06_初来乍到：先基于主流 web 技术 Spring MVC 进行项目初始化

1、开篇

经过第一周的讲解，让我们对学习的内容有了基本的了解。其中需要依托“互联网教育系统”作为 RocketMQ 的学习案例，同时在过去一周的课程中也对案例项目进行了完整的介绍。从功能到业务逻辑，从业务逻辑到逻辑架构，又从逻辑架构到系统架构。在做足这些准备以后，今天就开始该项目的开发了。千里之行始于足下，这节课的主题是通过主流的 Spring MVC 对项目进行初始化。并且从这一节开始，会涉及到代码的部分，因此我们会将每次生成的代码都附在每节课程结束的位置给大家参考。将今天的课程分为一下几个部分：

- 创建项目
- 引入项目依赖
- 编写配置文件
- 创建启动类
- 创建 HealthController 的接口
- 运行项目

2、创建项目

我们通过 IntelliJ IDEA 创建项目，如图 1 所示，打开 IntelliJ IDEA，选择“Create New Project” 创建项目。

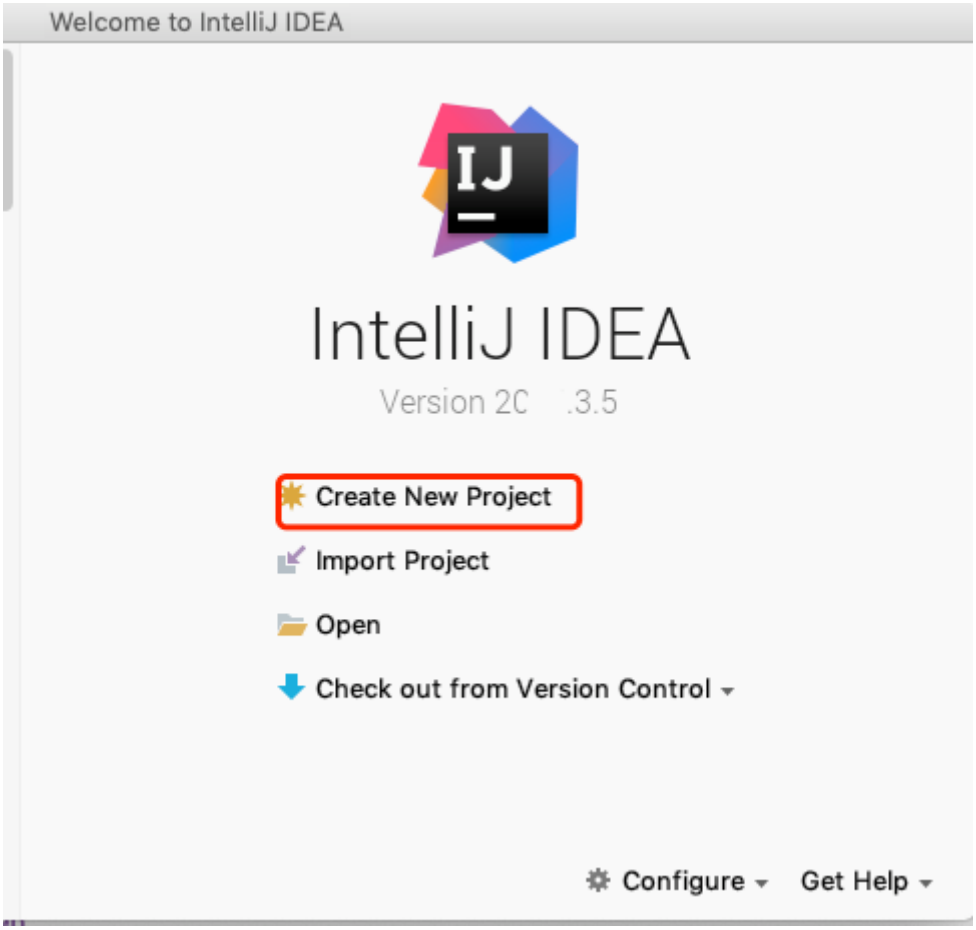


图 1 创建项目

如图 2 所示，在弹出的选框中，选择“Maven”表示通过 Maven 来创建项目，然后点击“Next”按钮。

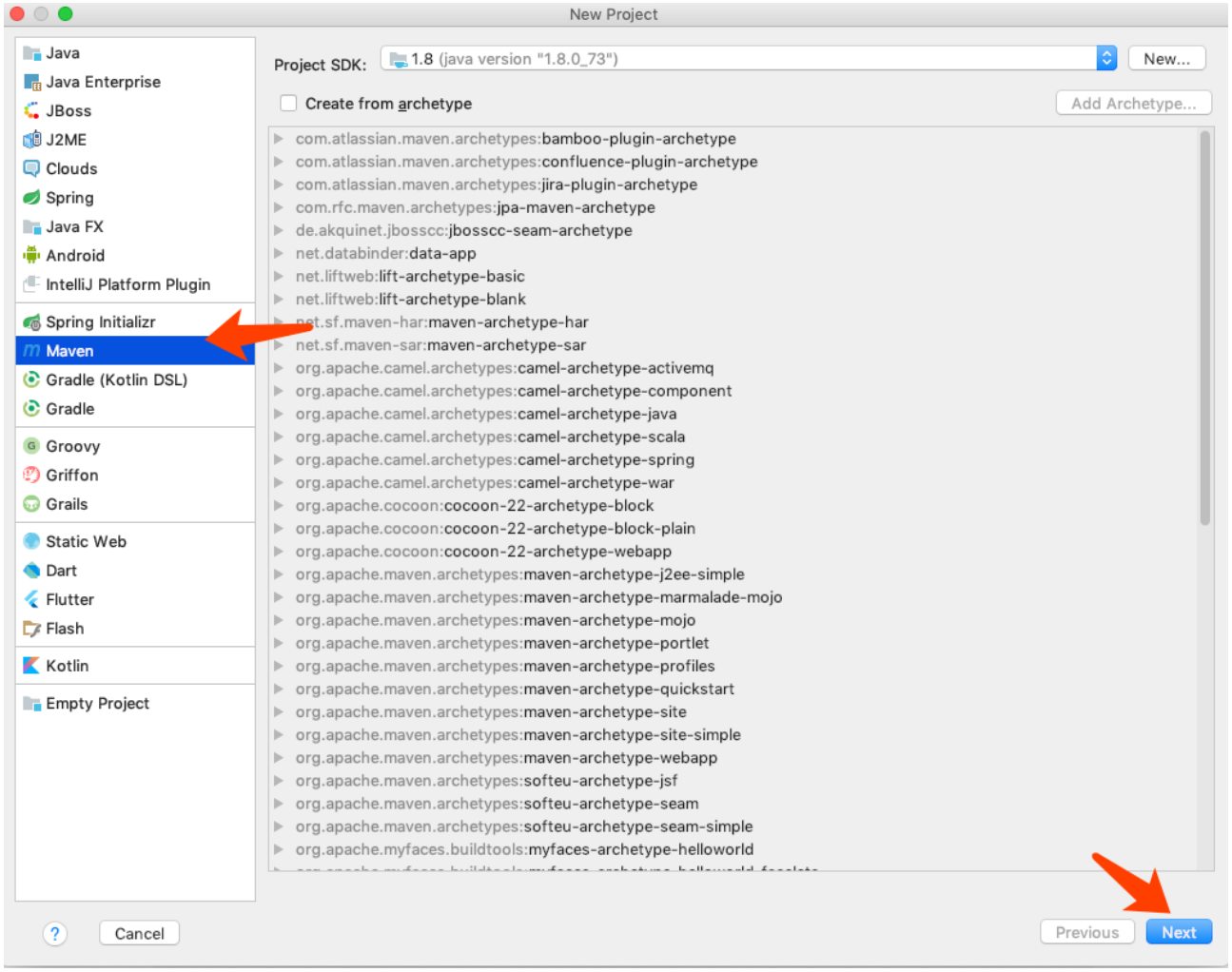


图 2 选择 Maven 创建项目

如图 3 所示在弹框中输入“GroupId”和“ArtifactId”的信息。分别是“com.ruyuan2020.little.project”和“little-project-spring”，然后点击“Next”进入下一步。

