Java EE架构

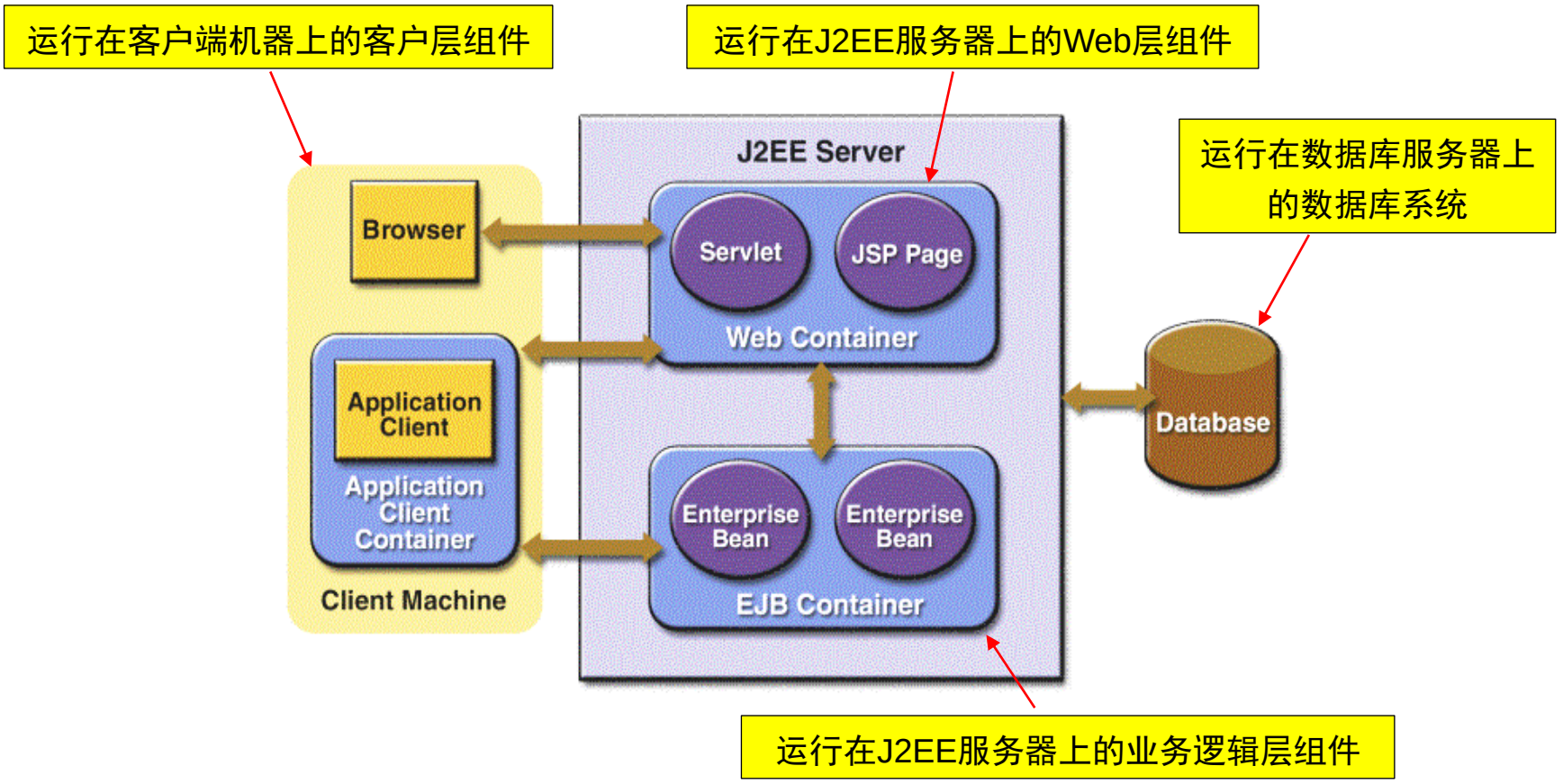
主讲教师：张智
计算机学院软件工程系
课程群：598986302

第1章 概述

1.1 Java EE简介

- Java EE－Java Enterprise Edition
  - Java EE-企业版(针对企业级应用) → 或称 J2EE、Java Web
  - Java SE-标准版针(针对普通PC应用)
  - Java ME-微型版(针对嵌入式设备及消费类电器)
- Java EE已成为企业级开发的首选平台之一。
- Java EE是一系列的技术规范，而并非是一个产品，更不是编程语言。

Java EE基本架构（B/S模式）



Java Web开发涉及的技术

Java EE由一整套服务（Services）、应用程序接口（APIs）和协议构成，它对开发基于Web的多层应用提供了功能支持

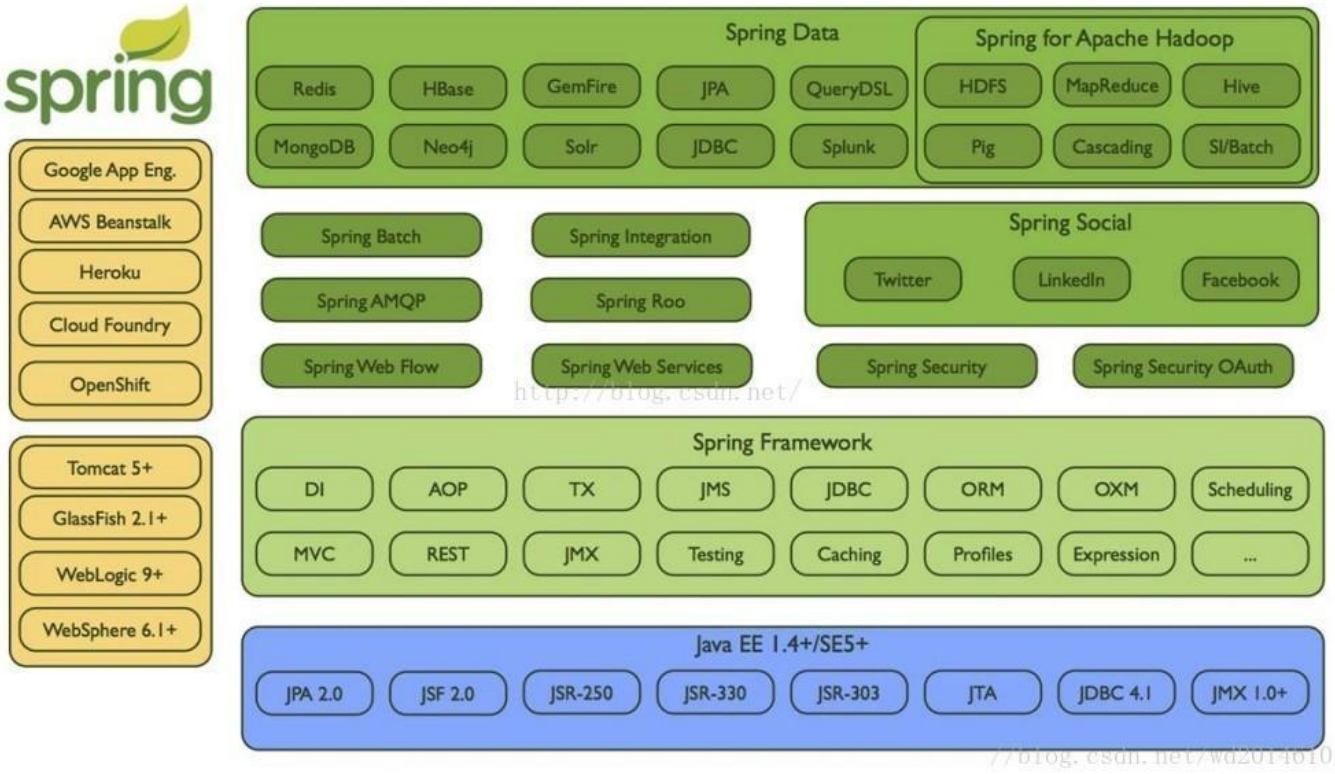
- Java Web开发涉及到众多开发框架和工具：
  - 开发语言：Java，Kotlin等
  - 构建工具：Gradle，Maven等
  - Web容器：JSP，Servlet，Tomcat等
  - 前端模版引擎：Thymeleaf（完全替代 JSP ）
  - 数据库技术：JDBC，Mysql等
  - ORM工具：JPA，Hibernate，Mybatis等
  - 开发框架：Struts，Spring，Spring MVC等
  - 缓存技术：Redis等
  - 消息组件：RabbitMQ等
  - 安全框架：Security、Shiro等

  …

SpringBoot整合一切

航空母舰：Spring Boot

Spring Boot 是由 Pivotal 团队提供的全新框架：
简化创建产品级的 Spring 应用和服务，简化配置文件，使用嵌入式web服务器，含有诸多开箱即用微服务功能…

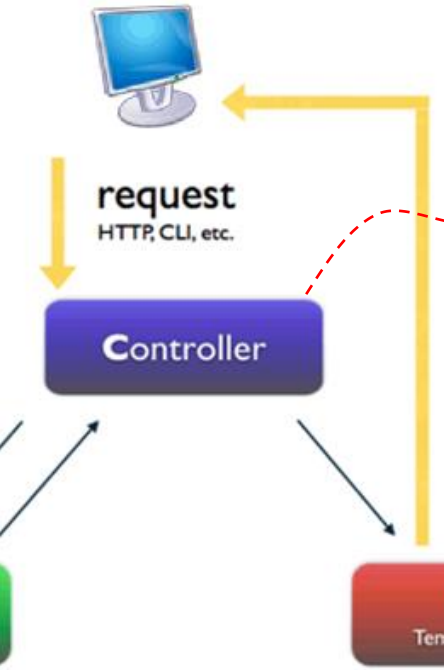


【返回】

1.2 MVC 开发模式

- Web 开发中最常用的分层开发模式：MVC （Model-View-Controller）

分层开发好处：将应用程序的用户界面和业务逻辑分离，使得代码具备良好的可扩展性、可复用性和可维护性。

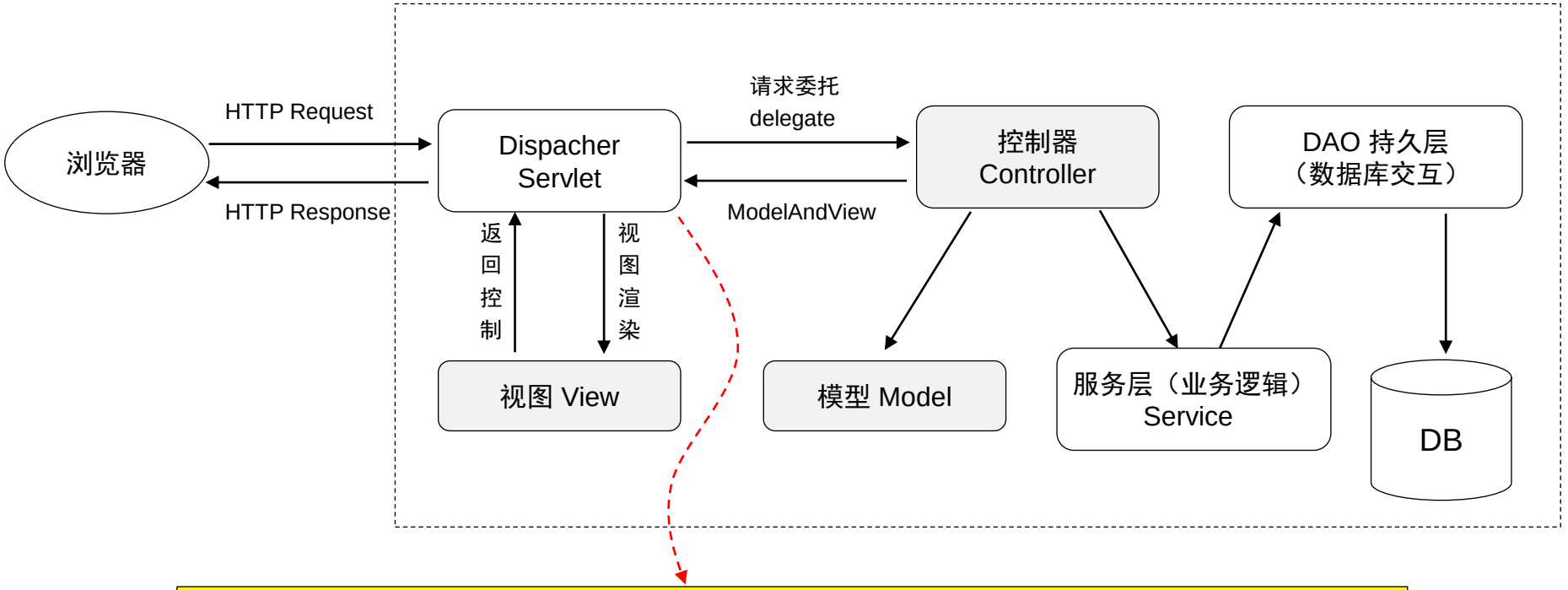


控制器C：处理浏览器的请求，决定如何调用业务层的数据增、删、改、查等业务操作，以及如何将结果返回给视图进行渲染。

模型M：应用的实体类，用于在内存中暂时存储数据，并在数据变化时通知控制器。

视图V：主要用来解析、处理、显示内容，并进行模板的渲染。

Spring MVC 模式



在整个 Spring MVC 框架中，Dispacher Servlet 处于核心位置，继承自 HttpServlet，负责协调和组织不同组件，完成控制器的路径映射匹配等处理，并返回响应工作。

