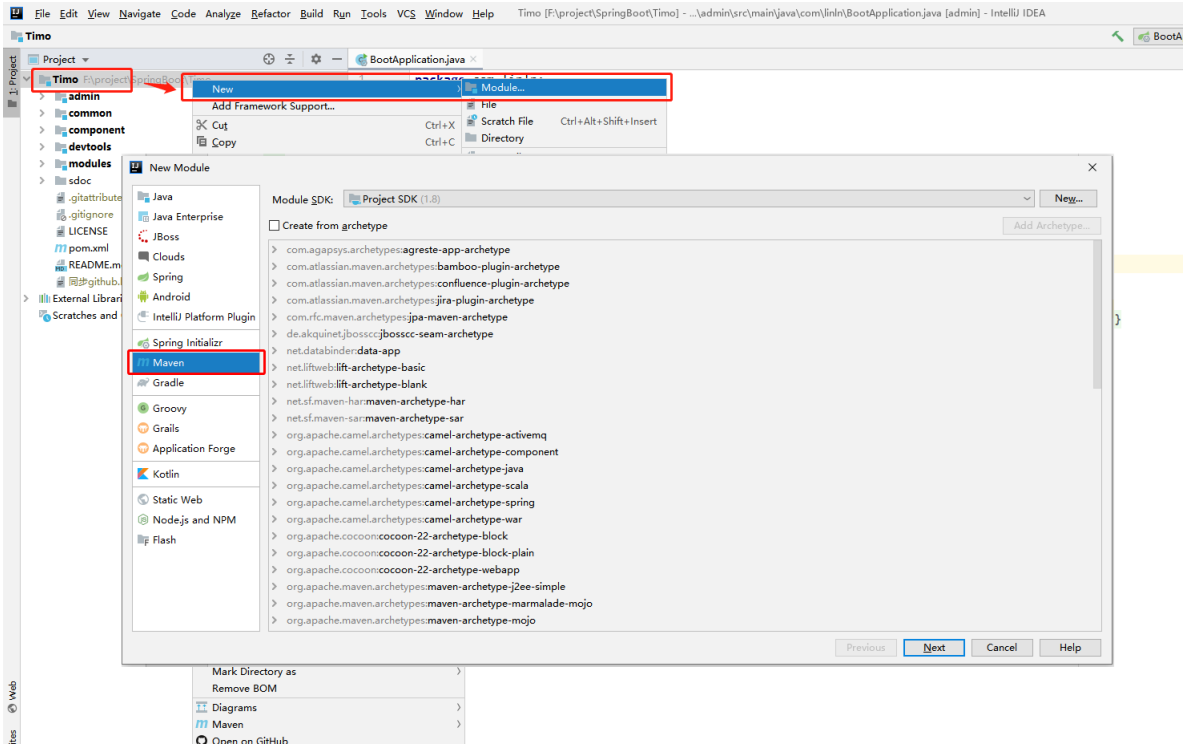
Timo创建前台API模块!

1.创建maven模块

以idea为例，在根目录下创建maven模块，如图：



2.修改API模块pom文件

引入公共模块和jwt模块即可，修改完成后在admin模块中引入新模块即可!

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>timo</artifactId>
        <groupId>com.linln</groupId>
        <version>2.0.3</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>webapi</artifactId>
    <packaging>jar</packaging>
    <name>API模块</name>

    <!-- 修改部分 -->
    <dependencies>
        <!--引入公共模块-->
        <dependency>
            <groupId>com.linln</groupId>
            <artifactId>common</artifactId>
            <version>${project.version}</version>
        </dependency>
```

```
        <!--引入jwt组件-->
        <dependency>
            <groupId>com.linln.component</groupId>
            <artifactId>jwt</artifactId>
            <version>${project.version}</version>
        </dependency>
    </dependencies>

</project>
```

3.API模块作为独立的项目

如果不想和后台模块耦合在一起，也可以将API模块作为一个可运行的独立项目！

1. 修改API模块pom文件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>timo</artifactId>
        <groupId>com.linln</groupId>
        <version>2.0.3</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>webapi</artifactId>
    <packaging>jar</packaging>
    <name>API模块</name>

    <!-- 修改部分 -->
    <dependencies>
        <!--引入公共模板-->
        <dependency>
            <groupId>com.linln</groupId>
            <artifactId>common</artifactId>
            <version>${project.version}</version>
        </dependency>
        <!--引入jwt组件-->
        <dependency>
            <groupId>com.linln.component</groupId>
            <artifactId>jwt</artifactId>
            <version>${project.version}</version>
        </dependency>
    </dependencies>

    <!-- 引入springboot插件，用于运行和打包 -->
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

```
</project>
```

2. 加入启动类

注意：启动类需要扫描到其他模块下的包路径，可以直接将启动类放在`com.linln`路径下，或者在`@SpringBootApplication`注解中加入`scanBasePackages = {"com.linln"}`扫描参数！

```
package com.linln;

@SpringBootApplication(scanBasePackages = {"com.linln"})
@EnableJpaAuditing // 使用jpa自动赋值
@EnableCaching // 开启缓存
public class ApiApplication extends SpringBootServletInitializer {

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

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder
application) {
        return application.sources(ApiApplication.class);
    }
}
```

3.加入配置文件

直接复制`admin模块`下的`application.yml`和`ehcache.xml`到API模块的资源目录下即可！

可优化`application.yml`文件，去除部分无关的配置。