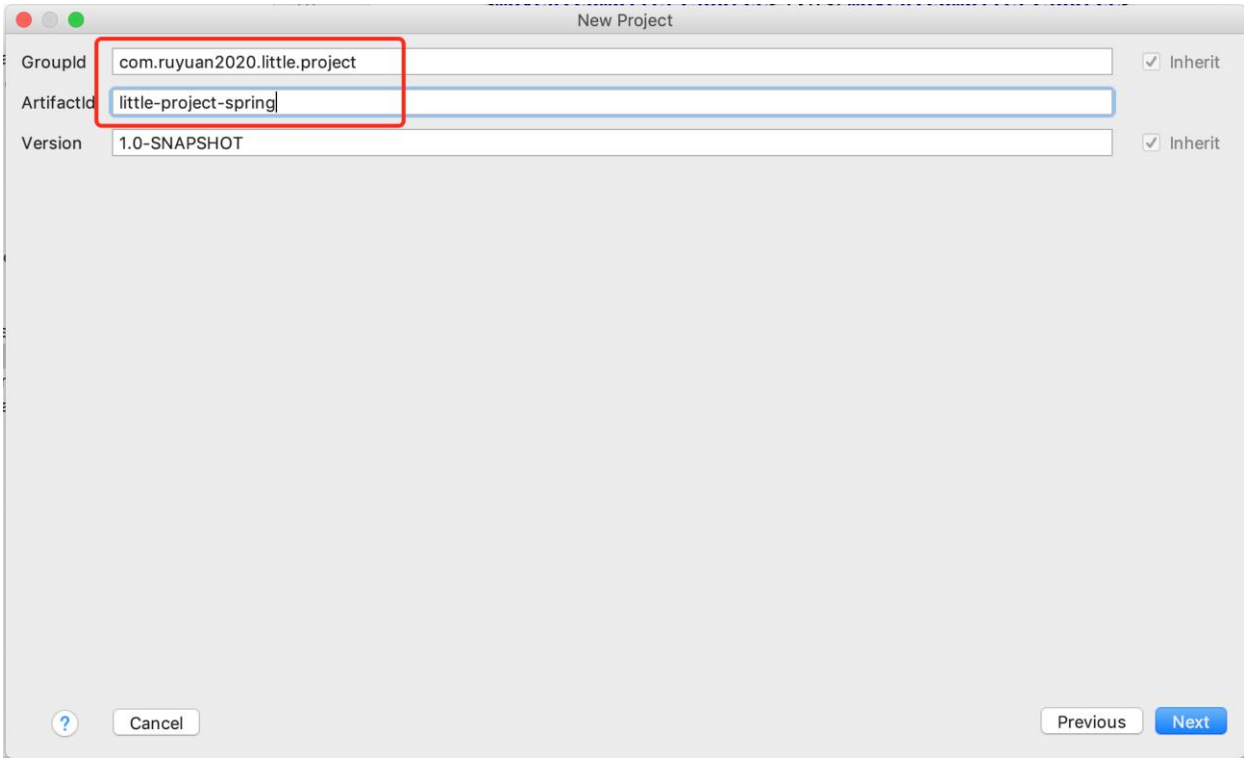


图 3 填写 GroupId 和 ArtifactId 的信息

如图 4 所示，在弹框中填写“Project name”为“little-project-spring”，然后点击“Finish”按钮表示完成项目创建。

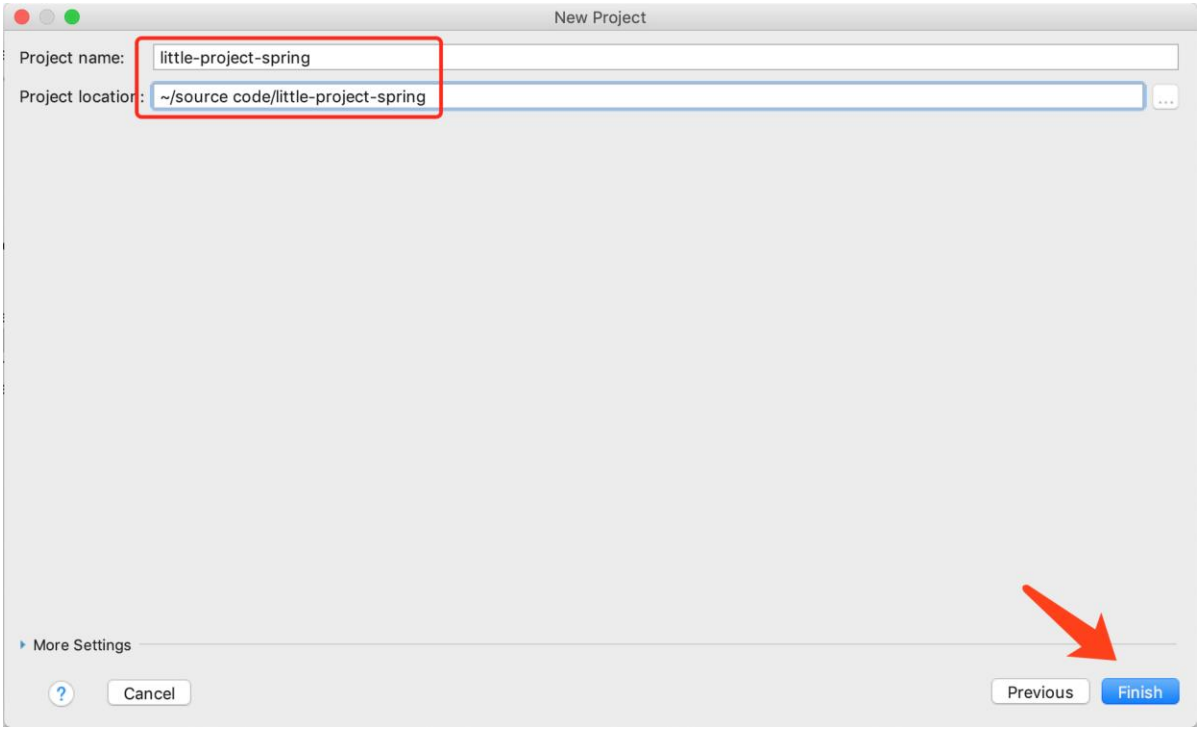


图 4 填写 Project name 完成项目创建

3、引入项目依赖

如图 5 所示，首先来看看引入依赖后的 pom.xml 文件，里面包含了很多组件的依赖项。相信在之前的小实战的课程中大家已经有过接触，这里选择几个儒猿小实战云平台提供的依赖给大家做介绍。

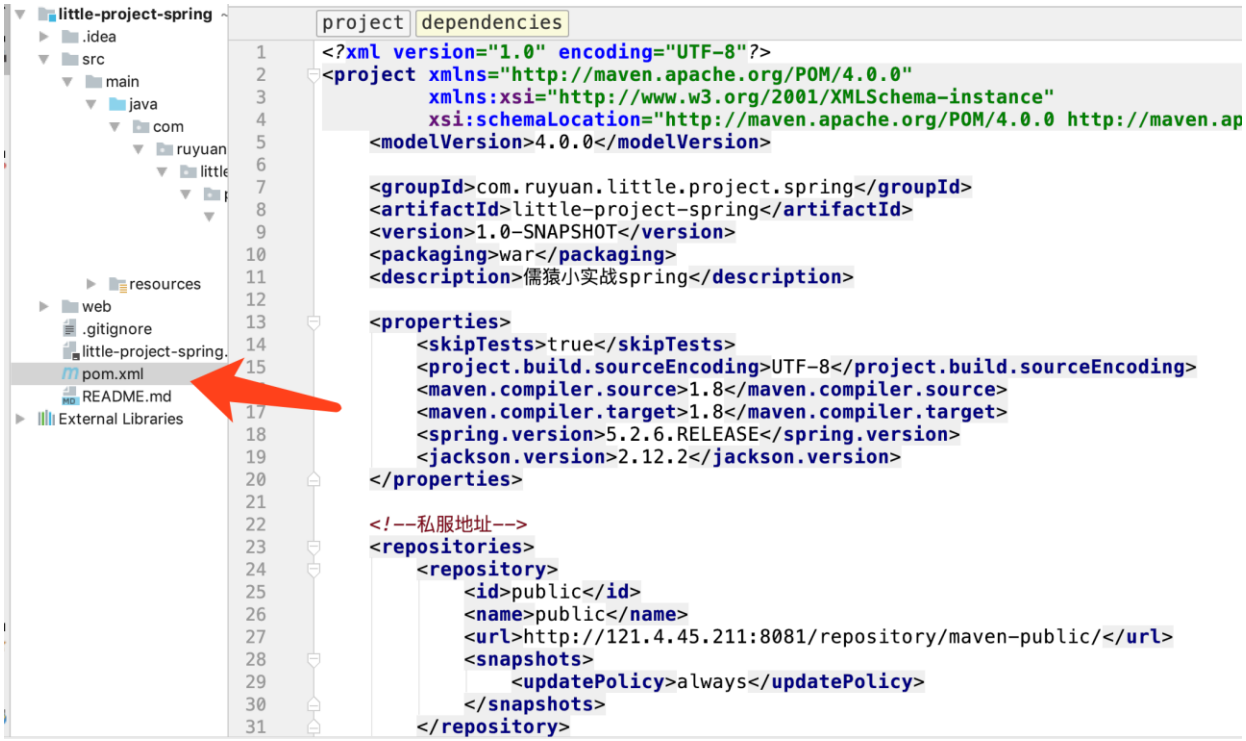


图 5 项目依赖

4、儒猿团队提供的公共依赖项

如图 6 所示，这里的小实战项目的公共依赖由儒猿团队提供的，这个依赖会贯穿整个项目，因此需要在此处引入，从图 6 中可以看出目前 little-project-common 的版本是 1.0.1-RELEASE。

```
<!-- 小实战项目的公共依赖 -->
<dependency>
    <groupId>com.ruyuan2020.little.project</groupId>
    <artifactId>little-project-common</artifactId>
    <version>1.0.1-RELEASE</version>
</dependency>
```