【返回】

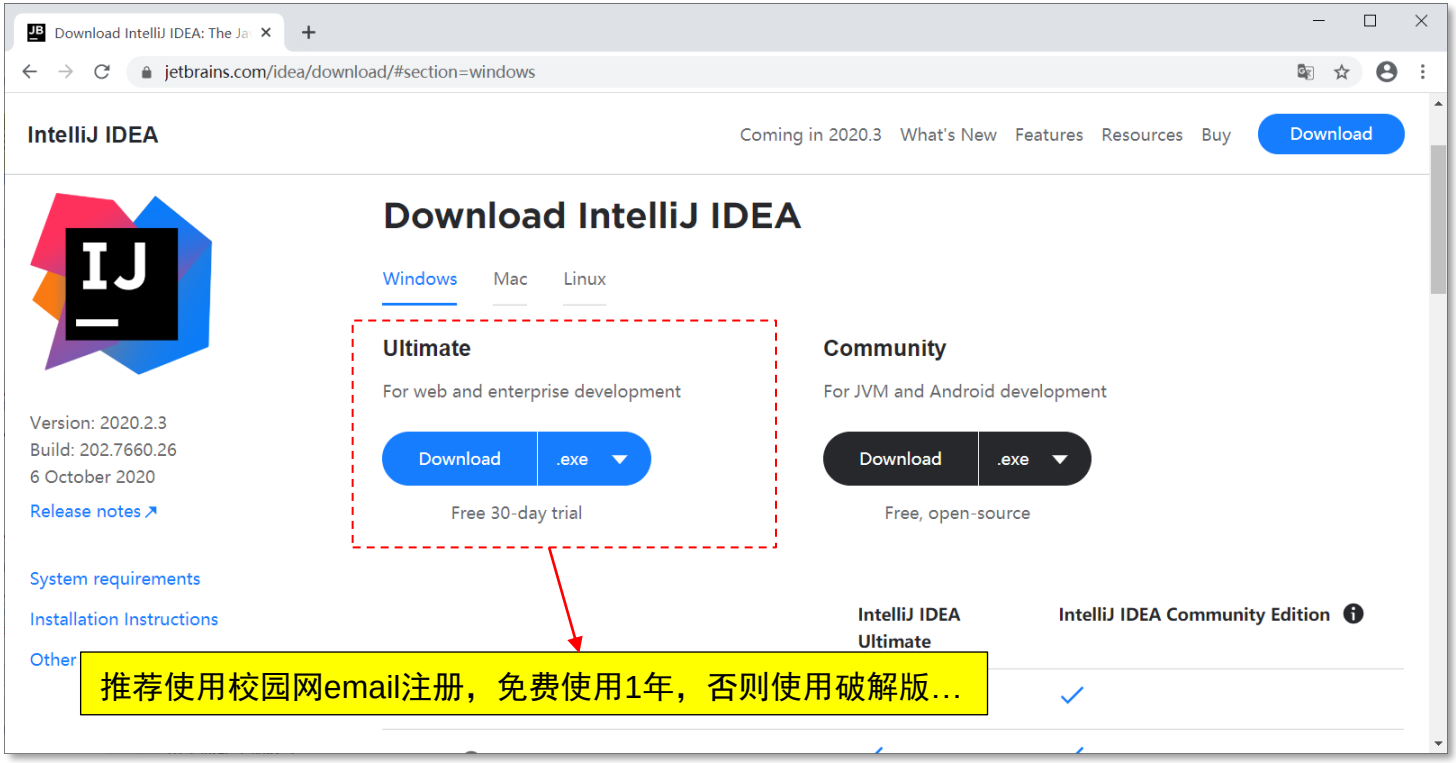
1.3 开发环境搭建

- 安装Idea Ultimate版
- 安装JDK
- 配置Maven ★★★
- 安装Postman
- 安装Mysql数据库

【返回】

1. 安装Idea Ultimate版

- 官网：https://www.jetbrains.com/idea/download/#section=windows



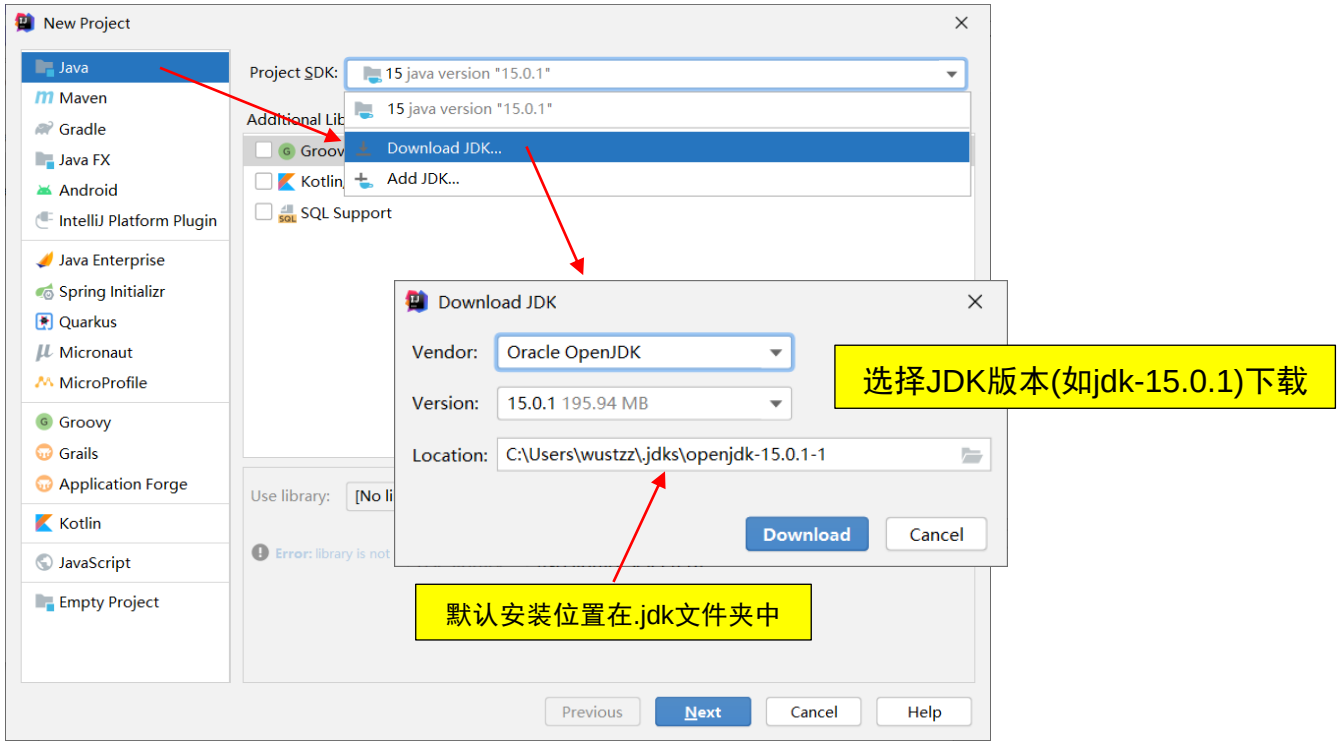
Idea Ultimate安装参考：

- 免费申请：
  - https://www.cnblogs.com/xicyannn/p/10505846.html
  - https://blog.csdn.net/kanchaishaonian/article/details/81214904
- 破解版：
  - https://www.cnblogs.com/kkakura/p/13686904.html

【返回】

2. 安装安装JDK

- 启动Idea → 新建项目 → 选择Java → 在Project SDK中选择下载JDK

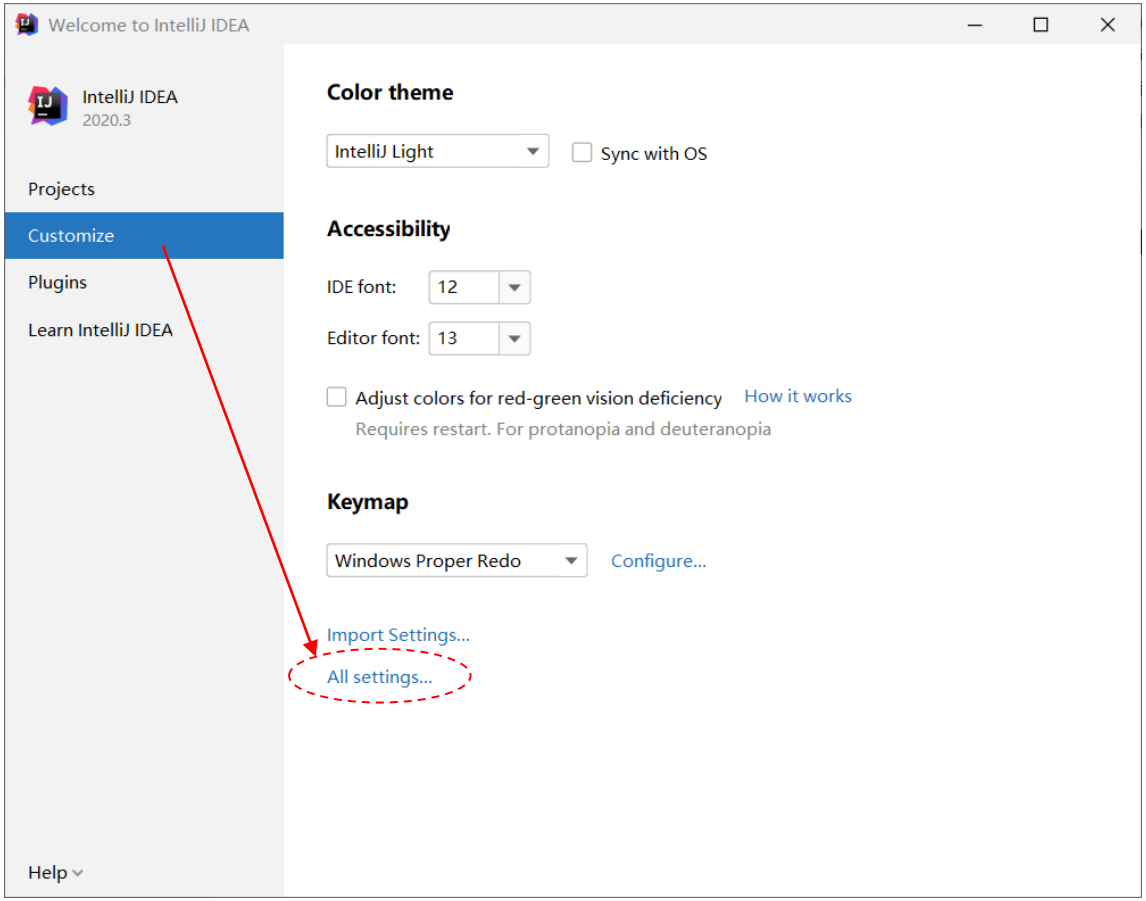


【返回】

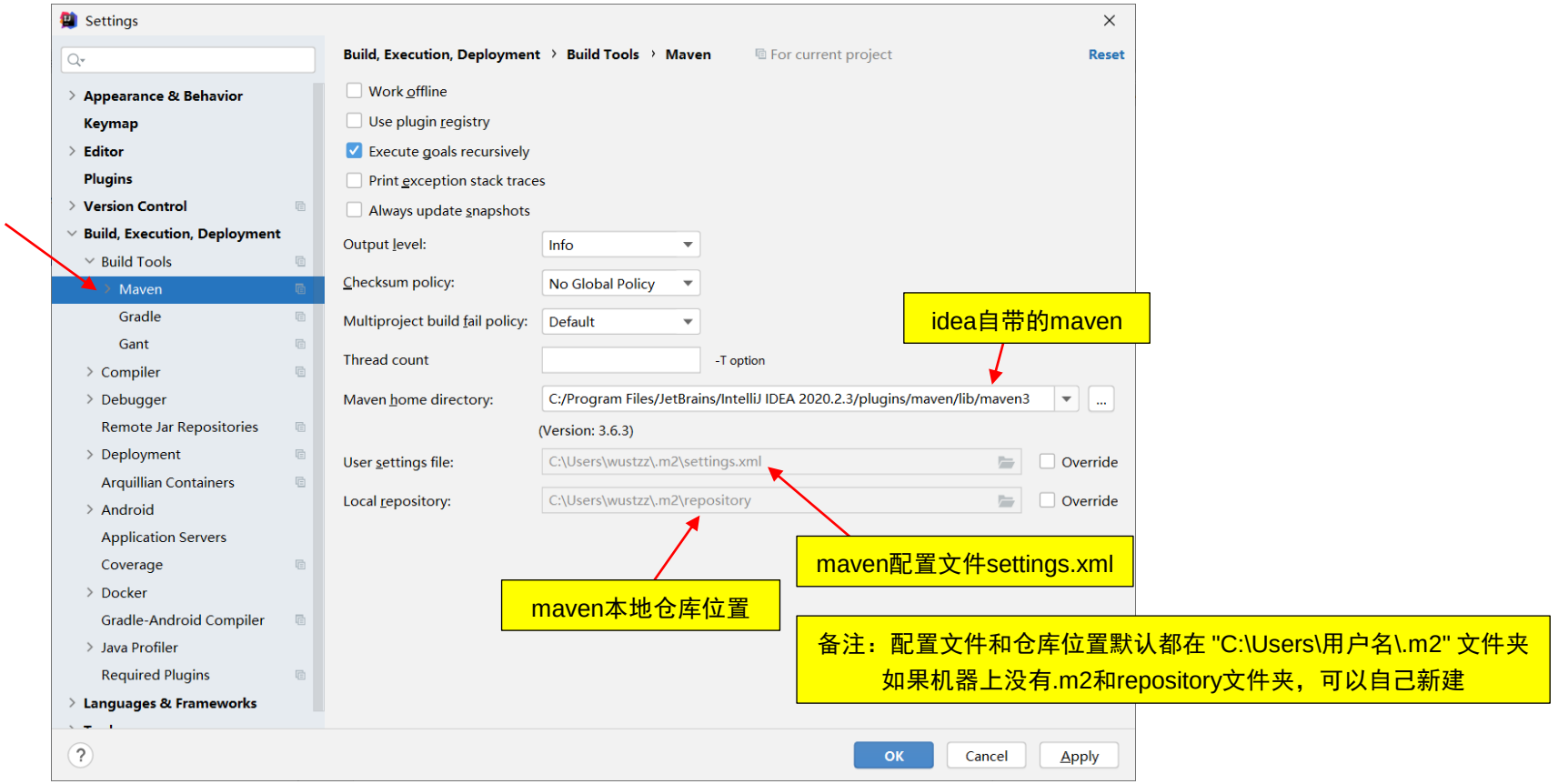
3. 配置Maven

Maven是一种基于项目对象模型(POM project object model)，通过配置文件(pom.xml)来管理项目构建的一种软件项目管理工具。

- 启动Idea → Customize → All Settings… → 查看Maven配置



Maven配置


Settings
Build, Execution, Deployment › Build Tools › Maven
For current project
Reset
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Build Tools
Maven
Gradle
Gant
Compiler
Debugger
Remote Jar Repositories
Deployment
Arquillian Containers
Android
Application Servers
Coverage
Docker
Gradle-Android Compiler
Java Profiler
Required Plugins
Languages & Frameworks
Work offline
Use plugin registry
Execute goals recursively
Print exception stack traces
Always update snapshots
Output level:
Info
Checksum policy:
No Global Policy
Multiproject build fail policy:
Default
Thread count
-T option
Maven home directory:
C:/Program Files/JetBrains/IntelliJ IDEA 2020.2.3/plugins/maven/lib/maven3
(Version: 3.6.3)
User settings file:
C:\Users\wustzz\.m2\settings.xml
Override
Local repository:
C:\Users\wustzz\.m2\repository
Override
OK
Cancel
Apply
idea自带的maven
maven配置文件settings.xml
maven本地仓库位置
备注：配置文件和仓库位置默认都在 "C:\Users\用户名\.m2" 文件夹
如果机器上没有.m2和repository文件夹，可以自己新建

