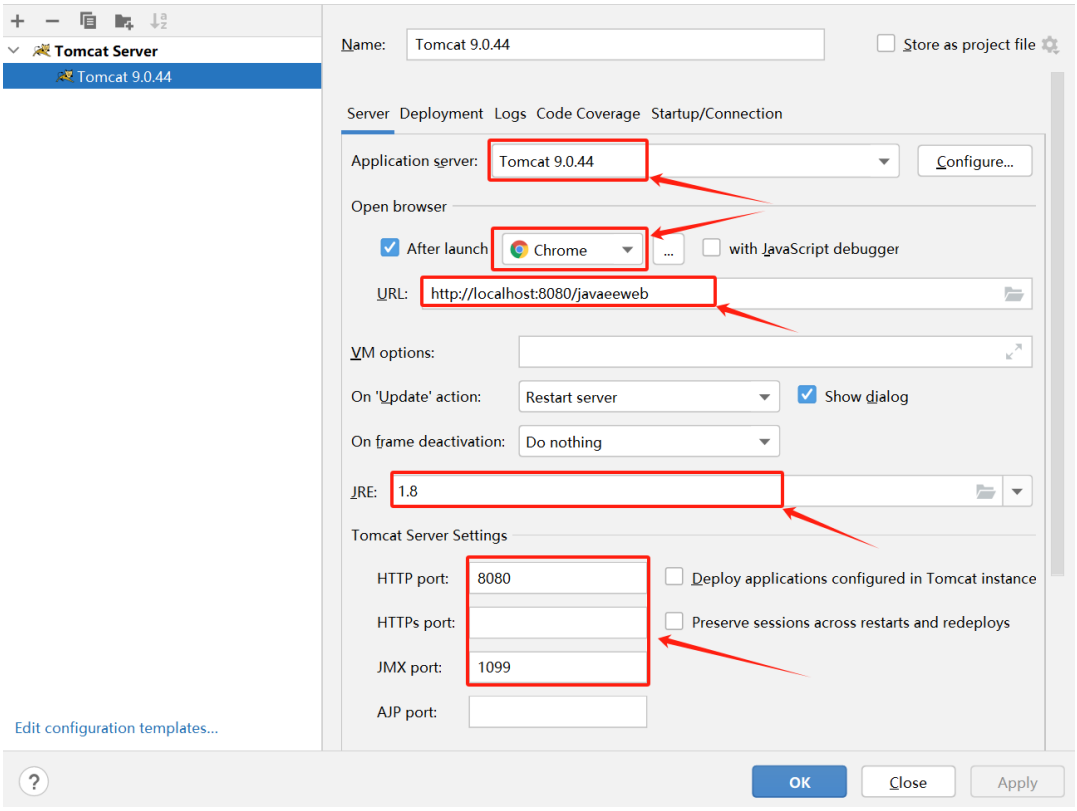


3、配置 Tomcat 服务器

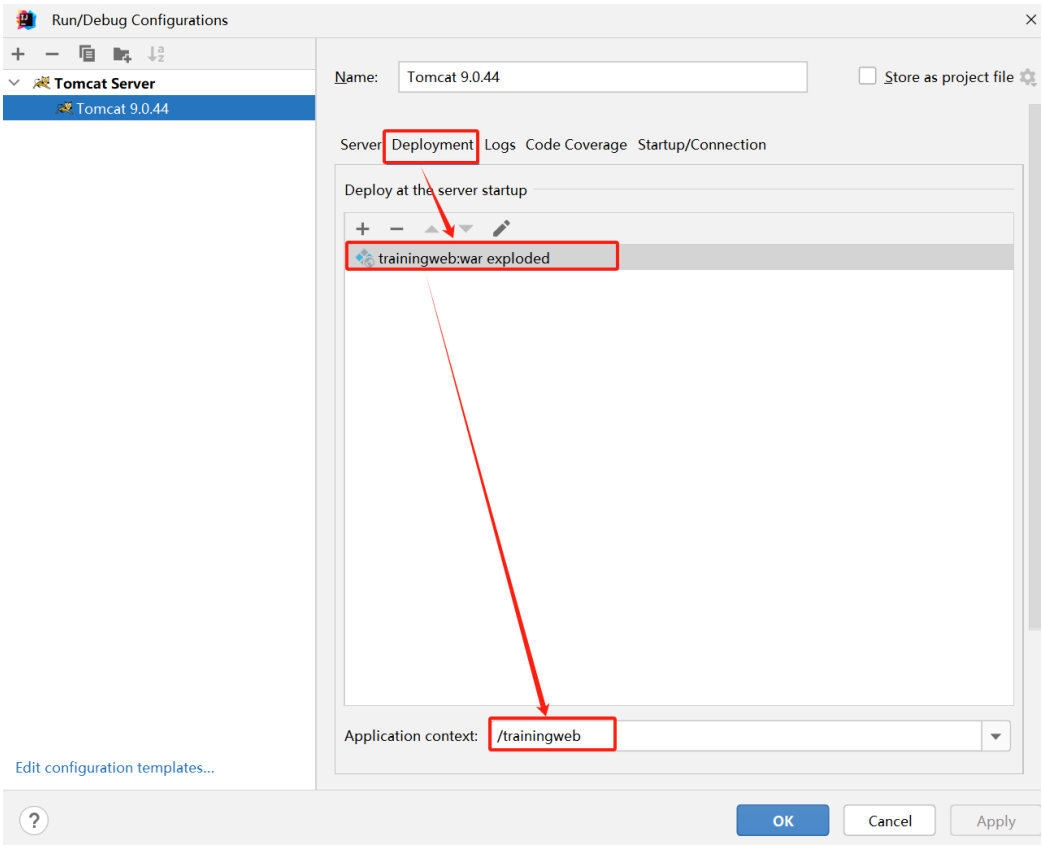
选择【**Edit Configuration**】→点击【**+**】按钮，拉到 Tomcat Server，选择→【**Local**】，如下图所示。