图 6 小实战项目的公共依赖

需要注意的是由于该组件出自儒猿团队，因此需要通过配置 Maven 仓库对应的私服地址才能够获取，我们把地址的配置放到下面，以供参考。

```
<!--私服地址-->
    <repositories>
        <repository>
            <id>public</id>
            <name>public</name>
            <url>http://121.4.45.211:8081/repository/maven-public/</url>
            <snapshots>
                <updatePolicy>always</updatePolicy>
            </snapshots>
        </repository>
    </repositories>
```

5、Spring 相关依赖项

如图 7 所示，作为本节课的主角，需要引入 Spring 相关的依赖项。包括 spring-context、spring-core、spring-web 以及 Spring-webmvc。

```
<!-- spring 相关依赖 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${spring.version}</version>
</dependency>
```

图 7 Spring 相关的依赖项

6、JSON 依赖项

如图 8 所示，由于接口返回 Object 时需要依赖 Jackson，因此这里需要引入 JSON 依赖项。

```
<!-- json依赖 -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>${jackson.version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>${jackson.version}</version>
</dependency>
```

图 8 JSON 依赖项

7、资源文件配置与 war 包插件

如图 9 所示，在 build 节点上的 resources/resource 节点中对资源目录进行配置。在 direcotry 节点中配置了资源文件目录，同时在 includes 节点中制定扫描“*.properties”和”*.xml”文件。另外在 plugins/plugin 节点中配置了 war 包的打包插件，会使用到 web.xml 的配置。

```
<build>
    <resources>
        <resource>
            <!--所在的目录-->
            <directory>src/main/resources</directory>
            <!--包括目录下的.properties,.xml文件都会扫描到-->
            <includes>
                <include>**/*.properties</include>
                <include>**/*.xml</include>
            </includes>
            <filtering>false</filtering>
        </resource>
    </resources>

    <!--打war包的插件-->
    <finalName>little-project-spring</finalName>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.2</version>
            <configuration>
                <webXml>web\WEB-INF\web.xml</webXml>
            </configuration>
        </plugin>
    </plugins>

</build>
```

图 9 应用程序包启动项的入口

8、编写配置文件

说完了依赖项，再来看看配置文件。如图 10 所示，在 resources 目录下面有一个 spring-web.xml 的配置文件。其中会制定扫描 web 相关的 bean，这里通过 base-package 指定包名：“com.ruyuan.little.project.spring”。同时在 annotaion-driven

节点中的 message-converters 节点下面定义了 Json 格式到 httpMessage 格式的转换 bean，也就是说通过它们之间的转换可以通过“org.springframework.http.converter.json.MappingJackson2HttpMessageConverter”类完成。

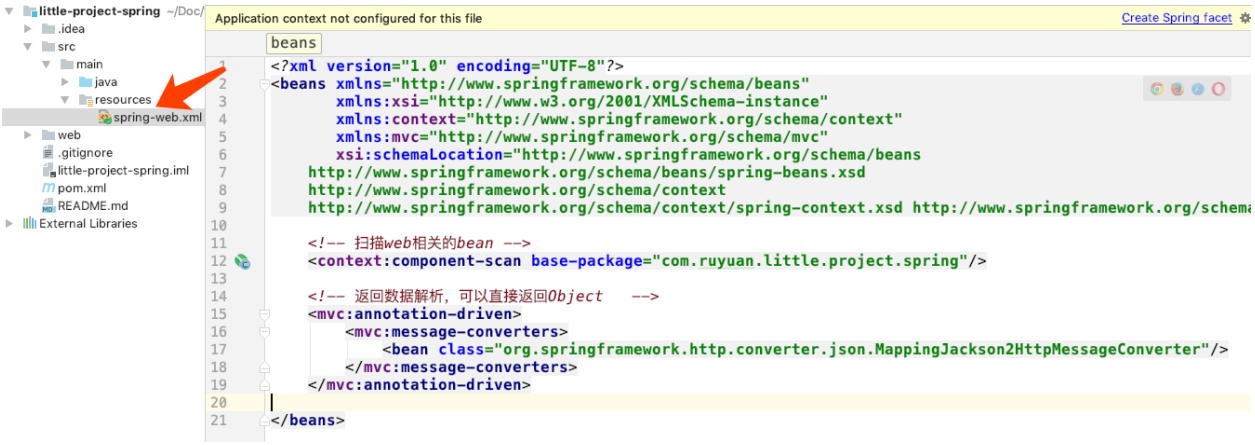


图 10 配置文件

9、创建 HealthController 的接口

整个项目会以服务的形式暴露给前端应用调用，为了检查服务的健康状况，我们建立一个 HealthController。如图 11 所示，定义 HealthController 类加上“@RestController”的 annotation。在访问服务的根目录且服务运行正常时会返回“成功”，否则就视为服务不可用。

图 11 HealthController

10、运行项目

完成以上步骤以后尝试运行项目，不过在运行项目之前我们需要尝试配置 tomcat。如图 12 所示，进入 IntelliJ 的“run“-”Edit Configurations”菜单。