配置maven仓库镜像

- 在 .m2 文件夹中新建 settings.xml 文件：

settings.xml文件

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <mirrors>
    <mirror>
        <id>nexus-aliyun</id>
        <mirrorOf>*</mirrorOf>
        <name>Nexus aliyun</name>
        <url>http://maven.aliyun.com/nexus/content/groups/public</url>
    </mirror>
  </mirrors>
</settings>
```

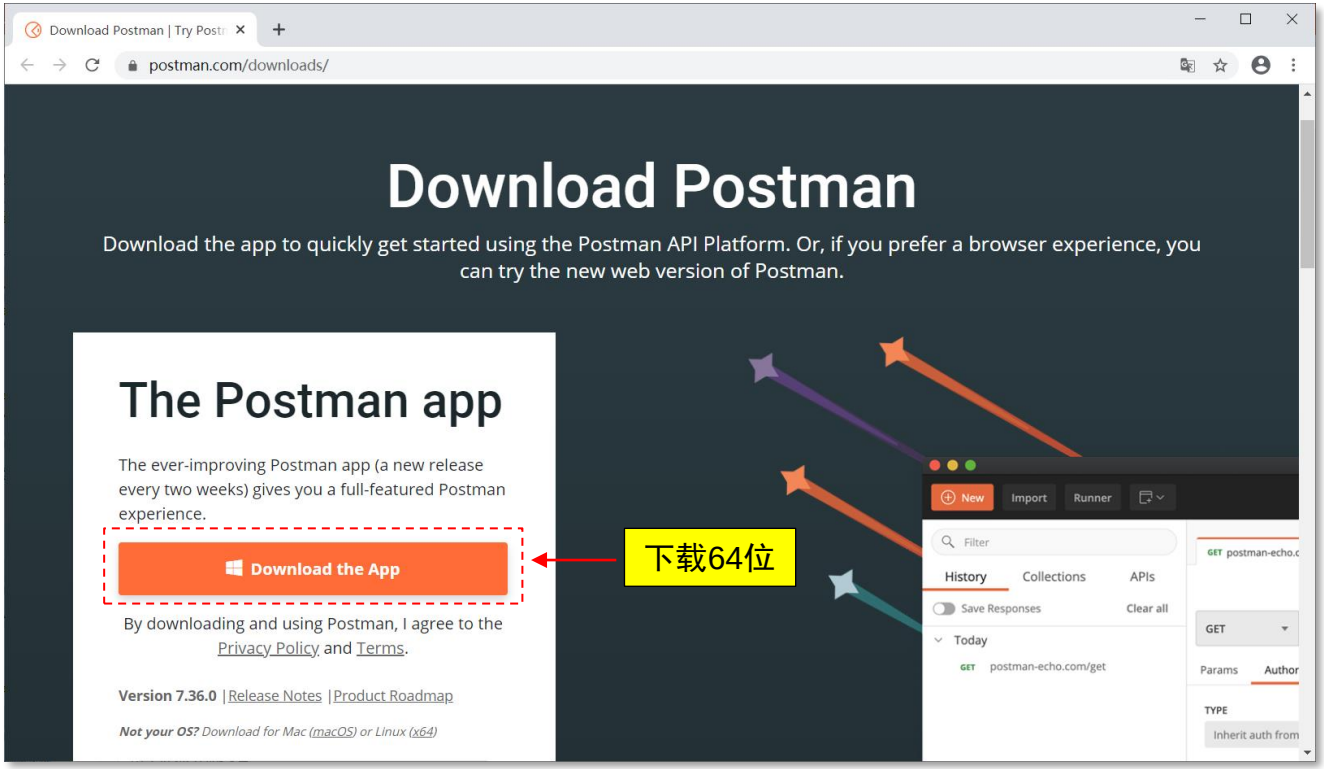
使用阿里云maven仓库镜像来添加依赖

【返回】

4. 安装Postman

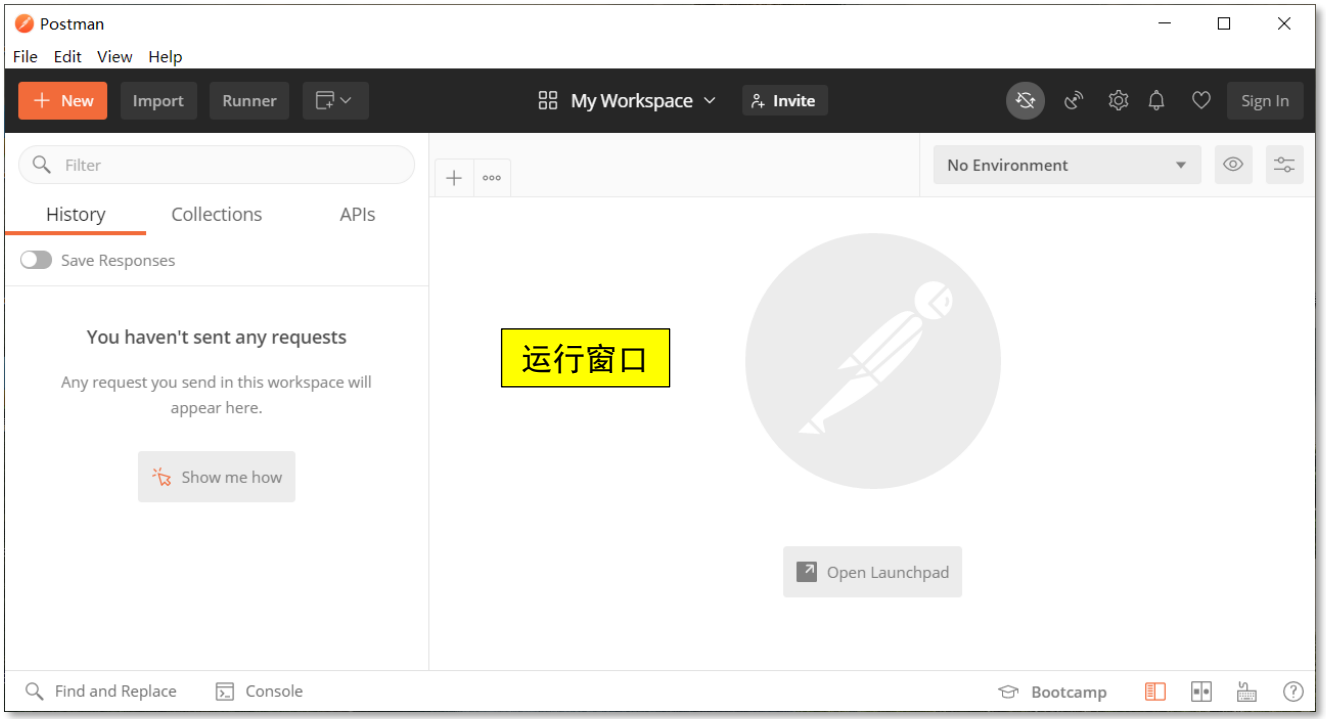
Postman是一个 http 请求模拟工具，常用于测试后台接口

- 官网： https://www.postman.com/downloads/



Postman安装教程

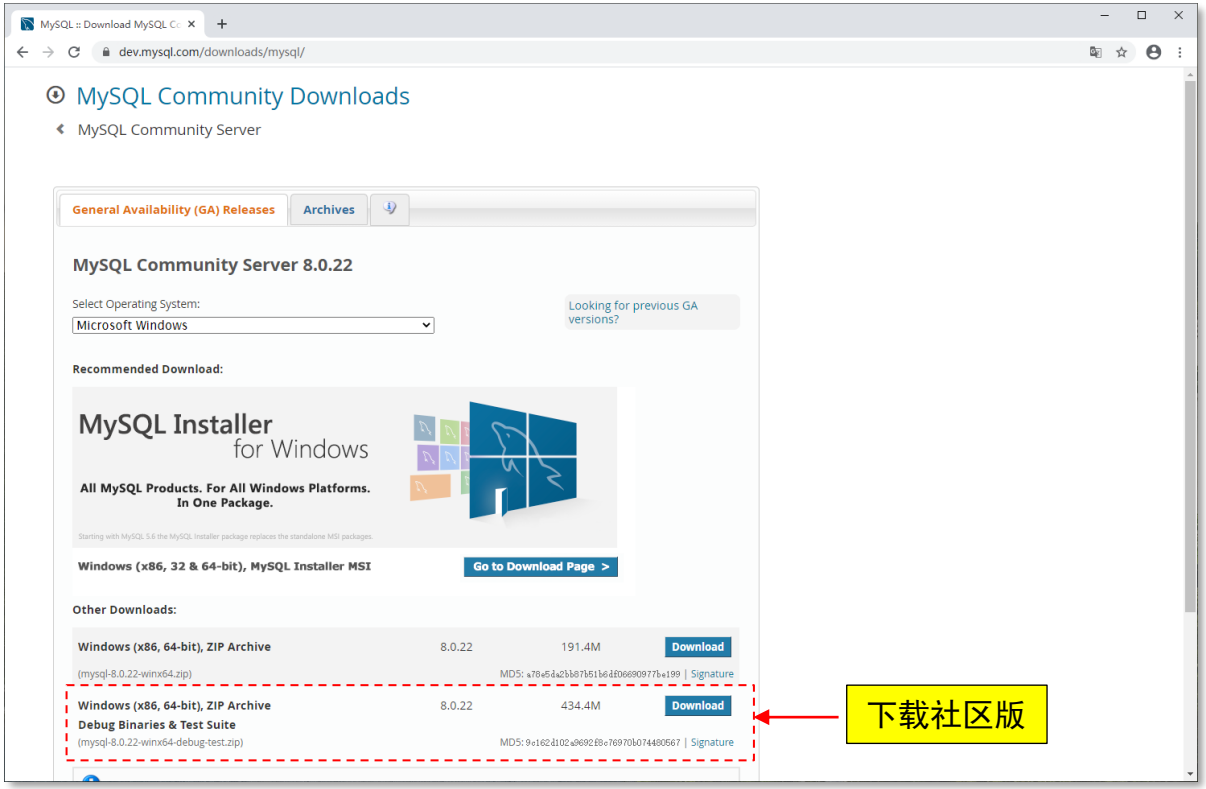
- https://blog.csdn.net/weixin_43184774/article/details/100578557



【返回】

5. 安装Mysql数据库（社区版）

- 官网： https://dev.mysql.com/downloads/mysql/

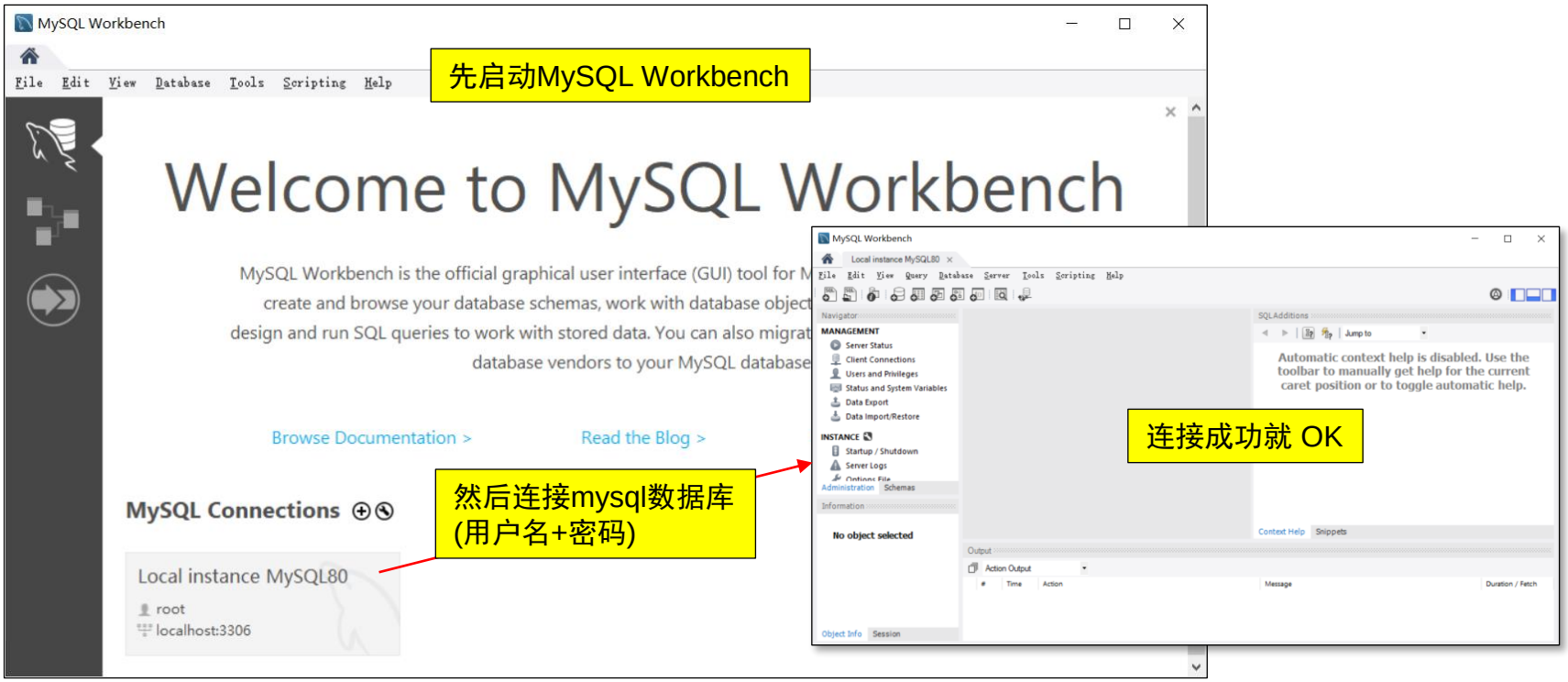


Mysql社区版安装教程

- https://blog.csdn.net/qq_44116023/article/details/108723390
- https://blog.csdn.net/weixin_40845165/article/details/84104842

使用MySQL Workbench连接数据库

- Workbench 提供可视化SQL开发、数据库建模、数据库管理等功能。



【返回】

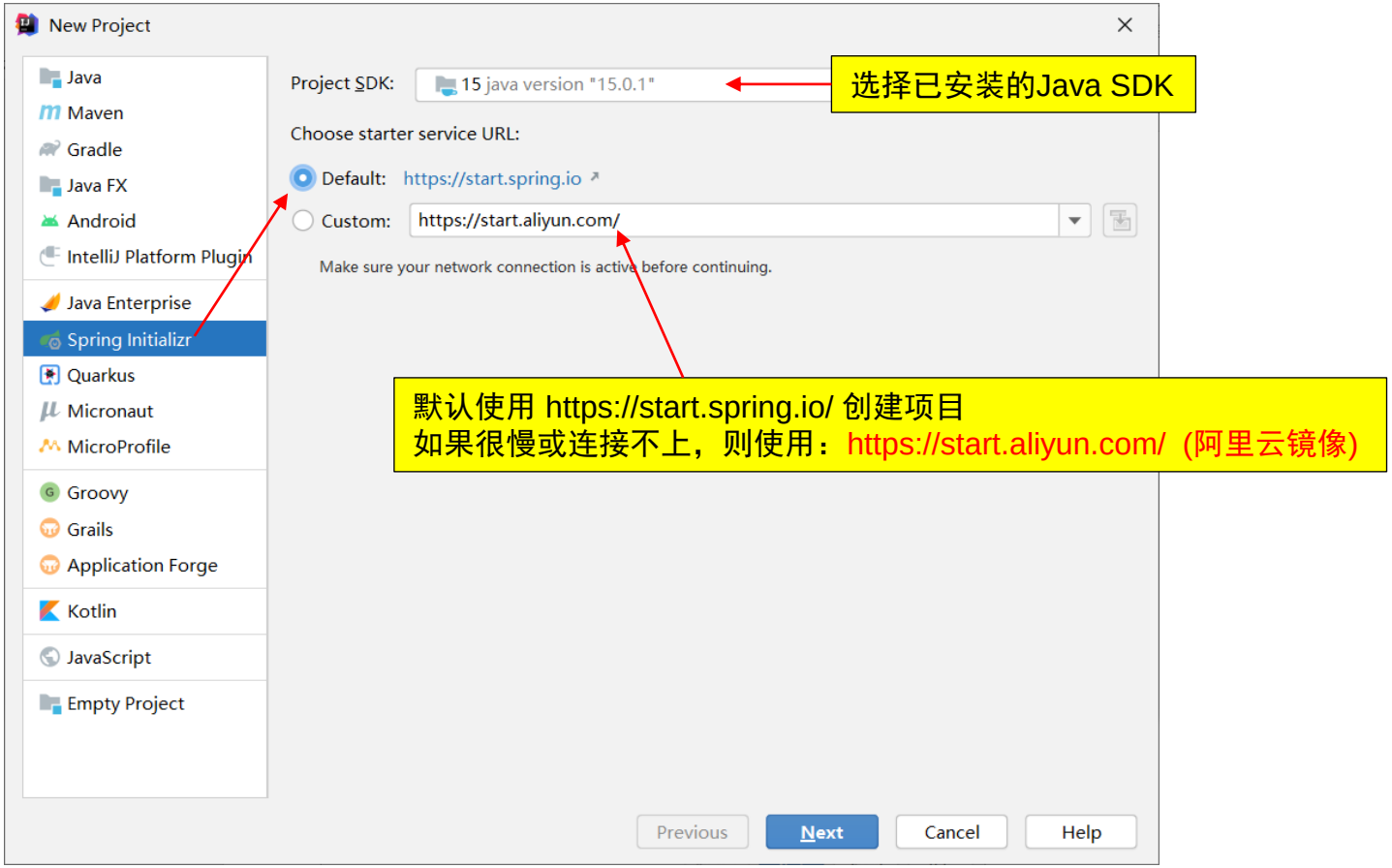
1.4 Hello MVC

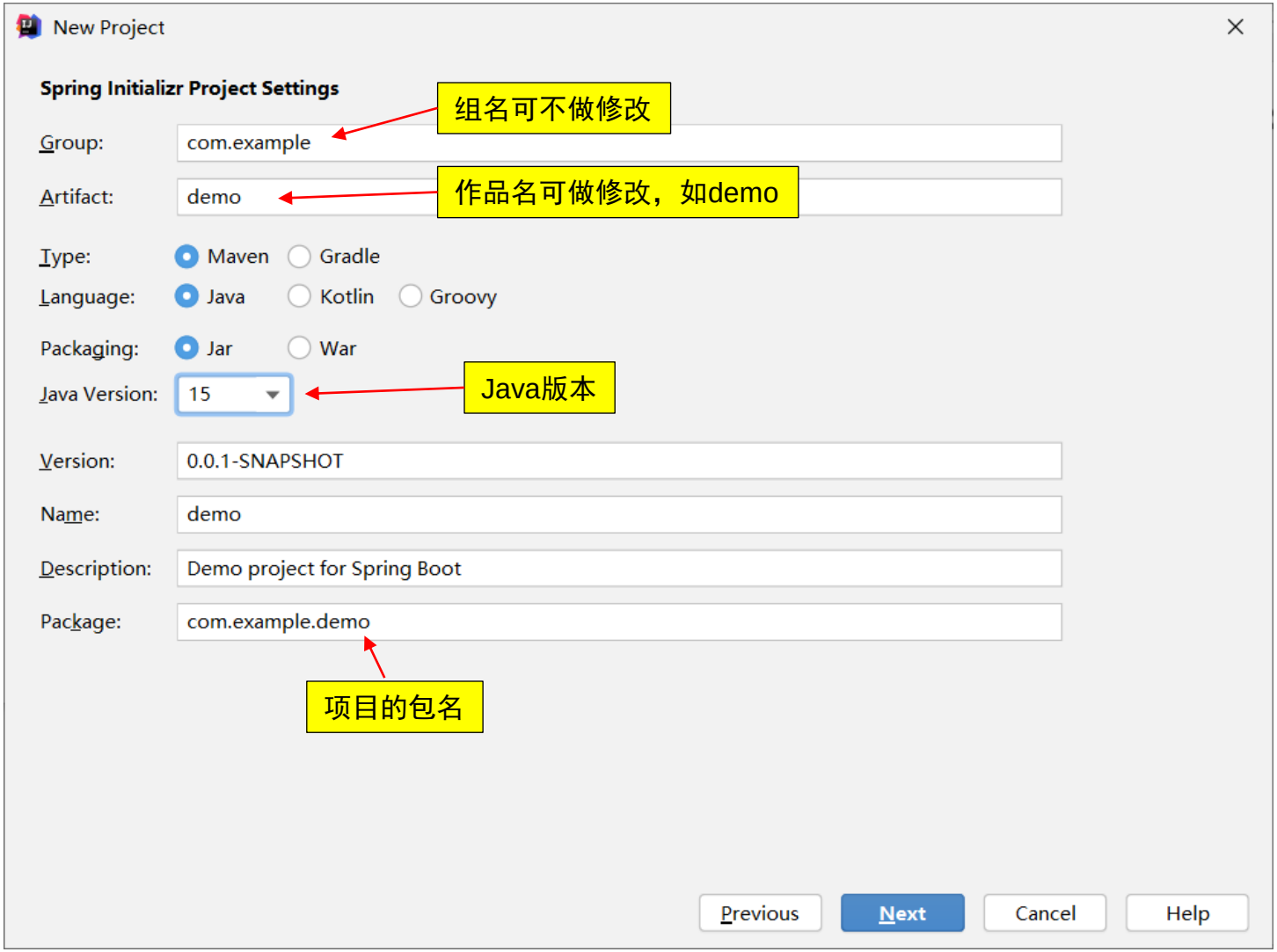
- 新建项目
- 添加控制器（Controller）
- 添加视图（View）
- 添加模型（Model）

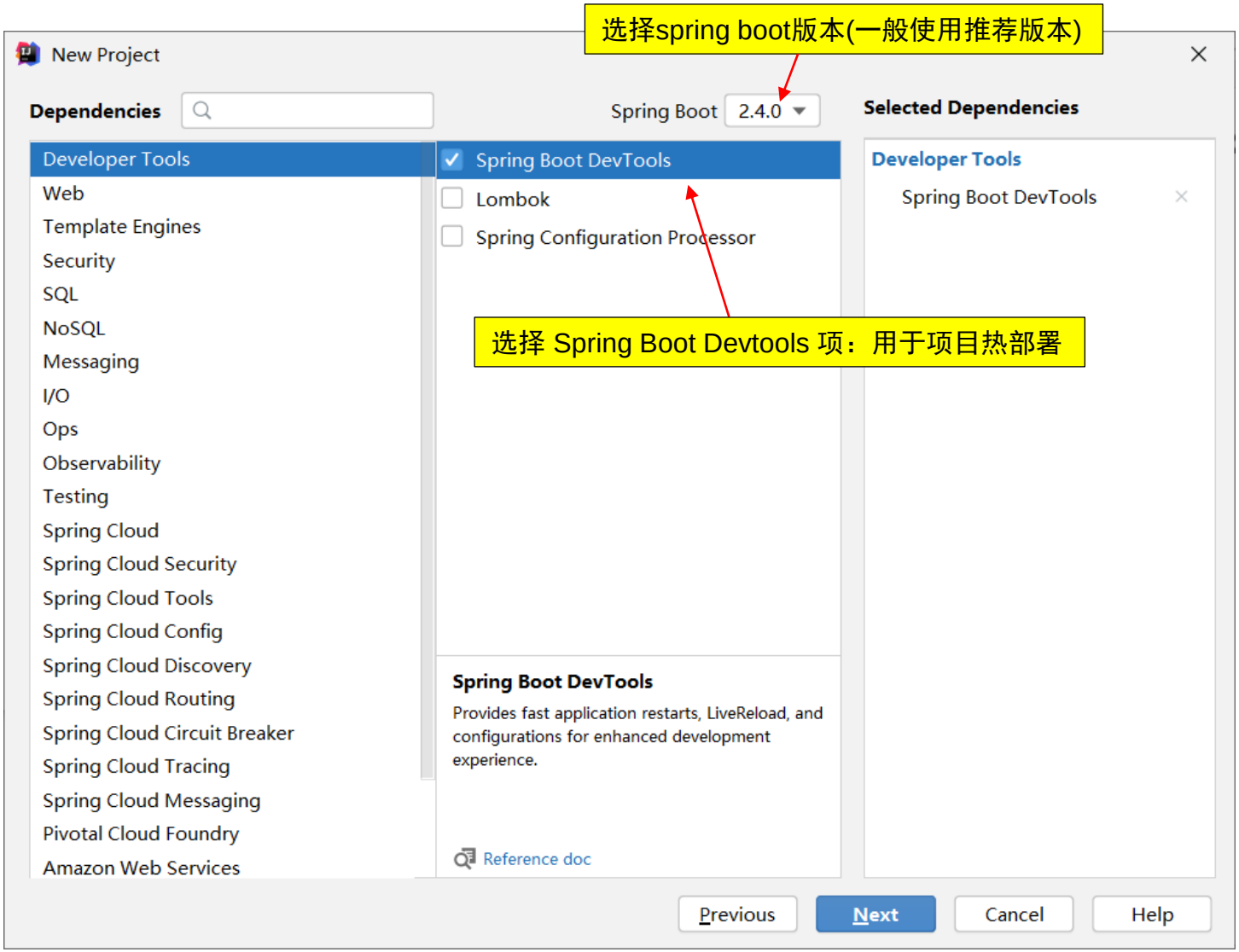
【返回】

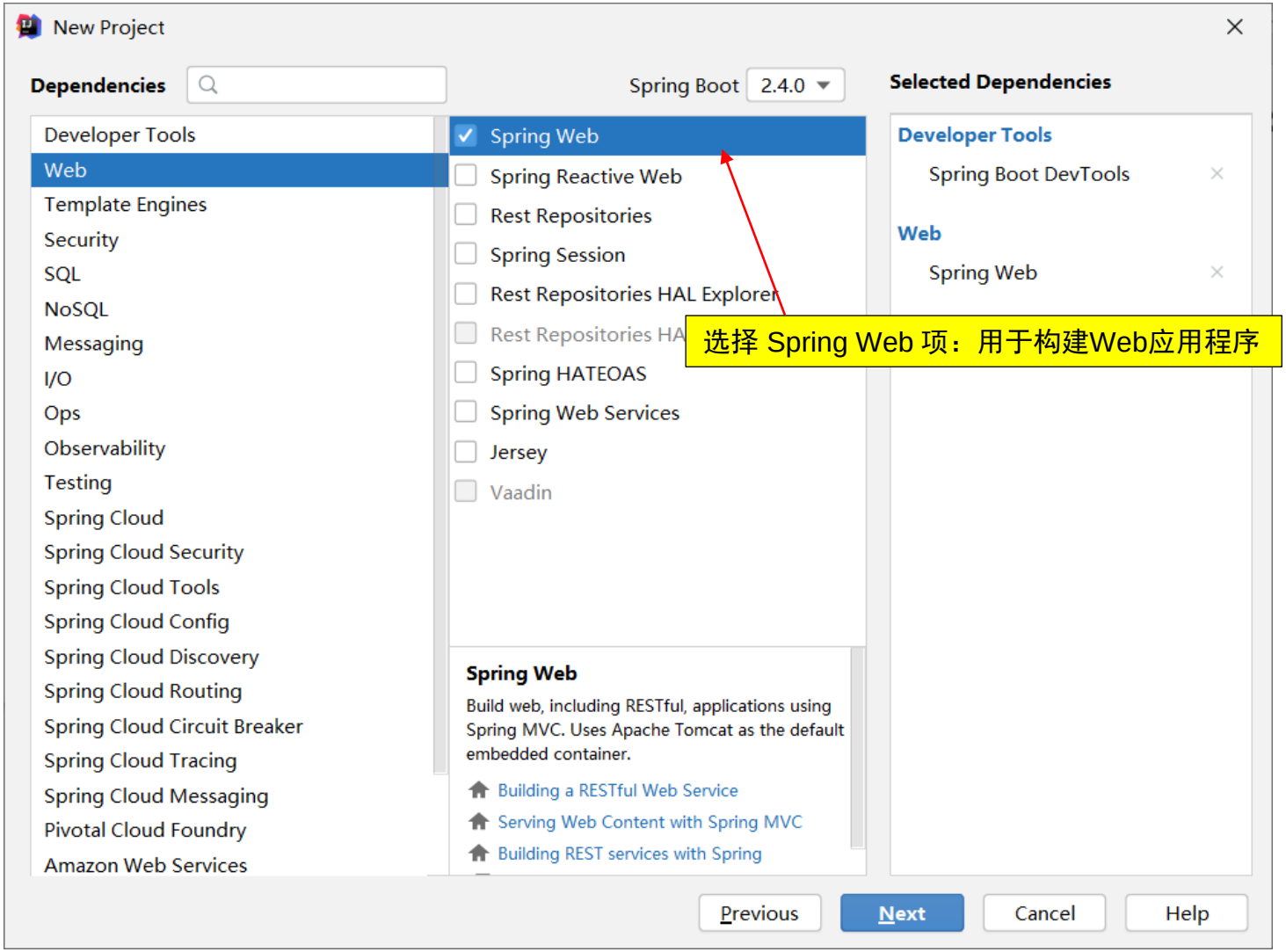
1. 新建项目

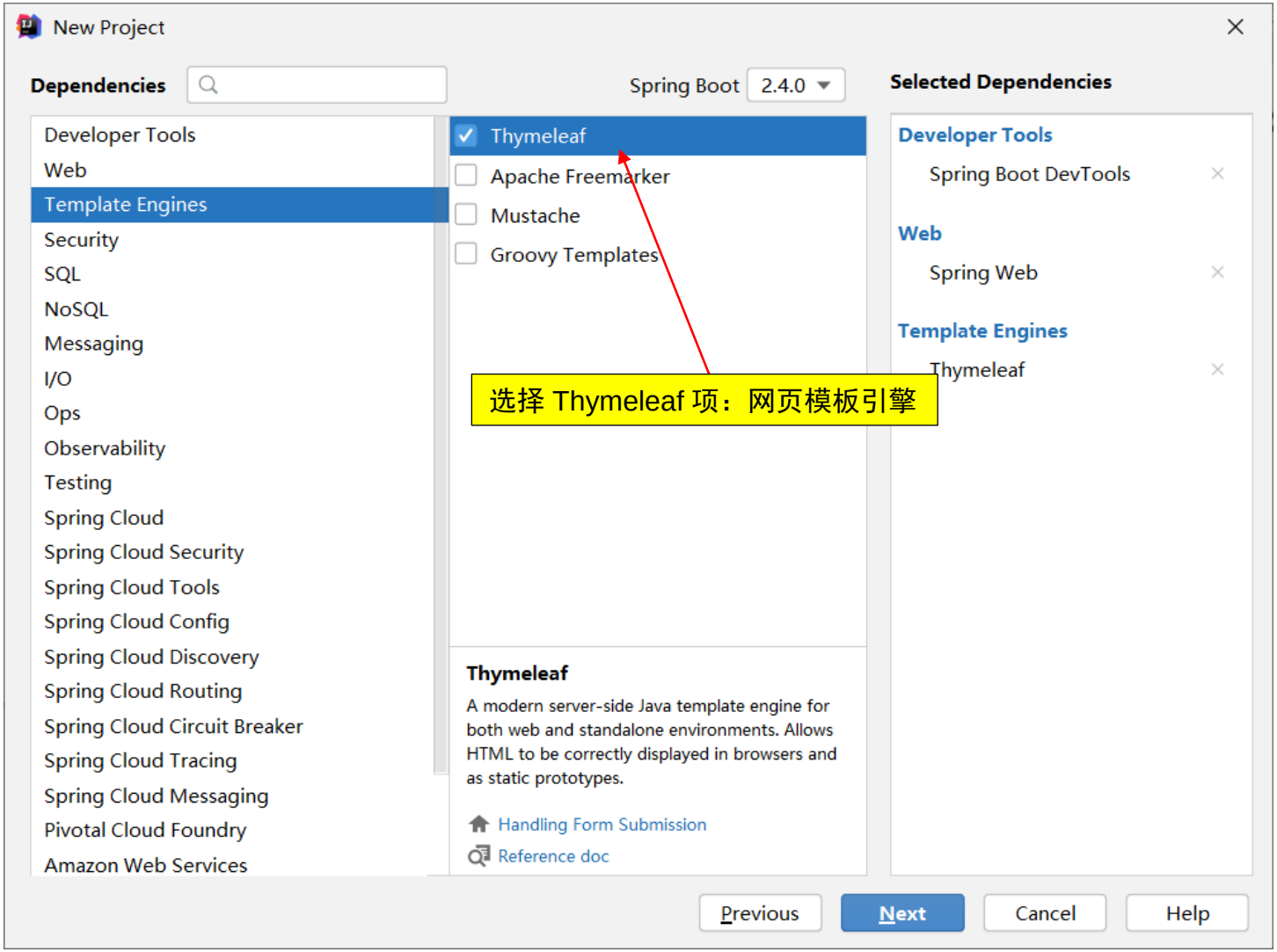
- New Project → 通过 Spring Initializr 创建 Spring Boot 项目。