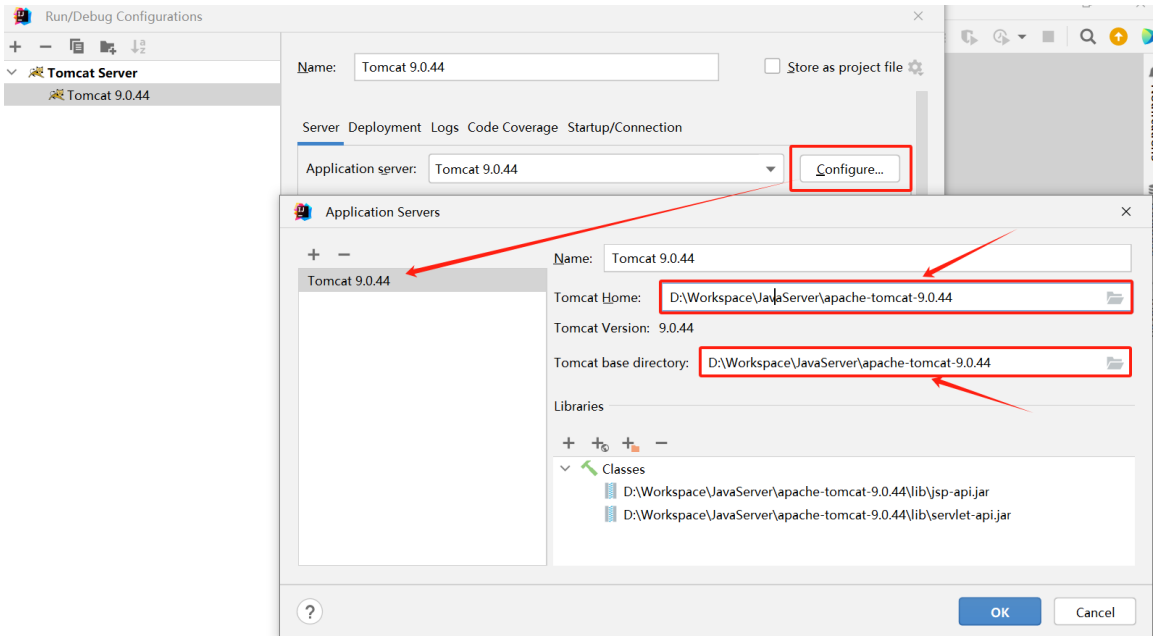
Step1：在 Server 页签配置 Tomcat 服务器，配置如下



Step2：切换到 Deployment 页签，设置部署包为 trainingweb:war exploded，配置如下



Step3：其中 Tomcat 服务器需要指向真实的 Tomcat 目录（指向你本地的 Tomcat），配置如下。



四、 程序开发

第 1 步：配置 pom.xml

保持与第 10 章实验配置一致。

第 2 步：配置前端控制器 web.xml

保持与第 10 章实验配置一致。

第 3 步：配置处理器映射信息和视图解析器 Spring-mvc.xml 文件

在 src/main/resources 目录下创建 spring-mvc.xml 文件，用于配置处理器映射信息和视图解析器。在实验 10 的配置基础上，加上本章的配置。具体配置如下所示。

spring-mvc.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- 配置 Spring MVC 要扫描的包 -->
    <context:component-scan base-package="com.xhxg.controller.chapter10"/>
    <context:component-scan base-package="com.xhxg.controller.chapter11"/>

    <!-- 配置视图解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/pages/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