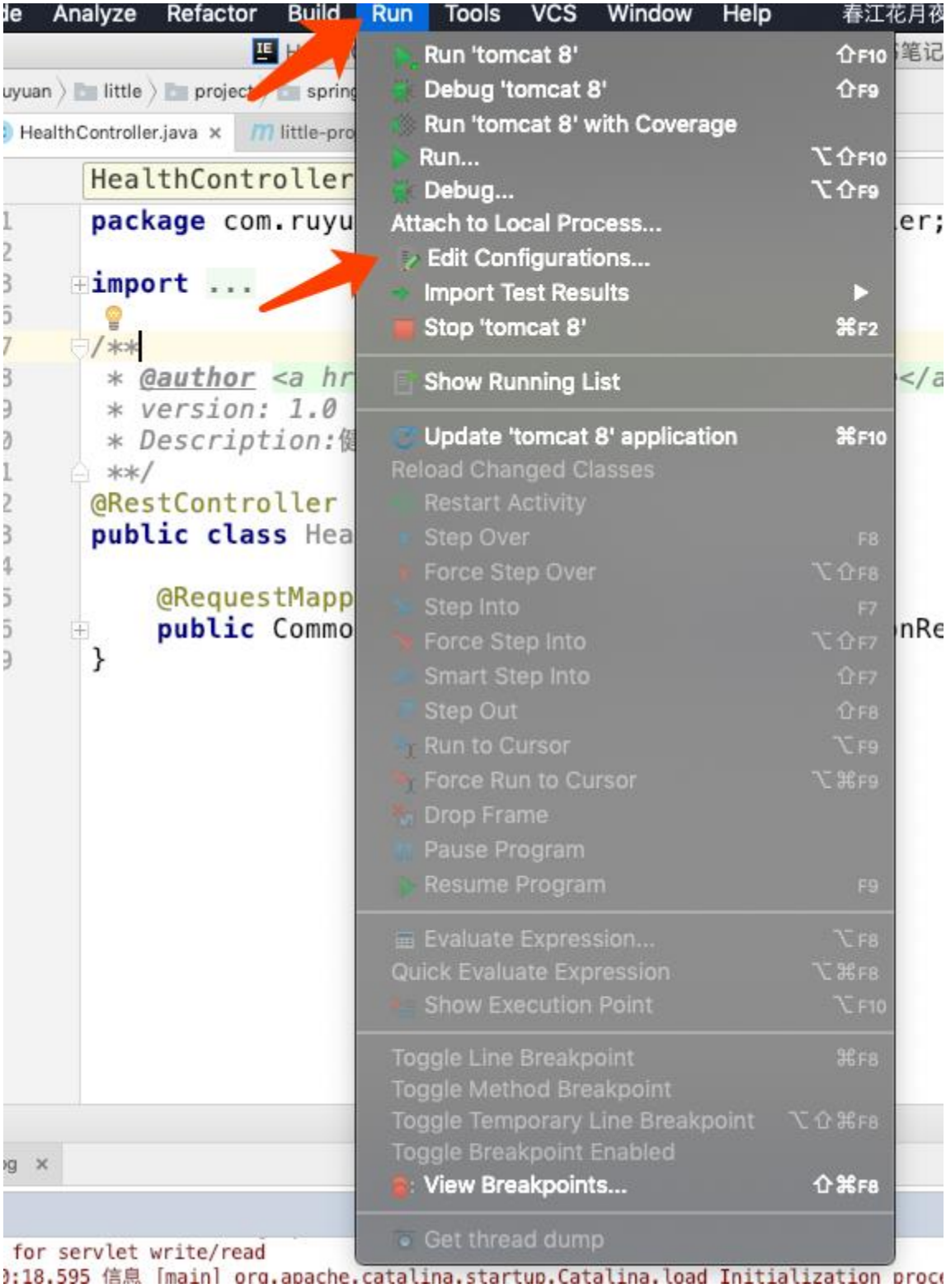
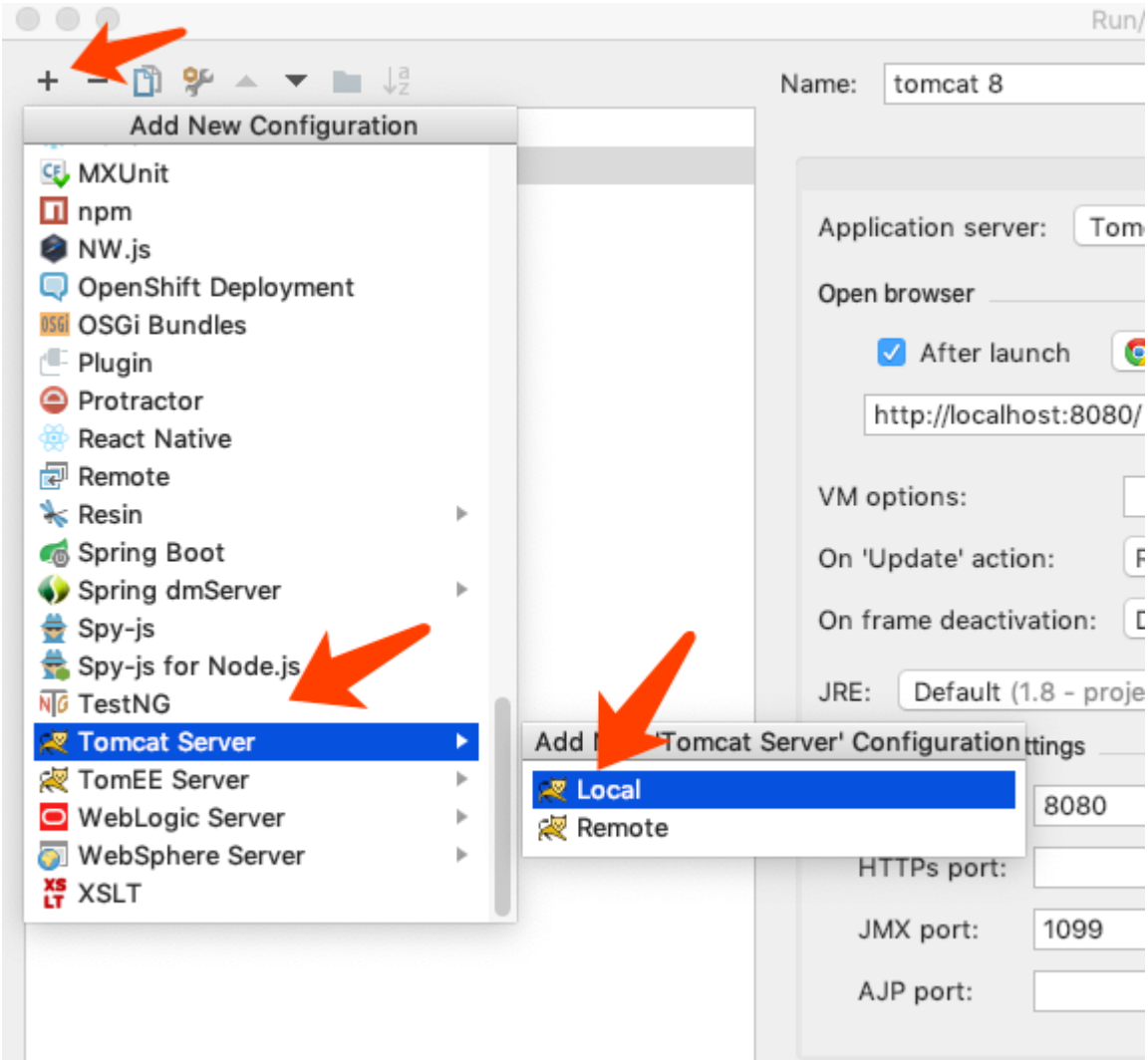