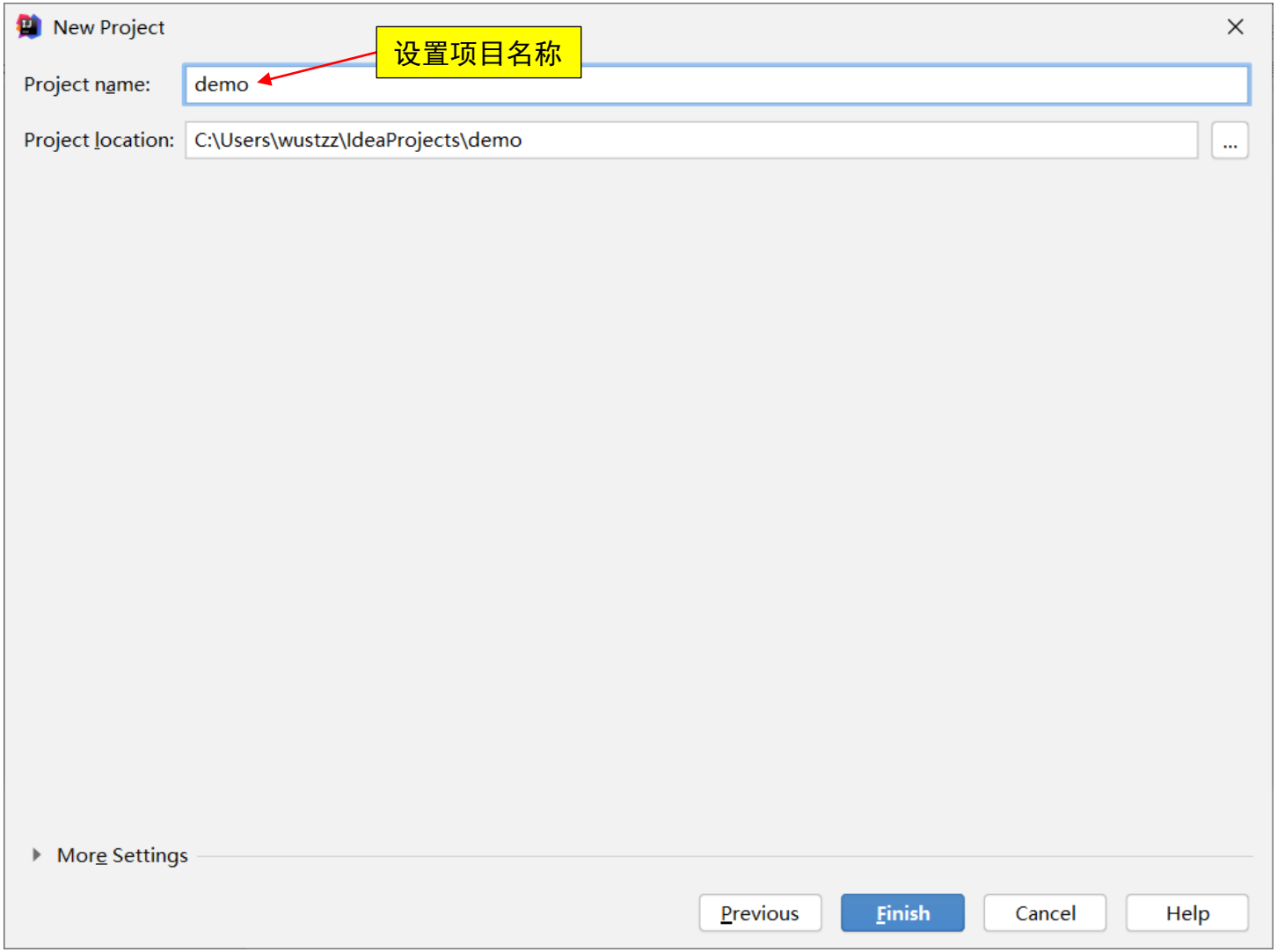



New Project
Spring Initializr Project Settings
组名可不做修改
Group: com.example
作品名可做修改，如demo
Artifact: demo
Type: Maven Gradle
Language: Java Kotlin Groovy
Packaging: Jar War
Java版本
Java Version: 15
Version: 0.0.1-SNAPSHOT
Name: demo
Description: Demo project for Spring Boot
Package: com.example.demo
项目的包名
Previous
Next
Cancel
Help

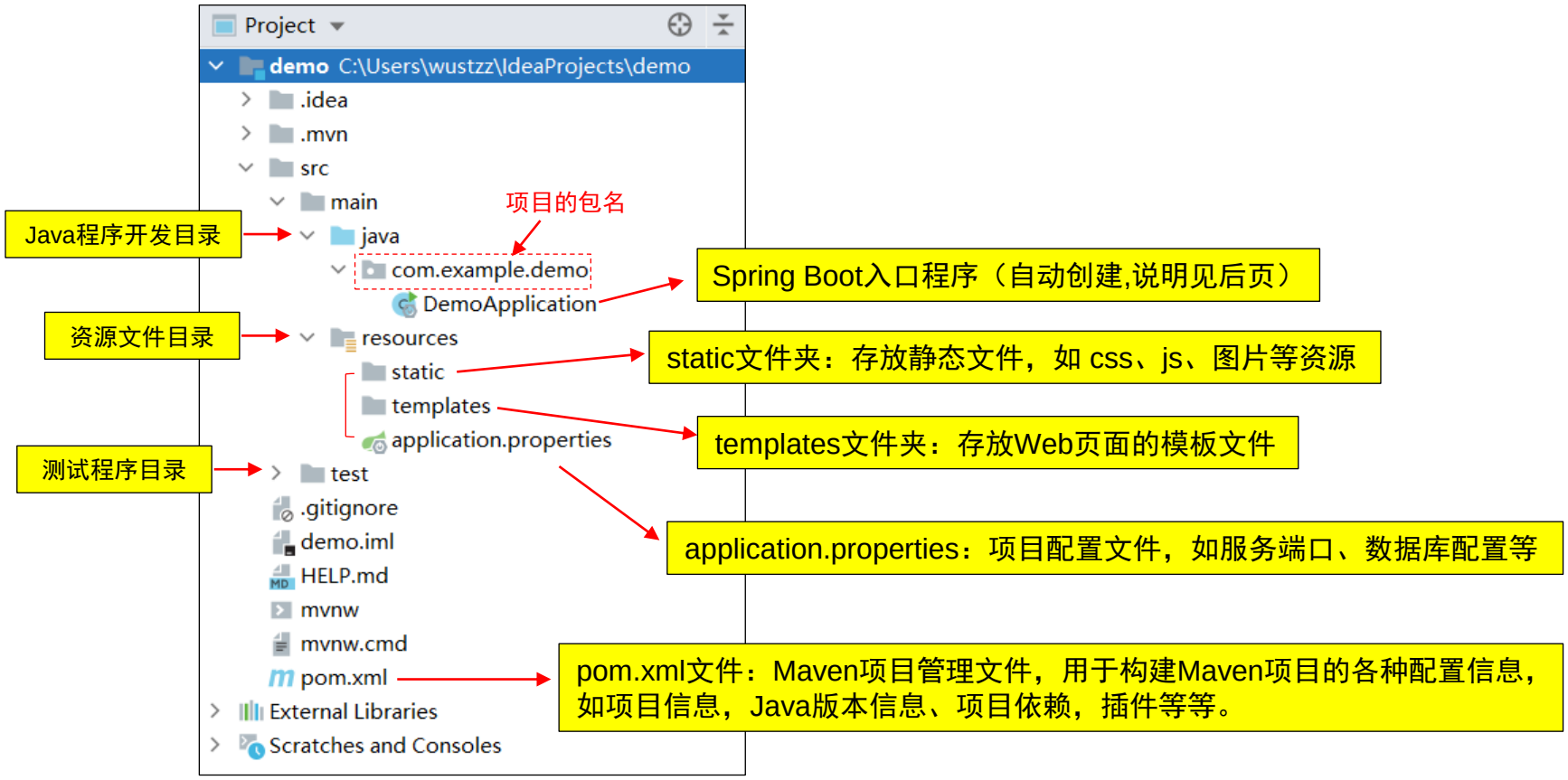

选择spring boot版本(一般使用推荐版本)
New Project
Dependencies
Spring Boot 2.4.0
Selected Dependencies
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Tracing
Spring Cloud Messaging
Pivotal Cloud Foundry
Amazon Web Services
Spring Boot DevTools
Lombok
Spring Configuration Processor
选择 Spring Boot Devtools 项：用于项目热部署
Developer Tools
Spring Boot DevTools
Spring Boot DevTools
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
Reference doc
Previous
Next
Cancel
Help


New Project
Dependencies
Spring Boot 2.4.0
Selected Dependencies
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Tracing
Spring Cloud Messaging
Pivotal Cloud Foundry
Amazon Web Services
Spring Web
Spring Reactive Web
Rest Repositories
Spring Session
Rest Repositories HAL Explorer
Rest Repositories HA
Spring HATEOAS
Spring Web Services
Jersey
Vaadin
Developer Tools
Spring Boot DevTools
Web
Spring Web
选择 Spring Web 项：用于构建Web应用程序
Spring Web