</beans>
```

第 4 步：、创建处理器 Controller 类（基于 Ant 通配符风格的 URL 路径映射）

在项目 src/main/java 目录下创建代码包：**com.xhxg.controller.chapter11**，创建 **AntPathController** 类，使用**@Controller 注解**注册此类为处理器类。另外，还需要使用**@RequestMapping("/api/ant")注解**声明访问空间，代码如下所示：

AntPathController.java

```java
package com.xhxg.controller.chapter11;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

no usages
@Controller
@RequestMapping("/api/ant")
public class AntPathController {

    no usages
    @RequestMapping("/ant1?")
    public String ant1() {
        System.out.println("ant1-->? match");
        return "success";
    }

    no usages
    @RequestMapping("/ant2/*.do")
    public String ant2() {
        System.out.println("ant2-->/ant2/*.do match");
        return "success";
    }

    no usages
    @RequestMapping("/*/ant3")
    public String ant3() {
        System.out.println("ant3-->/*/ant3 match");
        return "success";
    }

    @RequestMapping("/**/ant4")
    public String ant4() {
        System.out.println("ant4-->/**/ant4 match");
        return "success";
    }

    no usages
    @RequestMapping("/ant5/*")
    public String ant5() {
        System.out.println("ant5-->/ant5/* match");
        return "success";
    }
}
```

以上配置 5 种适配 Ant 通配符风格的请求方法映射，这些方法都配置了访问成功后的跳转的页面，即 success。结合 spring-mvc.xml 配置文件中配置的视图解析器设定的前缀和后缀拼接，即可确定视图的最终返回路径。

第 5 步：、创建处理器 Controller 类（基于 RESTful 风格的 URL 路径映射）

在项目 src/main/java 目录下创建代码包：**com.xhxg.controller.chapter11**，创建 **RestfulController** 类，使用**@Controller 注解**注册此类为处理器类。另外，还需要使用**@RequestMapping("/api/rest")注解**声明访问空间，代码如下所示：

RestfulController.java

```java
package com.xhxg.controller.chapter11;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

no usages
@Controller
@RequestMapping("/api/rest")
public class RestfulController {
    //@PathVariable注解来获取URL中的参数
    no usages
    @GetMapping("/{id}")
    public String findById(@PathVariable("id") int id) {
        System.out.println("find by ID, ID=" + id);
        return "success";
    }

    no usages
    @DeleteMapping("/{id}")
    public String deleteById(@PathVariable("id") int id) {
        System.out.println("delete by ID, ID=" + id);
        return "success";
    }

    no usages
    @PutMapping("/{id}")
    public String updateById(@PathVariable("id") int id) {
        System.out.println("update by ID, ID=" + id);
        return "success";
    }