图 12 进入 Edit Configuration 菜单

如图 13 所示，在弹出框中选择左上角的“+”，在“Add New Configuration”菜单中选择“Tomcat Server”的“Local”。



如图 13 添加 tomcat 配置

如图 14 在新增的 tomcat 服务器配置中，给服务器起一个名字，由于我这里使用的是“tomcat 8”，让后通过“Configuration”按钮选择 Tomcat 所在的目录。接下来就是配置 Tomcat 的启动地址，我们使用默认的“http：//localhost：8090”。

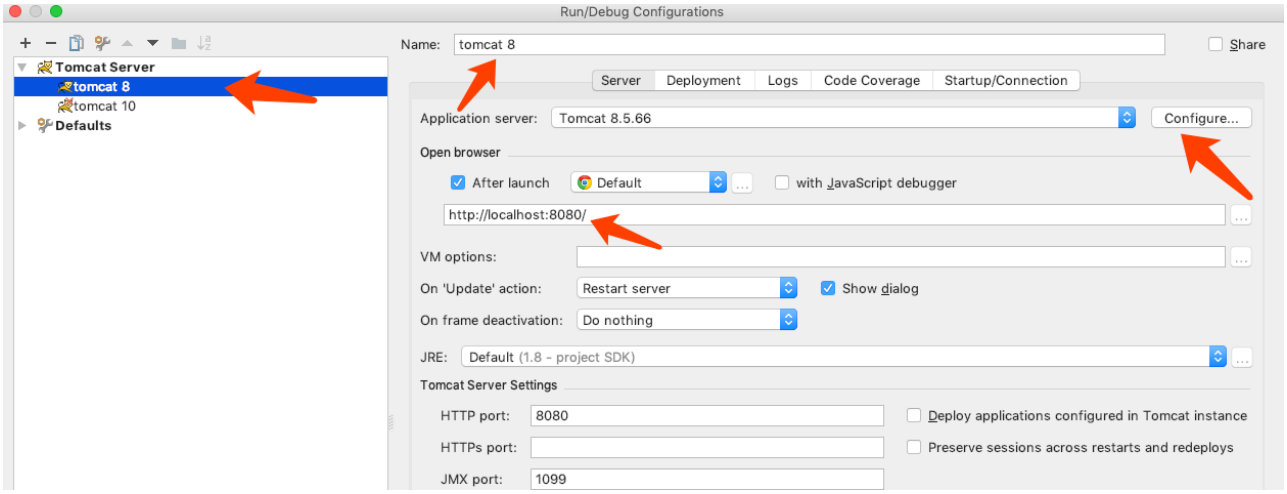


图 14 填写 Tomcat 基本信息

如图 15 所示在 Tomcat 配置窗口中点击“Deployment”按钮，选择左下角的“+”号，然后点击 Aritifact 用来设置 Tomcat 与项目之间的关联。