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
Building a RESTful Web Service
Serving Web Content with Spring MVC
Building REST services with Spring
Previous
Next
Cancel
Help


New Project
Dependencies
Spring Boot 2.4.0
Selected Dependencies
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Tracing
Spring Cloud Messaging
Pivotal Cloud Foundry
Amazon Web Services
Thymeleaf
Apache Freemarker
Mustache
Groovy Templates
选择 Thymeleaf 项：网页模板引擎
Thymeleaf
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.
Handling Form Submission
Reference doc
Developer Tools
Spring Boot DevTools
Web
Spring Web
Template Engines
Thymeleaf
Previous
Next
Cancel
Help

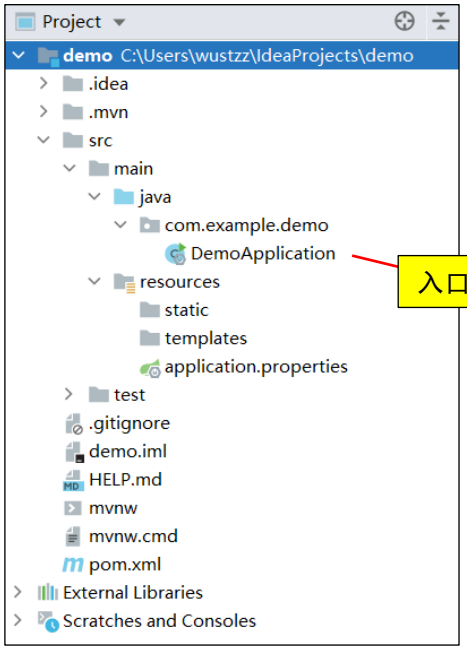

New Project
设置项目名称
Project name:
demo
Project location:
C:\Users\wustzz\IdeaProjects\demo
...
More Settings
Previous
Finish
Cancel
Help

demo项目结构


Project
demo C:\Users\wustzz\IdeaProjects\demo
.idea
.mvn
src
main
java
com.example.demo
DemoApplication
resources
static
templates
application.properties
test
.gitignore
demo.iml
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
项目的包名
Java程序开发目录
资源文件目录
测试程序目录
Spring Boot入口程序（自动创建,说明见后页）
static文件夹：存放静态文件，如 css、js、图片等资源
templates文件夹：存放Web页面的模板文件
application.properties：项目配置文件，如服务端口、数据库配置等
pom.xml文件：Maven项目管理文件，用于构建Maven项目的各种配置信息，如项目信息，Java版本信息、项目依赖，插件等等。

Spring Boot入口程序(类)

入口类默认名：项目名+Application



入口类