    @PostMapping("/{id}")
    public String add(@PathVariable("id") int id) {
        System.out.println("add by ID, ID=" + id);
        return "success";
    }
}
```

以上配置 Get、Post、Put、Delete 4 种 Restful 风格的请求方法映射，这些方法都配置了访问成功后的跳转的页面，即 success。结合 spring-mvc.xml 配置文件中配置的视图解析器设定的前缀和后缀拼接，即可确定视图的最终返回路径。

第 6 步：创建视图（View）页面

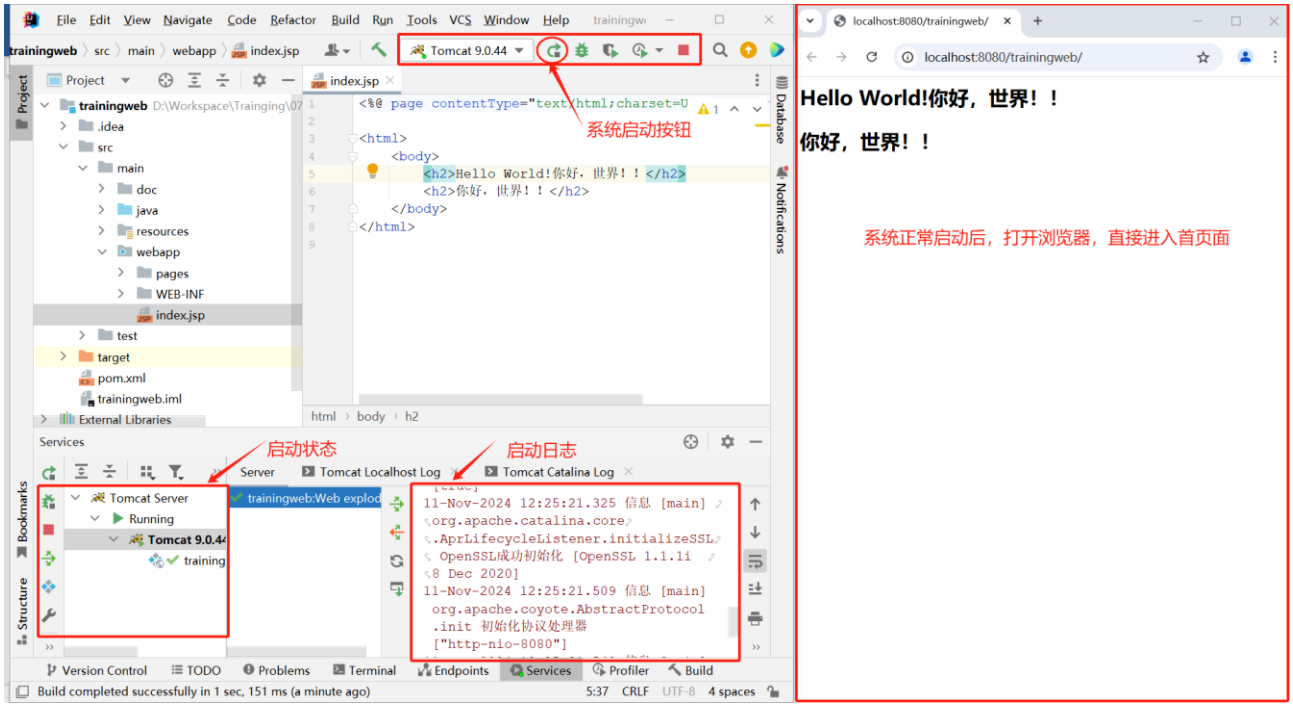
保持与第 10 章实验配置一致。

第 7 步：创建系统首页面

保持与第 10 章实验配置一致。

五、 系统运行

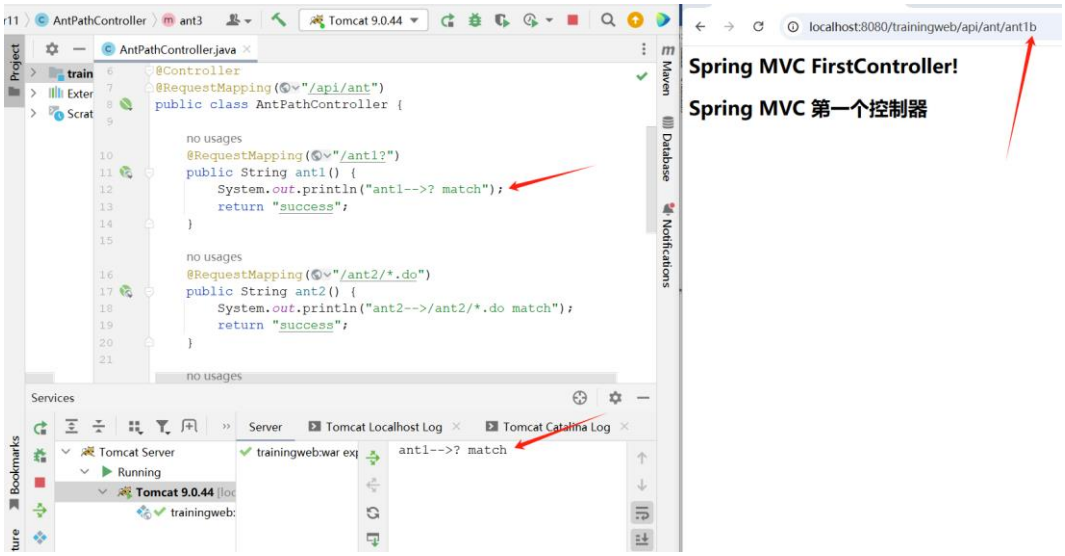