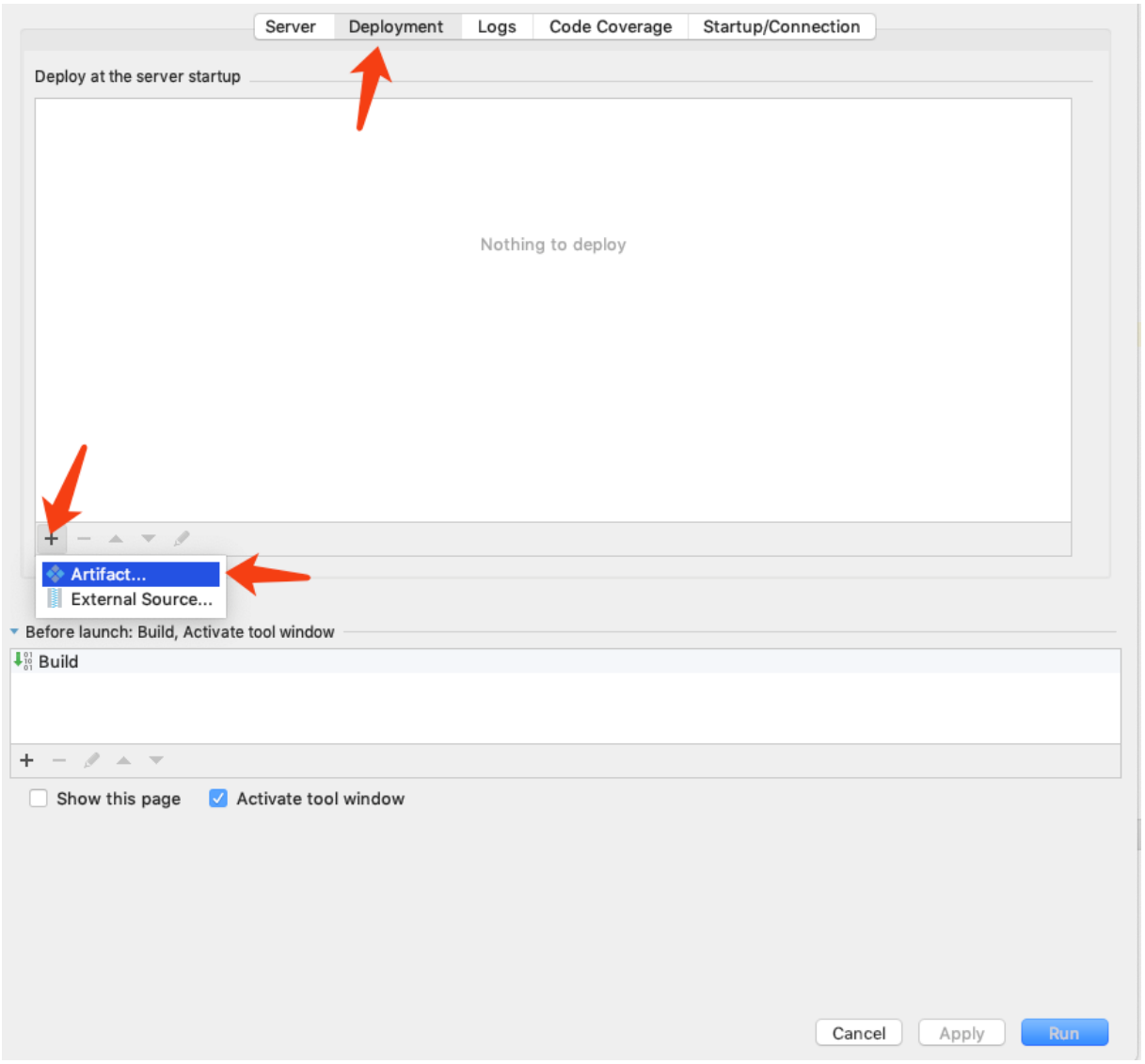


图 15 设置 Deployment 中的 Artifact

如图 16， 在弹出的窗体中选择”little-project-spring:war exploded” 然后点击“OK”。

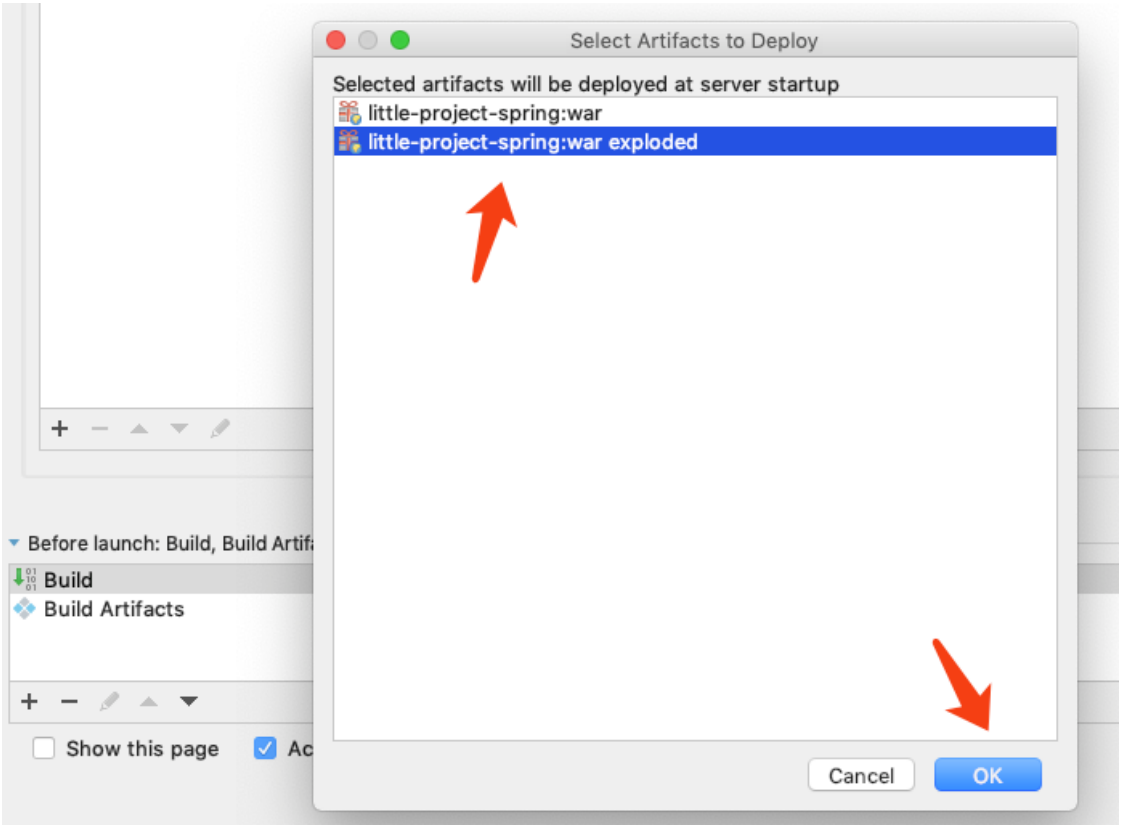


图 16 选择 little-project-spring:war exploded

完成上述配置以后，就可以通过 Tomcat 启动应用了。如图 17 所示，通过点击“Run tomcat 8”，启动配置好的 Tomcat 服务器，并且运行我们的项目。