```
package com.example.demo;

import ...

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

入口类 用 @SpringBootApplication 注解

使用 SpringApplication 的静态 run 方法来启动当前应用程序

什么是注解式编程

- 未来框架的趋势是“约定大于配置”，代码的封装会更严密。开发人员会将更多的精力放在代码的整体优化和业务逻辑上，注解式编程会被更加广泛地使用。
- 注解（annotation）可用来定义一个类、属性或方法，以便程序能被编译处理。它相当于一个说明，告诉应用程序某个被注解的类或属性是什么，要怎么处理。
- SpringBoot提供了大量的注解。

pom.xml：Maven项目管理文件

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.4.0</version>
</parent>

<groupId>com.example</groupId>
<artifactId>demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>demo</name>

<properties>
        <java.version>15</java.version>
</properties>
```

Spring Boot父级依赖，这样当前的项目就是Spring Boot项目了
它用来提供相关的Maven默认依赖。使用它之后，常用的包依赖
可以省去version标签。

项目信息

Java版本信息

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

各种依赖项

Thymeleaf 网页模板引擎

用于构建 Web 应用程序的 starter 组件

starter组件：Spring Boot为了简化配置，提供了非常多的starter，它先打包好与常用模块相关的所有jar包，并完成自动配置，这使得开发时不需要过多关注框架配置，只需关注业务逻辑即可

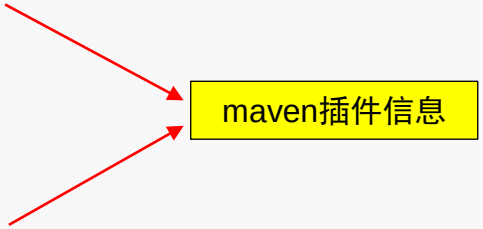
包含 RESTful风格框架、SpringMVC 和默认的嵌入式容器 Tomcat

使Spring Boot应用支持热部署 ★

热部署：为了更好支持程序调试，在项目进行修改之后不需要耗费时间重启，在程序正运行情况下即可实时生效。以节约时间和操作，提高开发效率。

SpringBoot项目单元测试

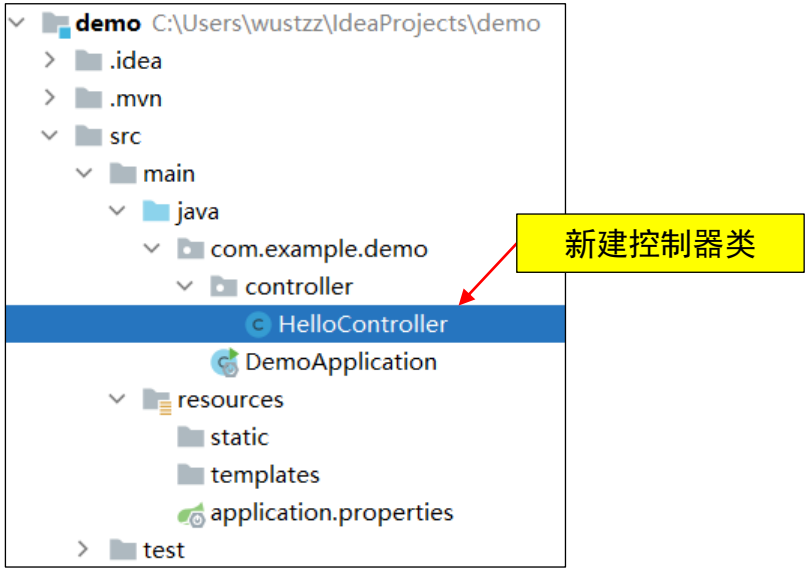
```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>14</source>
                <target>14</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```



【返回】

2. 添加控制器

- 在 com.example.demo 包中新建 controller 包
- 在 controller 包中新建类： HelloController



HelloController.java

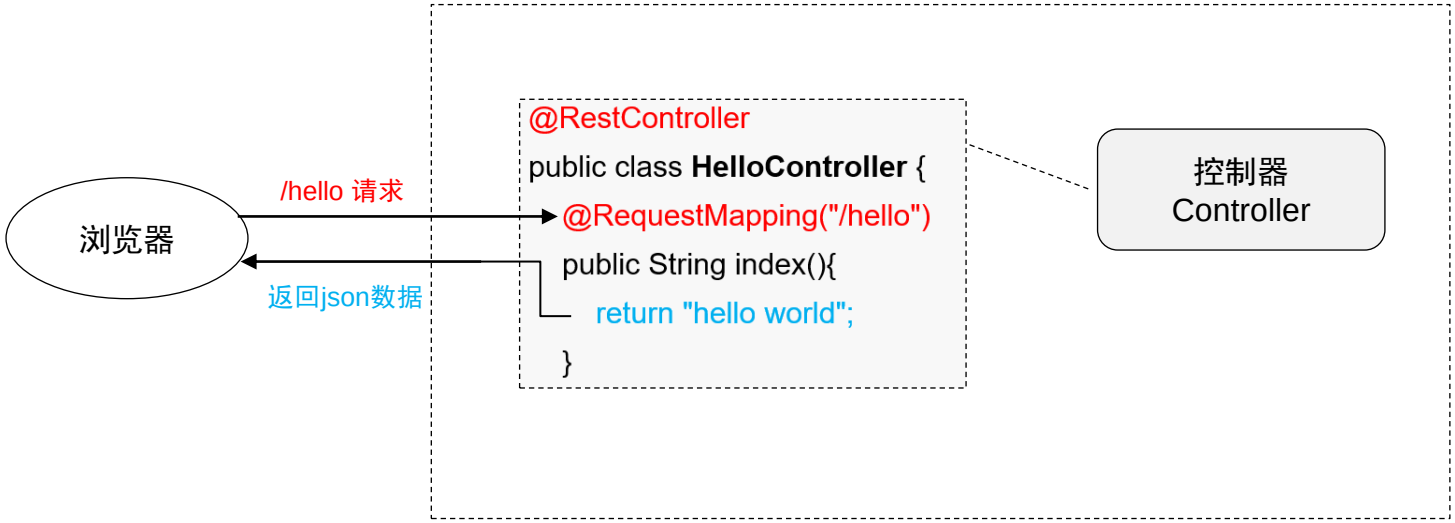
```
package com.example.demo.controller;
@RestController
public class HelloController {
    @RequestMapping("/hello")
    public String index(){
        return "hello world";
    }

    @RequestMapping("/test")
    public Map<String,String> test(){
        Map<String,String> map=new HashMap<String, String>();
        map.put("username","wustzz");
        map.put("password","123456");
        return map;
    }
}
```

@RestController 注解用于标注控制器类
该控制器里面的方法都以 json 格式输出

@RestController 相当于@Controller+@ResponseBody

@RequestMapping 注解用于配置 URL 映射
即确定请求所对应的处理方法 (后页图示)

将 URL 映射到方法


@RestController
public class HelloController {
@RequestMapping("/hello")
public String index(){
return "hello world";
}
浏览器
/hello 请求
返回json数据
控制器
Controller

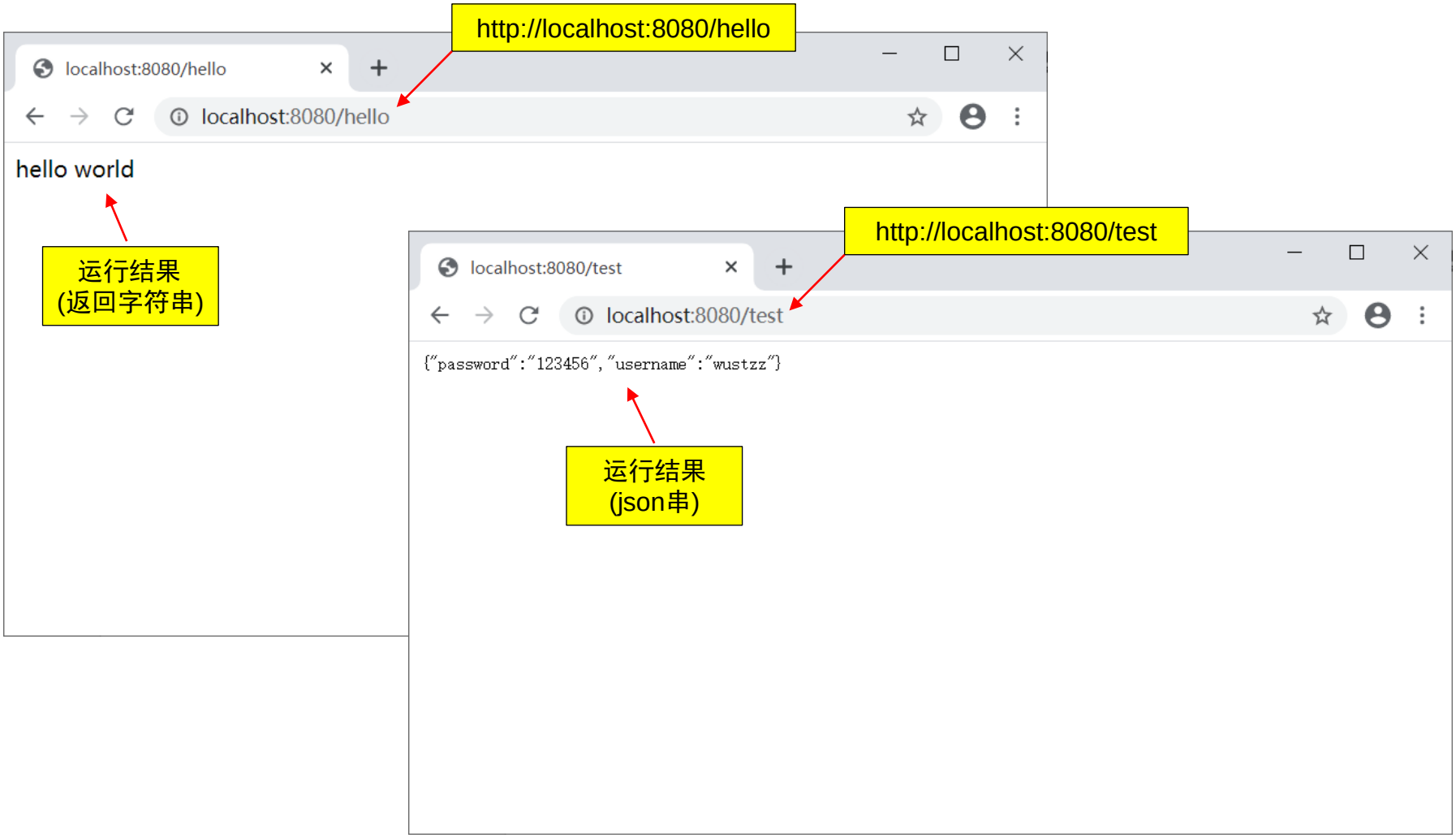
启动应用程序

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.0)