【第三章节】第 3 小节中已经配置好了 Tomcat 服务器（如没有配置好，请继续按第 3 小节的步骤配置好 Tomcat 服务器），所以我们直接启动此服务器即完成系统运行。



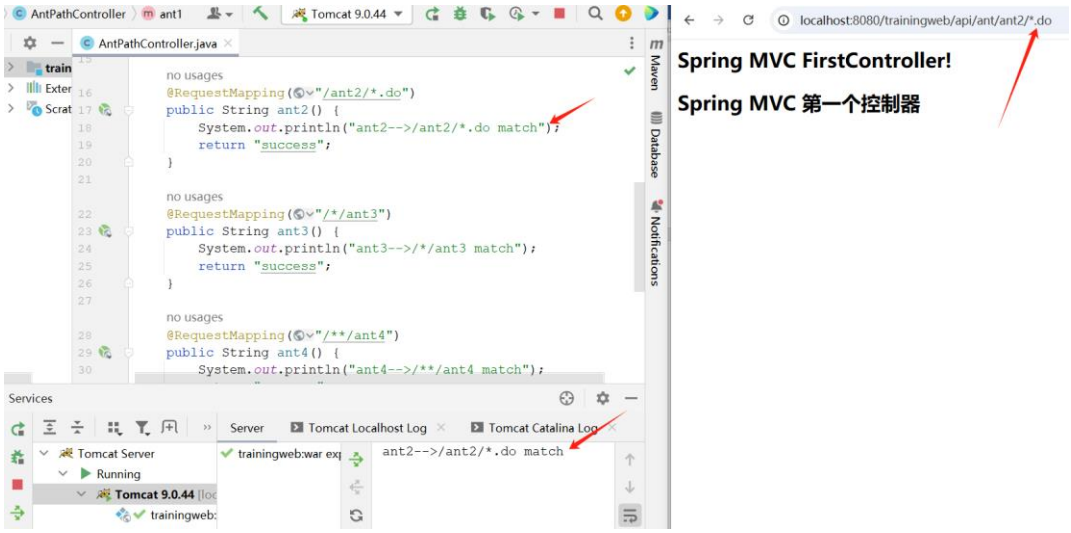
六、 系统测试

1、基于 Ant 通配符风格的 URL 路径映射测试

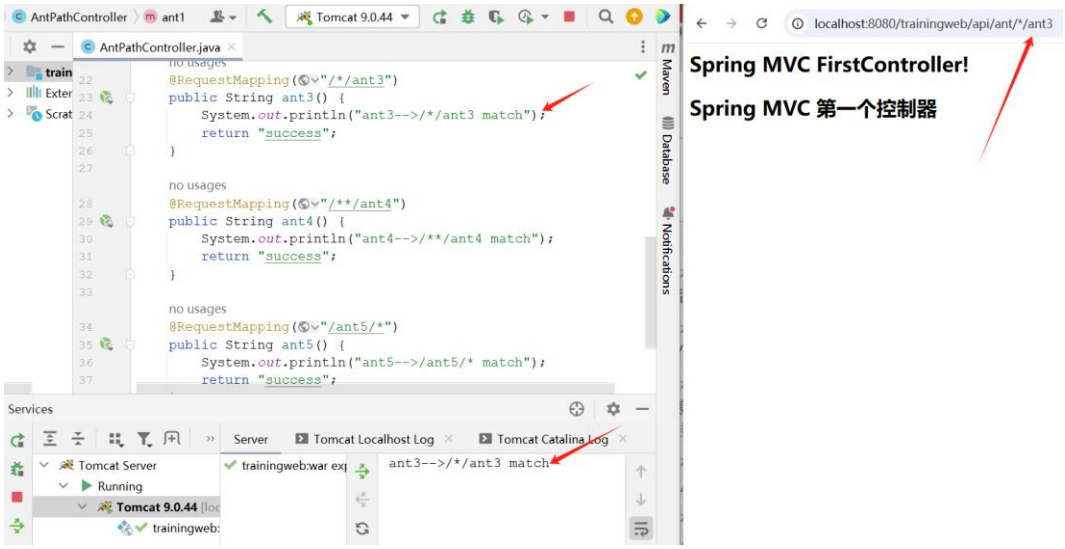
a) ?通配符：/ant1?



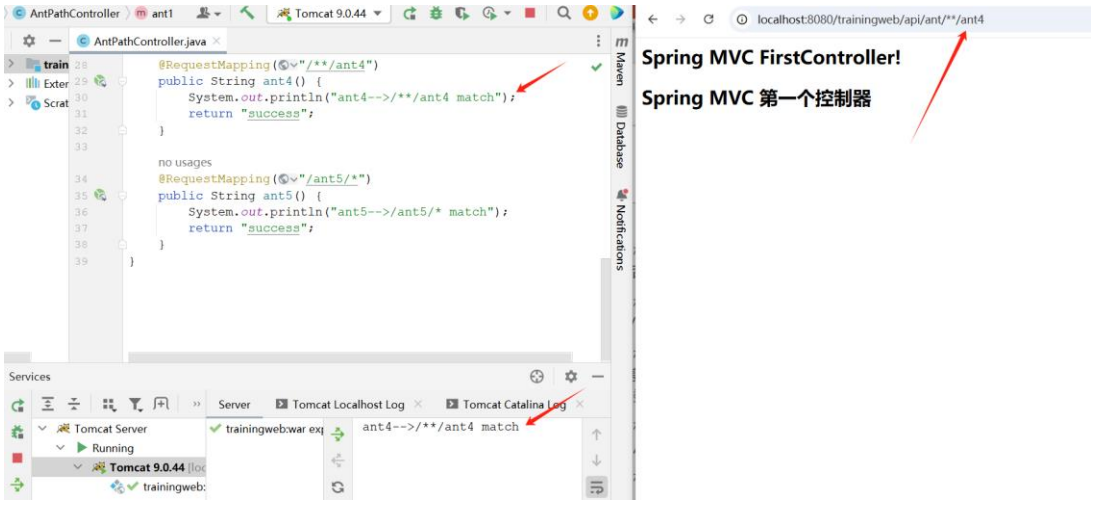
b) *通配符：/ant2/*.do