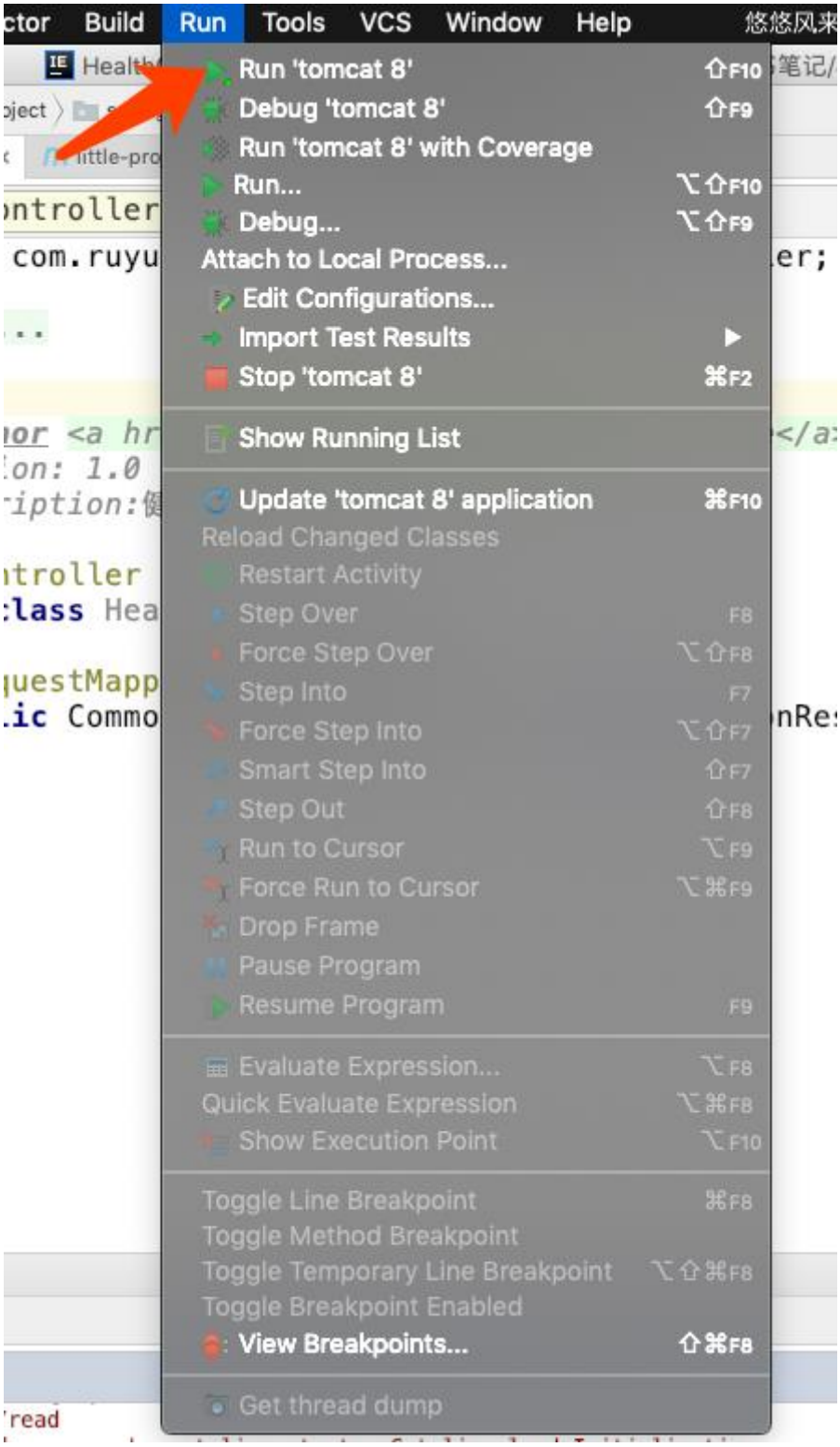

ctor
Build
Run
Tools
VCS
Window
Help
悠悠风来
Run 'tomcat 8' ⇧F10
Debug 'tomcat 8' ⇧F9
Run 'tomcat 8' with Coverage
Run... ⌥⇧F10
Debug... ⌥⇧F9
Attach to Local Process...
Edit Configurations...
Import Test Results
Stop 'tomcat 8' ⌘F2
Show Running List
Update 'tomcat 8' application ⌘F10
Reload Changed Classes
Restart Activity
Step Over F8
Force Step Over ⌥⇧F8
Step Into F7
Force Step Into ⌥⇧F7
Smart Step Into ⇧F7
Step Out ⇧F8
Run to Cursor ⌥F9
Force Run to Cursor ⌥⌘F9
Drop Frame
Pause Program
Resume Program F9
Evaluate Expression... ⌥F8
Quick Evaluate Expression ⌥⌘F8
Show Execution Point ⌥F10
Toggle Line Breakpoint ⌘F8
Toggle Method Breakpoint
Toggle Temporary Line Breakpoint ⌥⇧⌘F8
Toggle Breakpoint Enabled
View Breakpoints... ⇧⌘F8
Get thread dump

图 17 启动项目

同时打开浏览器，输入“http://localhost:8090”查看结果，此时请求会路由到 HealthController 中检查服务的健康状况。看到浏览器返回“{"code":200,"message":"成功","data":null}” 表示项目构建成功了。

11、总结

这一节作为实践的第一次课，带大家创建了项目框架。其中包括了几个部分：填写基本信息创建项目、设置项目的依赖组件、生成配置文件、创建 HealthController 检查服务健康以及运行项目。大家可以跟着每步的图文提示走一遍，我们把最终的代码放在最后供大家参考。下节课，会介绍基本服务以及对应的功能模块，这里将代码给大家，下期见，拜拜。

友情提示：本章讲述代码只是部分核心代码，完整代码请查阅文末链接中代码，谢谢。

注意事项：maven 依赖下载问题

1、注释本地的 maven setting 中的 repository 和 mirror 配置

2、删除自己 maven setting 中配置本地仓库地址下 ruyuan2020 的包（可能第一次拉了 lastupdated 的包）

3、在 idea 中点击 maven 依赖刷新按钮