2020-11-23 19:11:40.930  INFO 10860 --- [  restartedMain] com.example.demo.DemoApplication         : Starting DemoApplication using Java 15.0.1 on DESKTOP-78KGVHD with P
2020-11-23 19:11:40.931  INFO 10860 --- [  restartedMain] com.example.demo.DemoApplication         : No active profile set, falling back to default profiles: default
2020-11-23 19:11:40.975  INFO 10860 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properti
2020-11-23 19:11:40.975  INFO 10860 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.lev
2020-11-23 19:11:41.694  INFO 10860 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2020-11-23 19:11:41.702  INFO 10860 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-11-23 19:11:41.702  INFO 10860 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.39]
2020-11-23 19:11:41.763  INFO 10860 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2020-11-23 19:11:41.763  INFO 10860 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 787 ms
2020-11-23 19:11:41.878  INFO 10860 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2020-11-23 19:11:41.986  WARN 10860 --- [  restartedMain] ion$DefaultTemplateResolverConfiguration : Cannot find template location: classpath:/templates/ (please add som
2020-11-23 19:11:42.027  INFO 10860 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2020-11-23 19:11:42.058  INFO 10860 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2020-11-23 19:11:42.067  INFO 10860 --- [  restartedMain] com.example.demo.DemoApplication         : Started DemoApplication in 1.417 seconds (JVM running for 2.187)
```

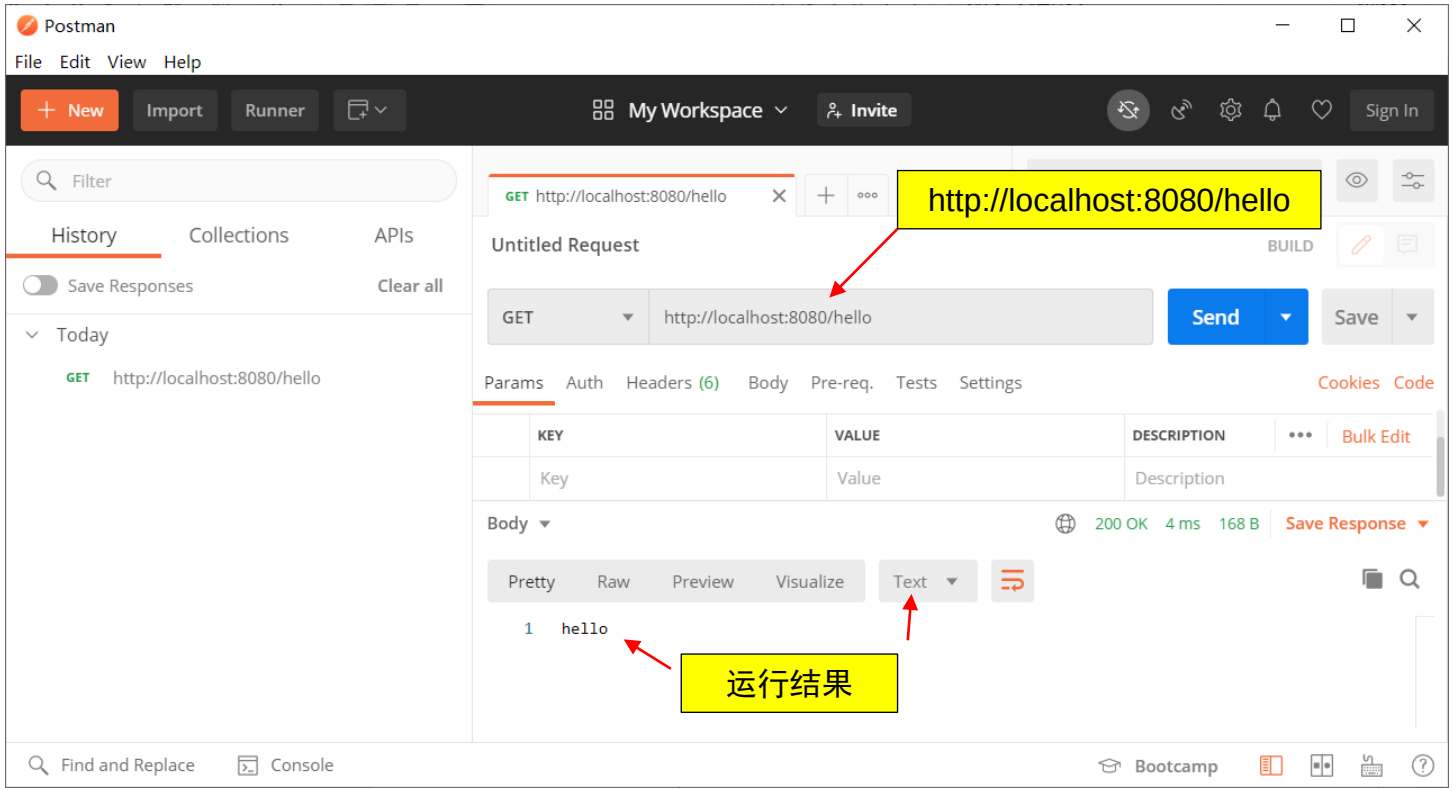
内置tomcat服务器
服务端口默认8080

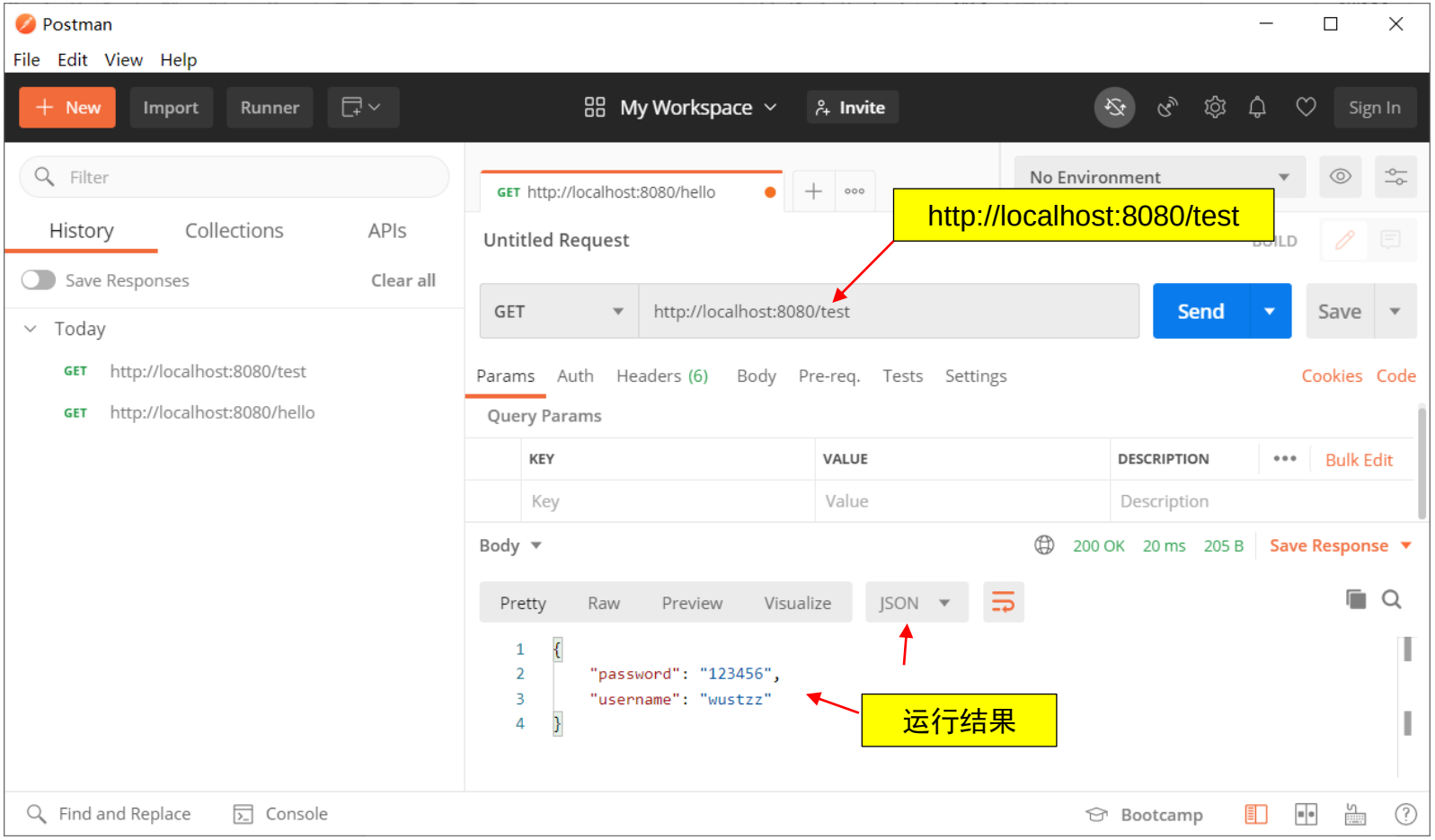
成功启动

在浏览器中测试请求：


http://localhost:8080/hello
localhost:8080/hello
localhost:8080/hello
hello world
运行结果
(返回字符串)
http://localhost:8080/test
localhost:8080/test
localhost:8080/test
{"password":"123456","username":"wustzz"}
运行结果
(json串)

使用Postman测试接口：




Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Sign In
Filter
History
Collections
APIs
Save Responses
Clear all
Today
GET http://localhost:8080/test
GET http://localhost:8080/hello
GET http://localhost:8080/hello
No Environment
http://localhost:8080/test
Untitled Request
GET
http://localhost:8080/test
Send
Save
Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
200 OK
20 ms
205 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "password": "123456",
3 "username": "wustzz"
4 }
运行结果
Find and Replace
Console
Bootcamp

修改控制器注解：

HelloController.java

```
package com.example.demo.controller;
@Controller
public class HelloController {
    @RequestMapping("/hello")
    public String index(){
        return "hello";
    }
}
```

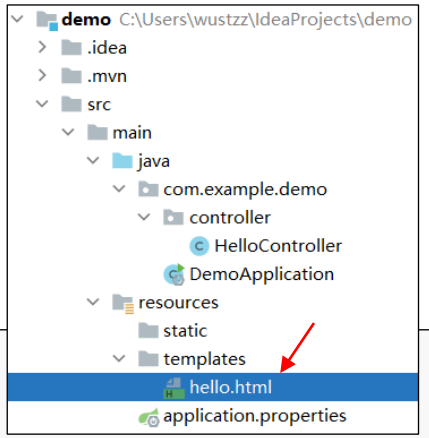
如果需要返回到指定页面，则需要用@Controller 注解控制器类

此时返回的是"hello.html"页面

【返回】

3. 添加视图：hello.html

- 在 templates 文件夹中新建 hello.html 页面：



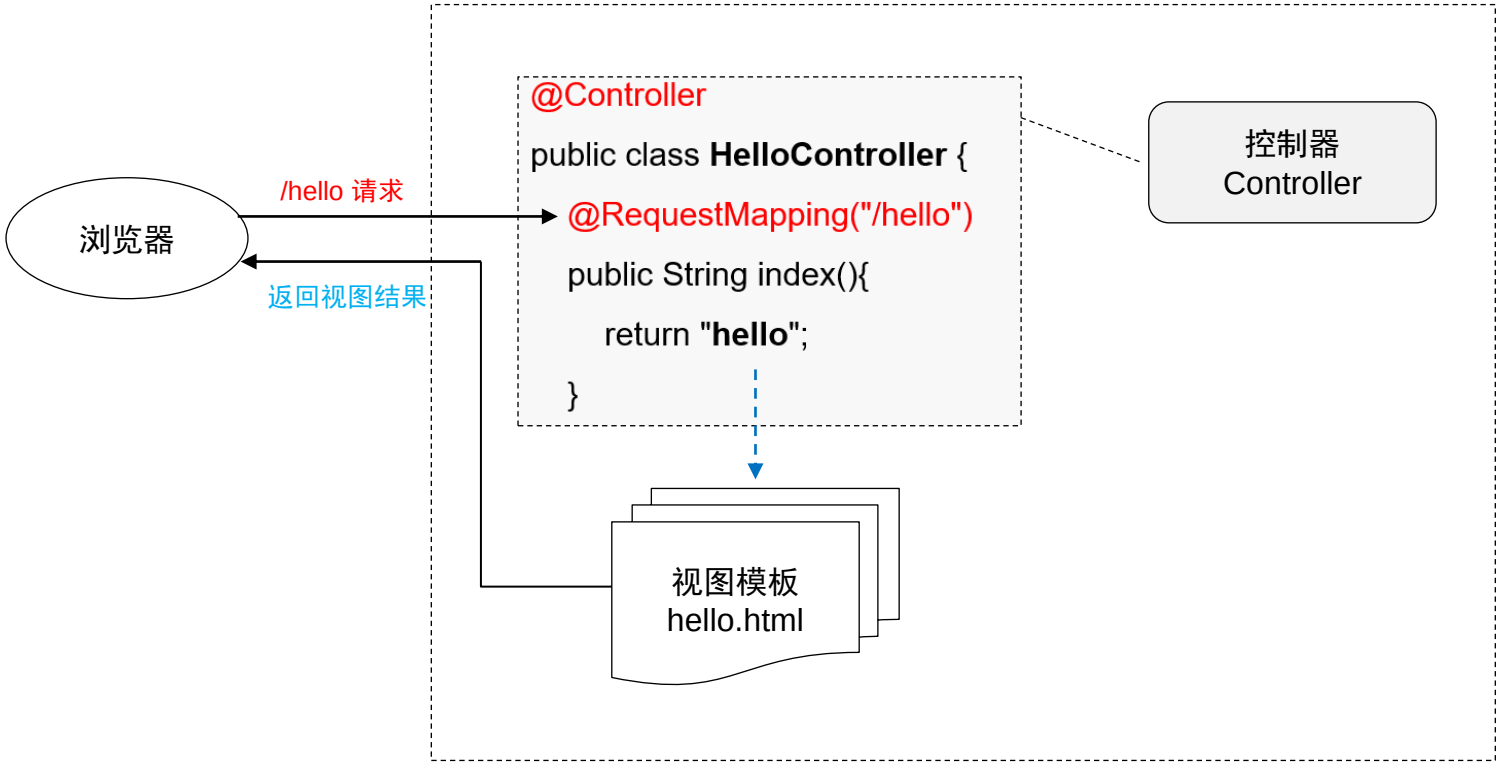

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>hello</title>
</head>
<body>
    <h1>hello world</h1>
</body>
</html>
```

表示html5页面

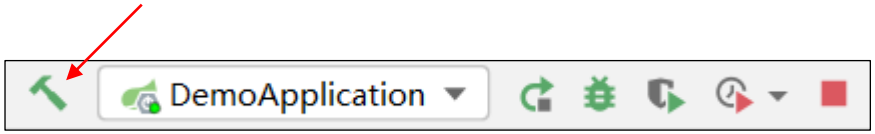
新添加的内容

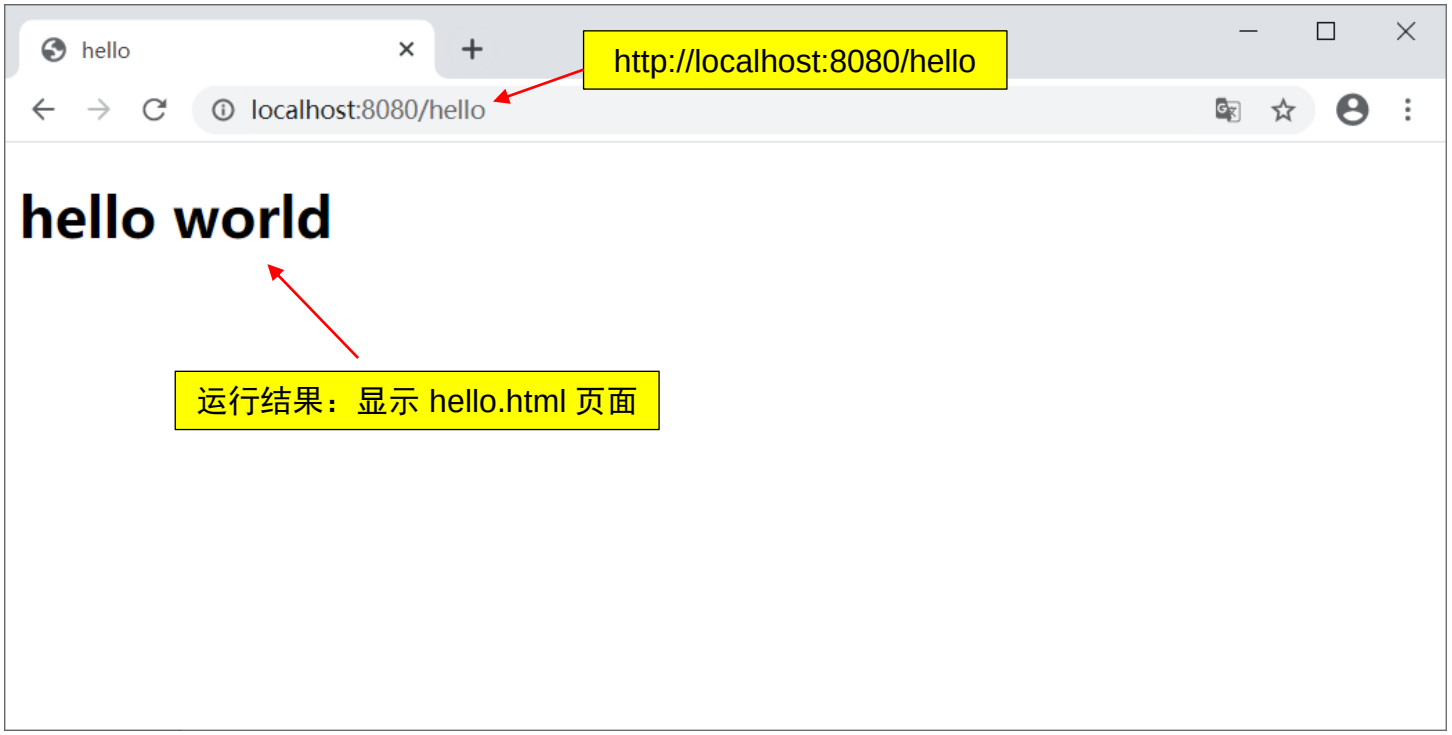
hello.html

将 URL 映射到方法


@Controller
public class HelloController {
@RequestMapping("/hello")
public String index(){
return "hello";
}
控制器
Controller
浏览器
/hello 请求
返回视图结果
视图模板
hello.html

点击“锤子”进行热部署，然后在浏览器中测试请求：





【返回】

4. 添加模型：传递数据

- 修改控制器代码：红色部分

```
@Controller
public class HelloController {
    @RequestMapping("/hello")
    public String index( Model model ){
        String  myname = "wustzz";
        model.addAttribute("name",myname);
        return "hello";
    }
}
```

方法中添加 Model 对象

Model对象添加属性和值(key-value)