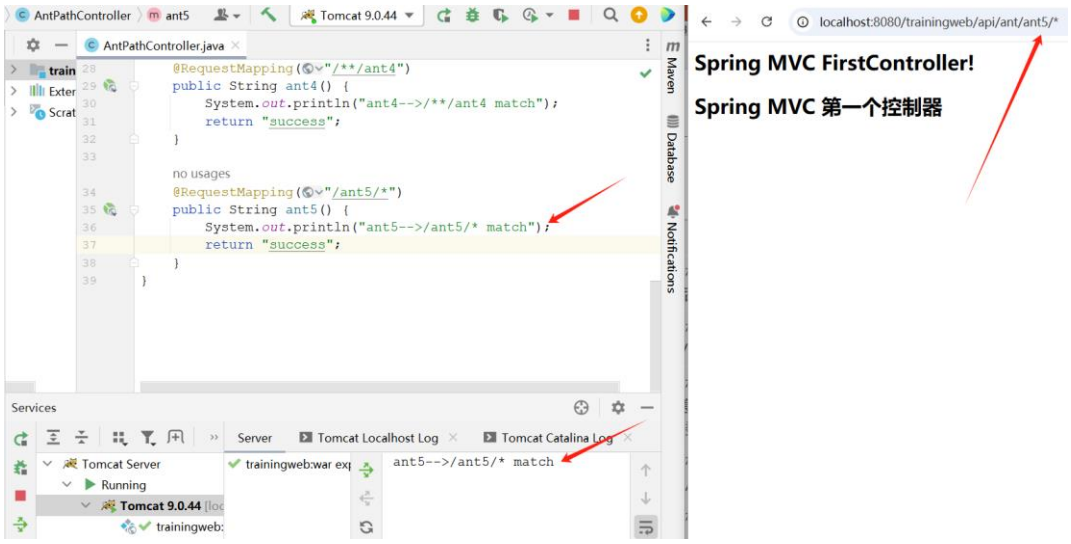
c) *通配符：/*/ant3



d) **通配符：/**/ant4

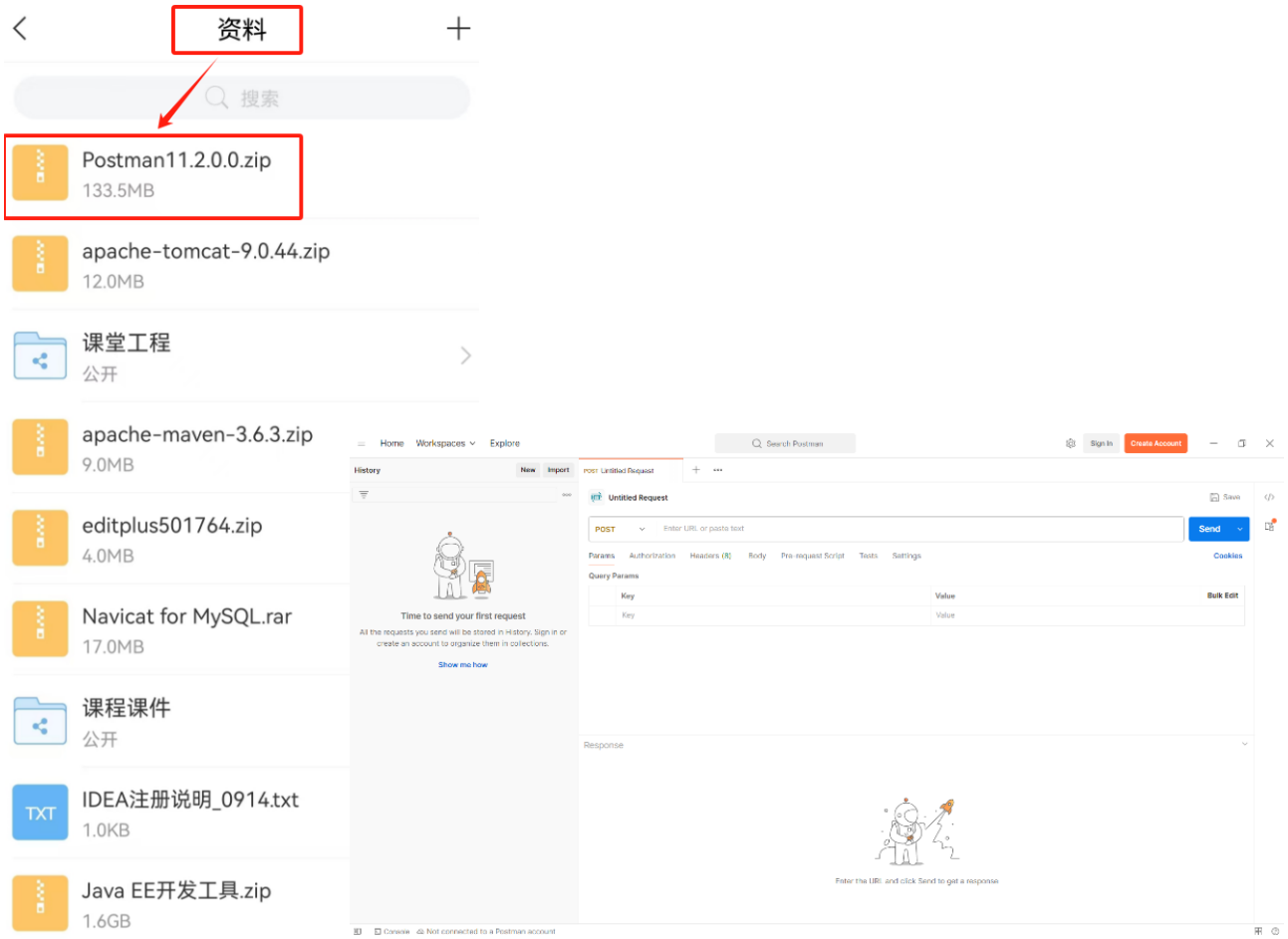


e) **通配符：/ant5/**

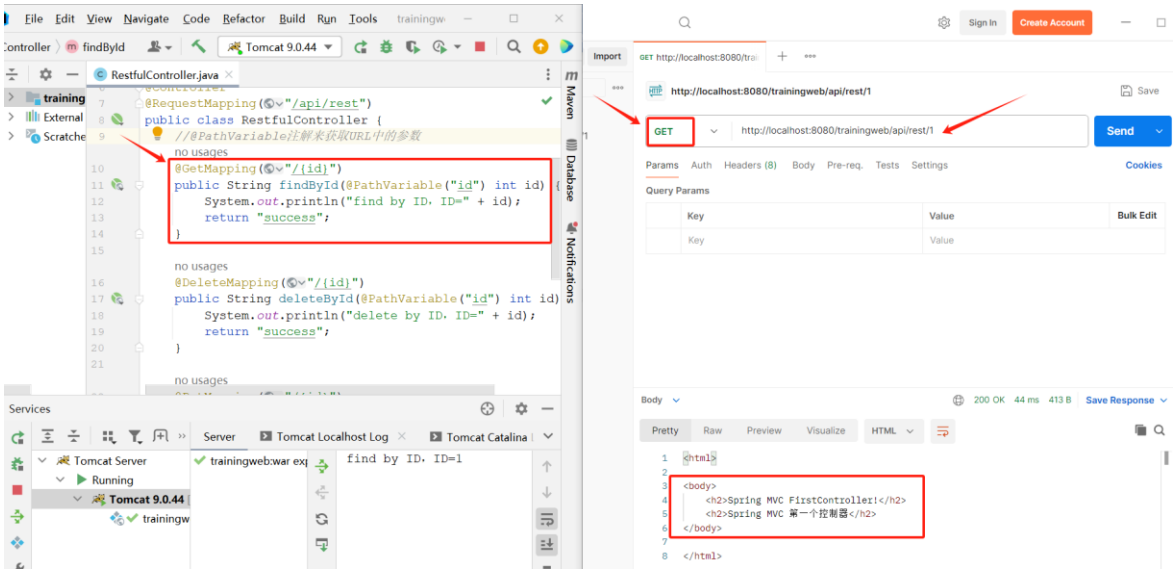


2、基于 RESTFul 风格的 URL 路径映射测试

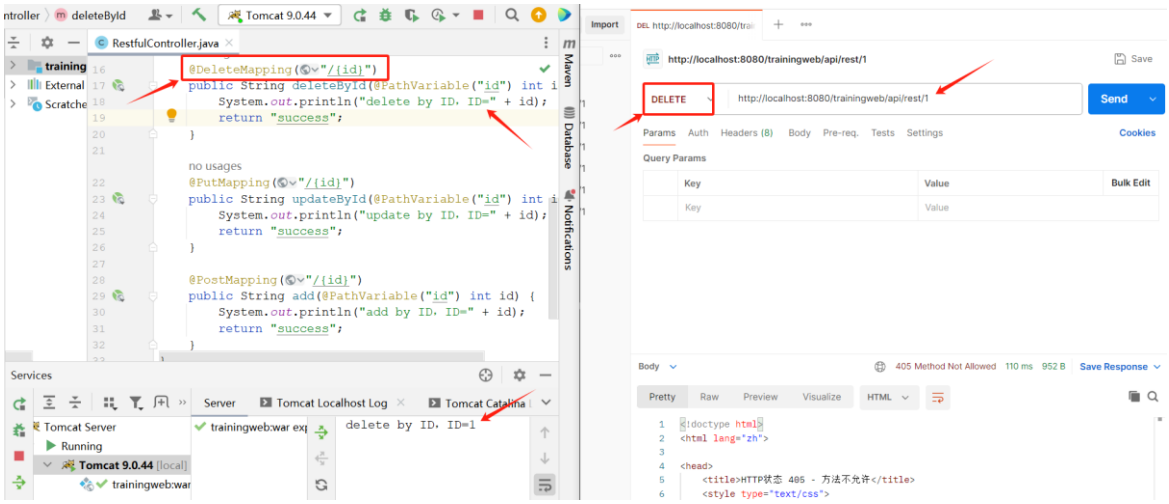
