

步骤：

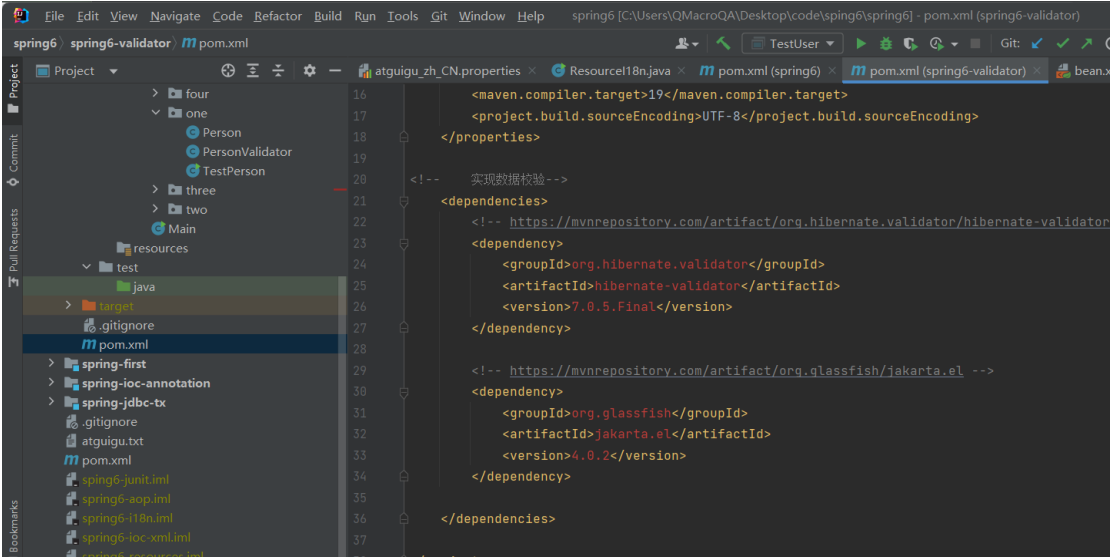
第一步 引入依赖

第二步 创建实体类，定义属性，  
创建对应set和get方法

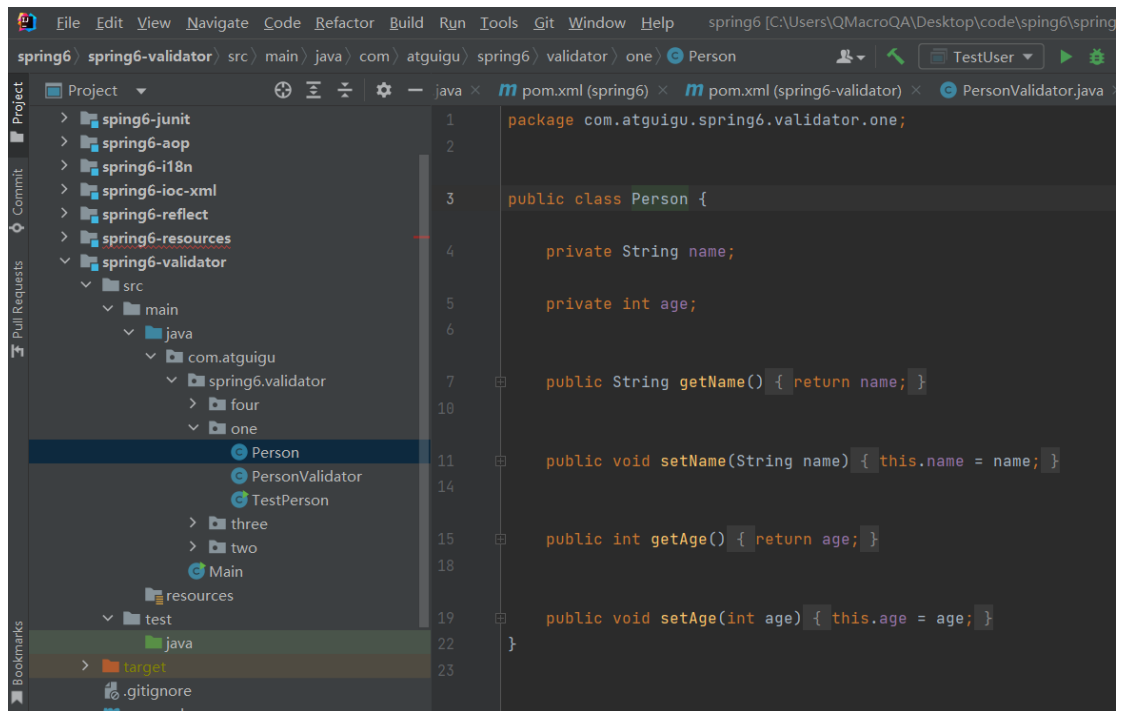
第三步 创建类，实现接口，实现接口  
的方法，编写校验逻辑

第四步 最终测试

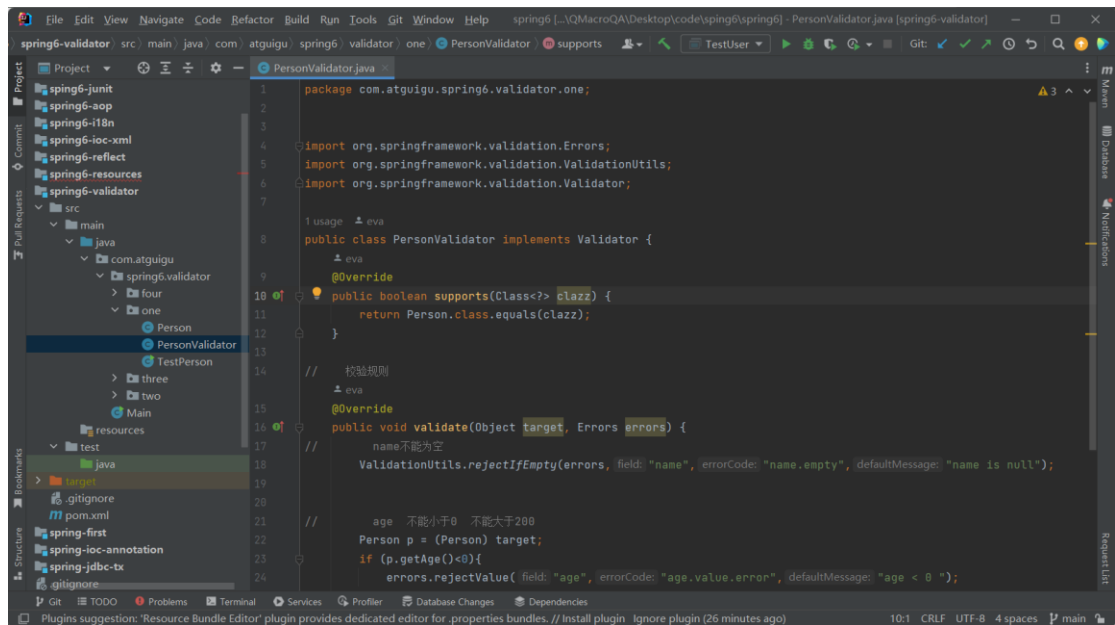
环境准备：



```
16 <maven.compiler.target>19</maven.compiler.target>
17 </maven.compiler.target>
18 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
19 </project.build.sourceEncoding>
20 </properties>
21 <!-- 实现数据校验 -->
22 <dependencies>
23 <!-- https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-validator -->
24 <dependency>
25 <groupId>org.hibernate.validator</groupId>
26 <artifactId>hibernate-validator</artifactId>
27 <version>7.0.5.Final</version>
28 </dependency>
29 <!-- https://mvnrepository.com/artifact/org.glassfish/jakarta.el -->
30 <dependency>
31 <groupId>org.glassfish</groupId>
32 <artifactId>jakarta.el</artifactId>
33 <version>4.0.2</version>
34 </dependency>
35 </dependencies>
```



校验:



```
// 校验规则
// 1. name 不能为空
// 2. age 不能小于0 不能大于200
// 3. ...

@Override
public void validate(Object target, Errors errors) {
    // 1. name 不能为空
    ValidationUtils.rejectIfEmpty(errors, field: "name", errorCode: "name.empty", defaultMessage: "name is null");

    // 2. age 不能小于0 不能大于200
    Person p = (Person) target;
    if (p.getAge() < 0) {
        errors.rejectValue(field: "age", errorCode: "age.value.error", defaultMessage: "age < 0 ");
    } else if (p.getAge() > 200) {
        errors.rejectValue(field: "age", errorCode: "age.value.error.old", defaultMessage: "age > 200 ");
    }
}
}
```

测试：