属性名　　属性值

- 修改视图代码：红色和蓝色部分

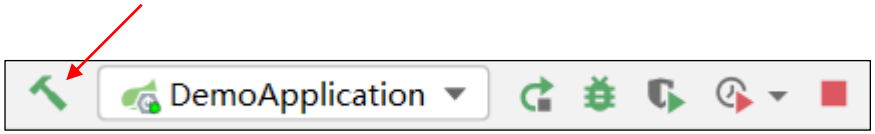
添加 Thymeleaf 命名空间

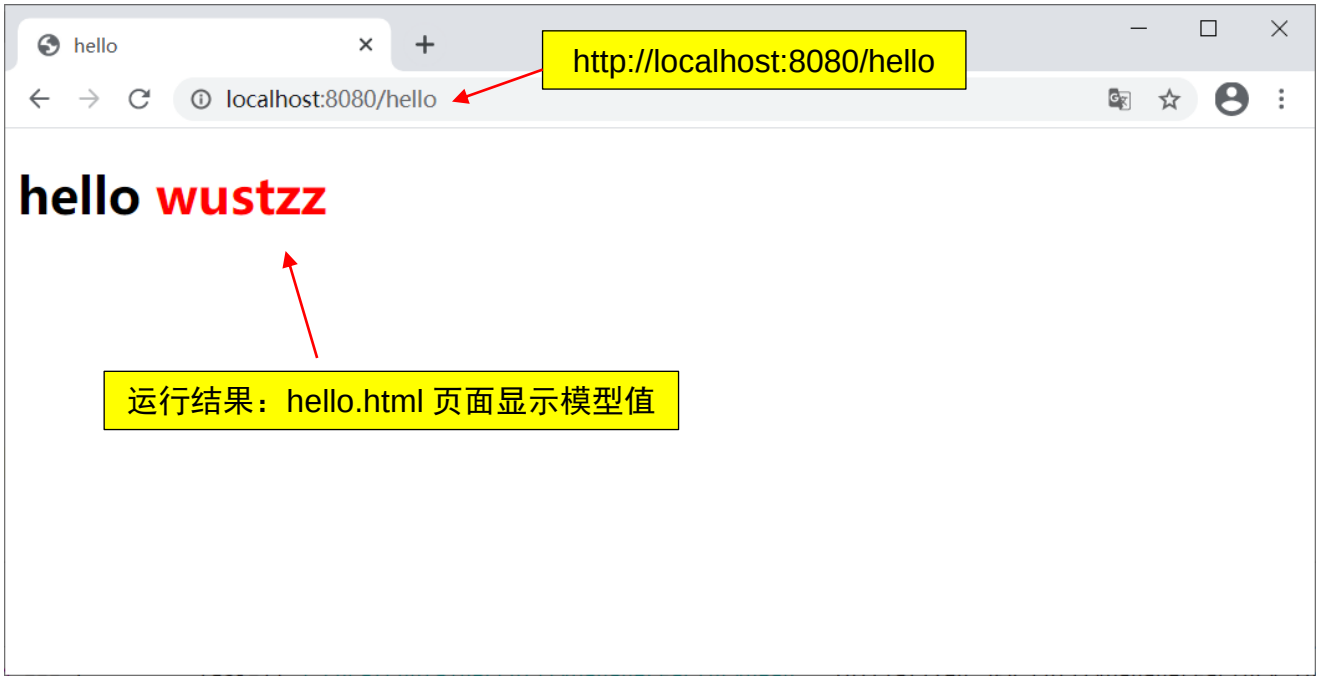
```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>hello</title>
</head>
<body>
    <h1>hello  <span th:text="${name}" style="color:red;">world</span></h1>
</body>
</html>
```

th:text：thymeleaf模板标签，用于进行文本替换

${ 属性变量名 }：变量表达式

点击“锤子”进行热部署，然后在浏览器中测试请求：

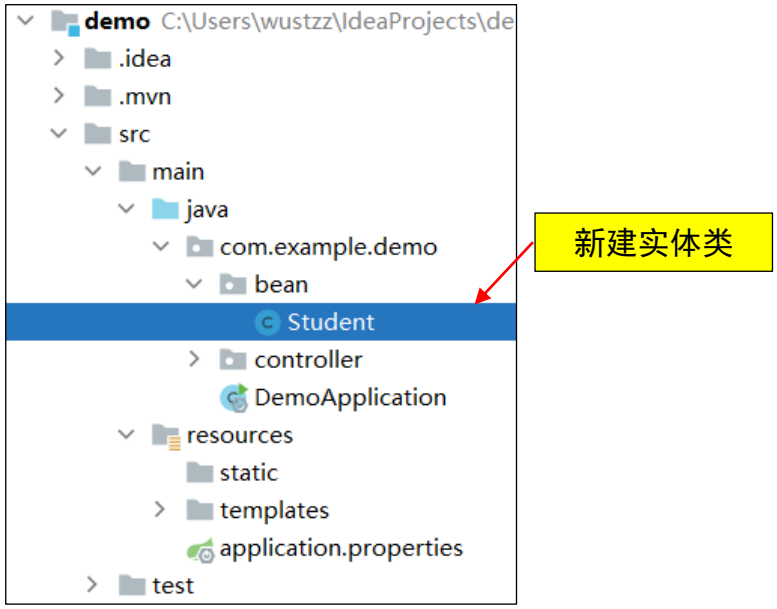




添加实体类模型

实体类模型也称作 bean、entity 或 pojo 等

- 在 com.example.demo 包中新建 bean 包
- 在 bean 包中新建类：Student



Student.java

```
package com.example.demo.bean;
public class Student {
    private Integer id;  //学号
    private String name;  //姓名
    public Student() {
    }
    public Student(Integer id, String name) {
        this.id = id;
        this.name = name;
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

模型属性

构造函数

getter/setter

修改控制器代码

- 修改控制器代码：红色部分

```
@Controller
public class HelloController {
    @RequestMapping("/hello")
    public String index( Model model ){
        Student s=new Student(2019001,"小明");
        model.addAttribute("stu",s);
        return "hello";
    }
}
```

方法中添加 Model 对象

创建对象

Model对象添加属性和值(key-value)

属性名

属性值

- 修改视图代码：红色和蓝色部分

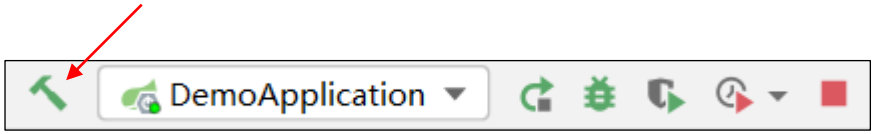
添加 Thymeleaf 命名空间

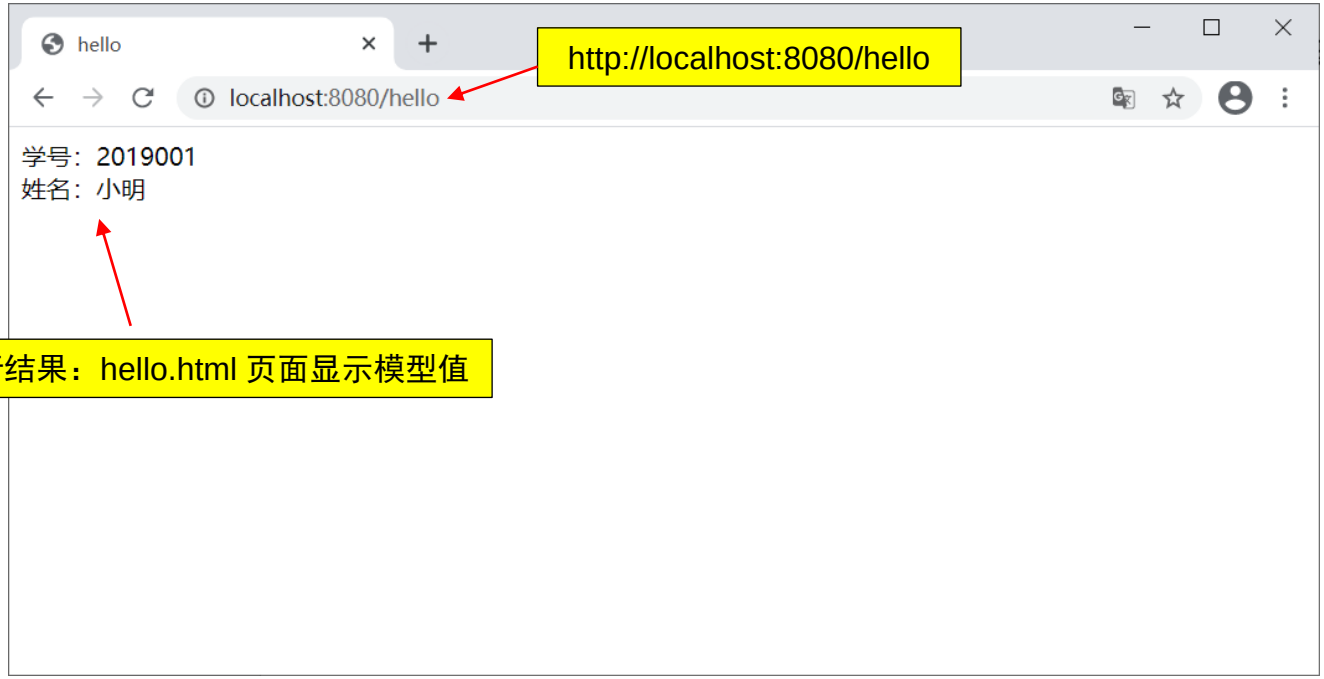
```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>hello</title>
</head>
<body>
学号：<p th:text="${stu.id}"></p> <br>
姓名：<p th:text="${stu.name}"></p>
</body>
</html>
```

th:text：thymeleaf模板标签，用于进行文本替换

${ 对象.属性名 }：变量表达式

点击“锤子”进行热部署，然后在浏览器中测试请求：





【返回】

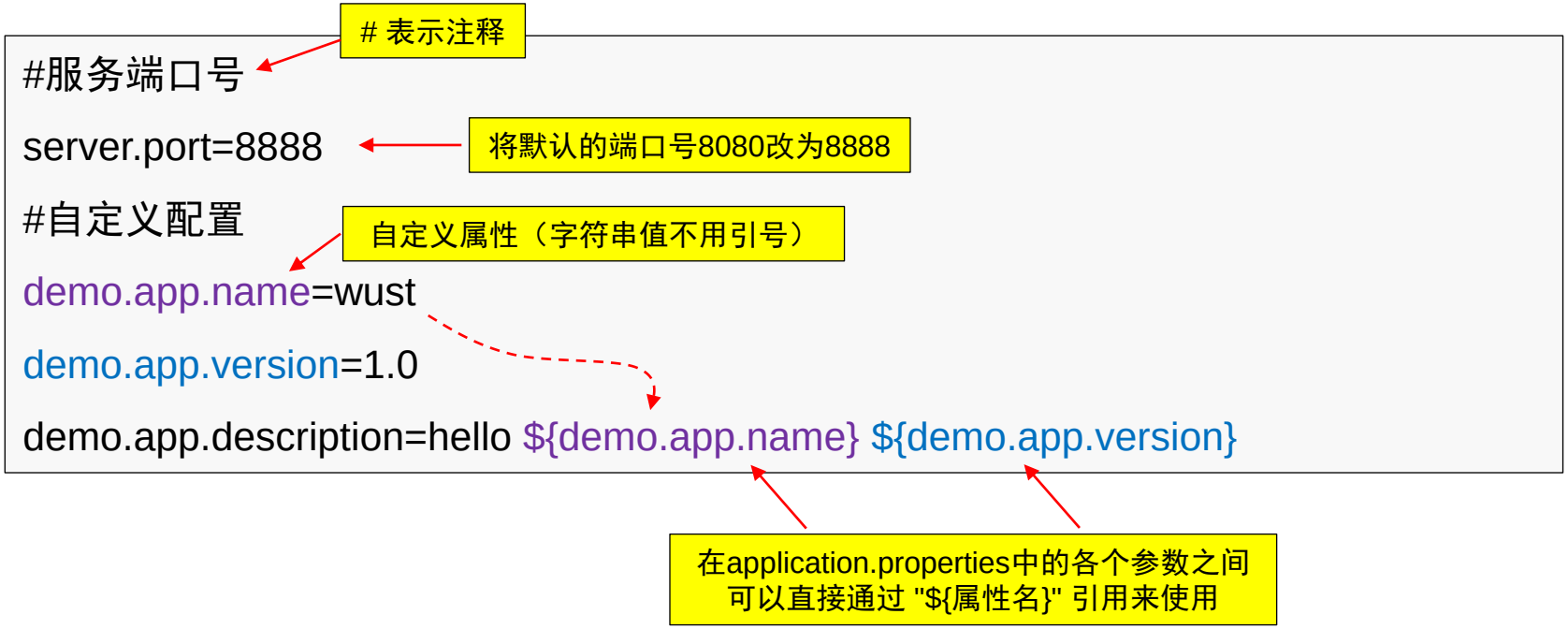
1.5 项目配置文件

- 配置文件有两种格式：
  - application.properties：以 properties 为结尾
  - application.yml：以 yml 或者 yaml 结尾

书写格式有差别

- Spring Boot几乎所有的配置都可以写在这个文件中，如果不配置，则使用默认配置。

配置文件使用示例：

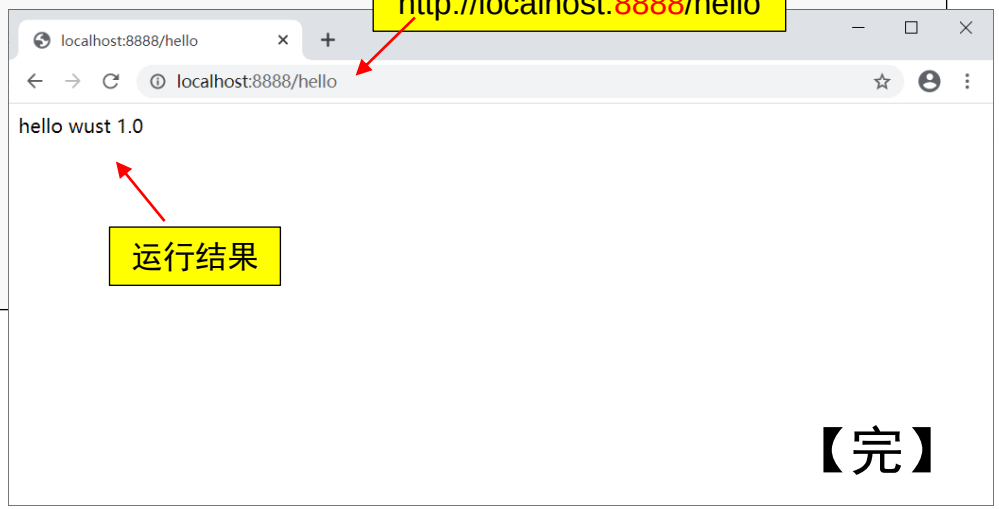

表示注释
#服务端口号
server.port=8888
将默认的端口号8080改为8888
#自定义配置
自定义属性（字符串值不用引号）
demo.app.name=wust
demo.app.version=1.0
demo.app.description=hello ${demo.app.name} ${demo.app.version}
在application.properties中的各个参数之间
可以直接通过 "${属性名}" 引用来使用

使用自定义属性

```
@RestController
public class HelloController {
    @Value("${demo.app.description}")
    private String desc;

    @RequestMapping("/hello")
    public String index(){
        return desc;
    }
}
```

@Value 注解给变量赋值：@Value("${属性名}")

端口号已改变



【完】