需要先下载 Postman 测试工具，路径：【学习通】→【资料】→【Postman 11.2.0.0.zip】，如下图所示，下载到你本地电脑，然后解压后直接双击启动。下载和启动截图如下：



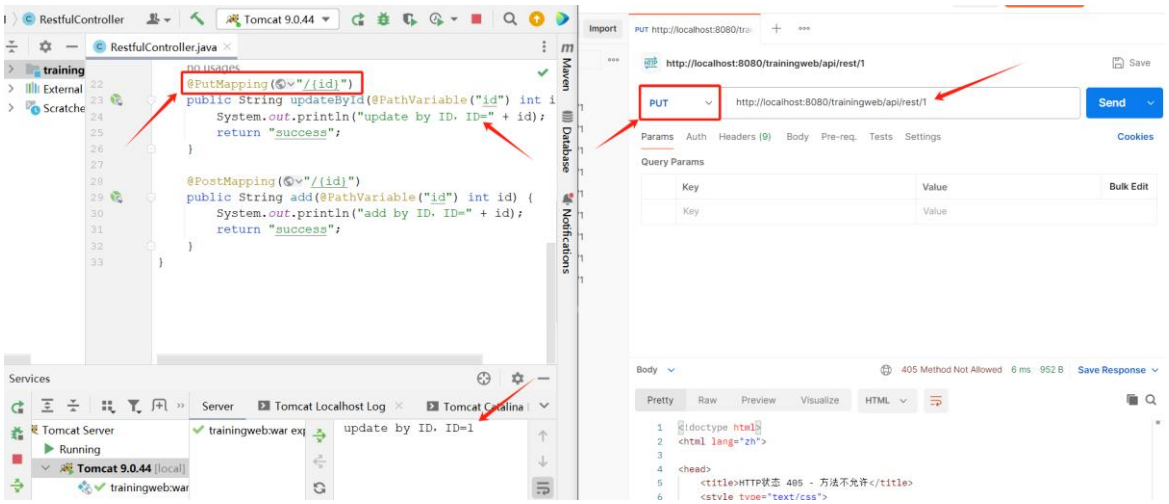
a)Get 请求



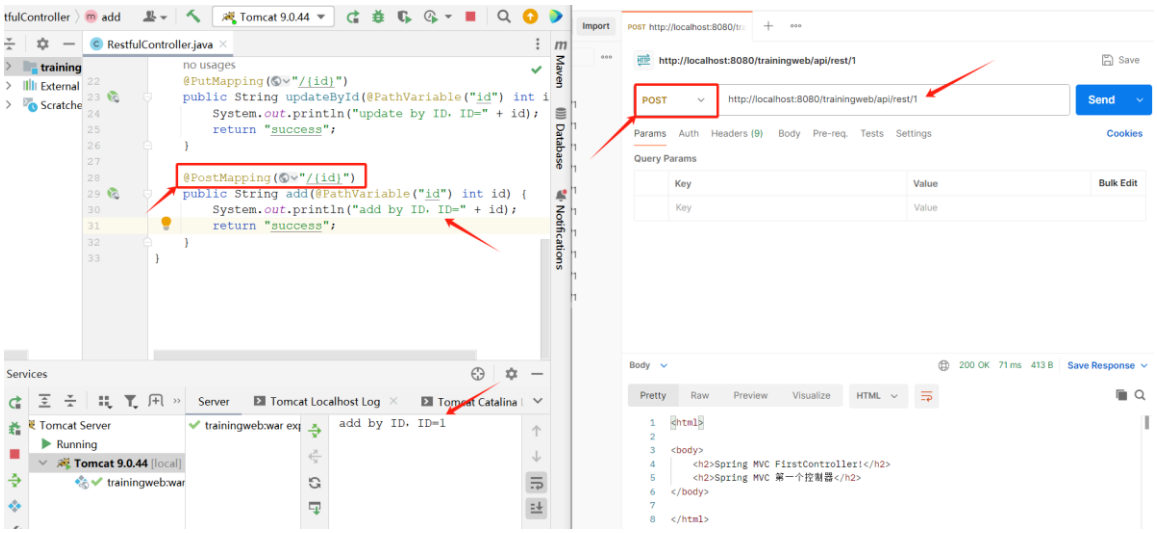
b)Delete 请求



c)Put 请求



d)Post 请求



七、 常见问题

待定。。。