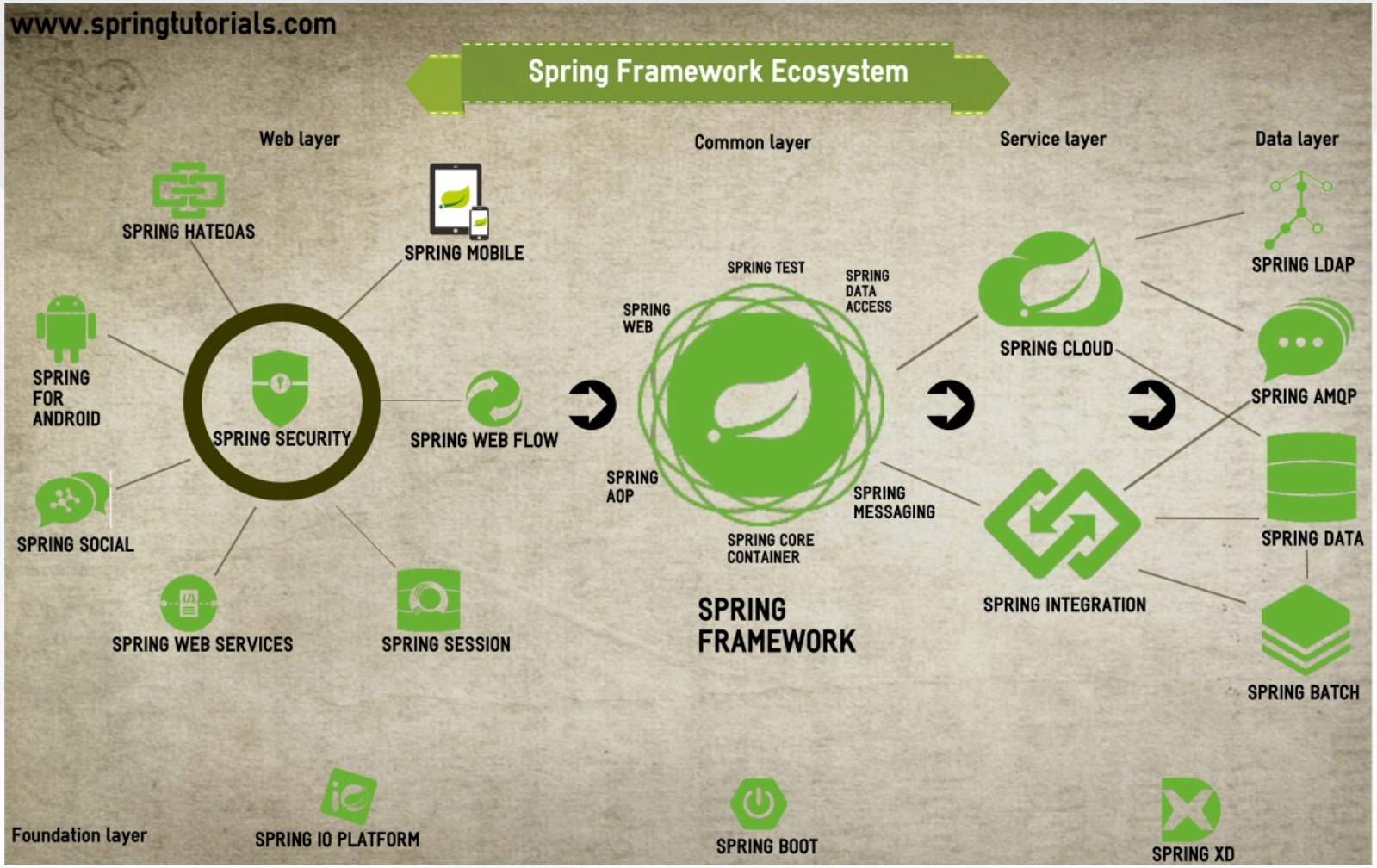
Spring

浙江大学城市学院
彭 彬
pengb@zucc.edu.cn

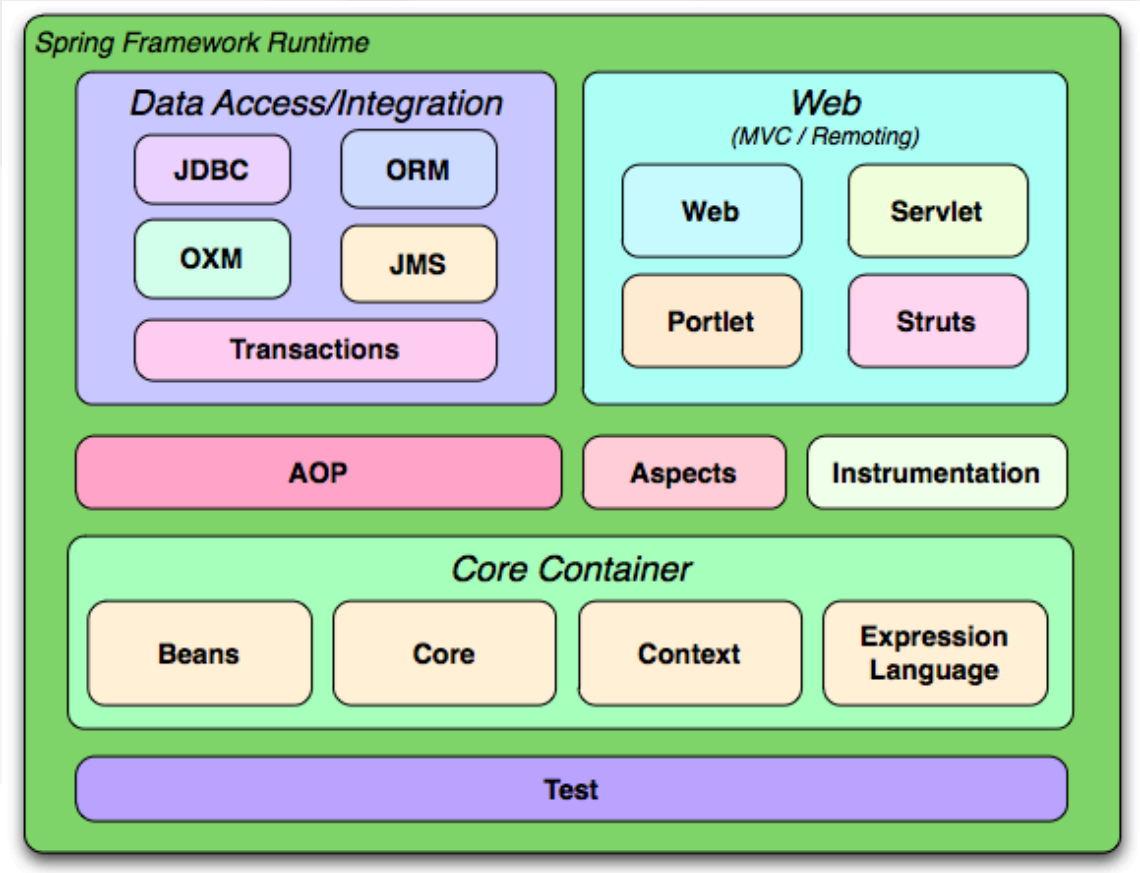
Spring生态

我们先看看Spring生态当前大概涉及的内容。Spring最大的优势是不断演进，并且提供了一站式，高质量的解决方案。“Spring走在了Java EE的前面”。



Spring Framework

Spring Framework是Spring起始的项目，也是以Java EE为基础构建的开发框架，现在仍旧是Spring应用的核心。



作为Spring学习最核心最常用的就是：

1）理解Spring基于IoC（DI）提供的容器及使用方法（大量的配置）；

2）基于Spring提供的Spring MVC框架，提供Web应用的基础（无论是RESTful设计还是页面应用）；

3）基于Spring Data框架提供的数据访问层；

Spring 的几个核心名词

POJO（ Plain Ordinary Java Object ）：POJO是一个简单的普通的Java对象，它不包含业务逻辑或持久逻辑等，但不是JavaBean、EntityBean等，不具有任何特殊角色和不继承或不实现任何其它Java框架的类或接口。

IoC（DI）：反转控制（依赖注入）

Aspect：切面；

AOP：面向切面的程序设计；

Context：上下文，可以理解为一个运行环境极其包含变量、定义等；

Bean：被Spring容器管理的，用于注入的Java对象；

MVC：Spring Web框架（Model-View-Controller设计模式）

Spring到底是什么

Spring诞生于2003年，是为了解决JavaEE标准复杂性问题而诞生的一个开发框架。Spring遵从JavaEE的标准，但是并不是一个被动实现JavaEE标准（JSR）的框架，而是精心选择了一部分JavaEE的标准，并且提供了很好的结构设计，最终完成了基于JavaEE标准，快速构建健壮的JavaEE程序目的。其核心实现的JSR包括：

- Servlet API (JSR 340)
- WebSocket API (JSR 356)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)
- JPA (JSR 338)
- JMS (JSR 914)
- as well as JTA/JCA setups for transaction coordination, if necessary.

Spring现在是什么

Spring和Java EE都在不断进化，Spring（5.0）目前运行最小需要JavaEE7.0标准（包括Servlet3.1+， JPA2.1+)的支持，并且提供了对Java8中新特性的支持。Spring最初可以视为

“有选择的实现并集成了最有用的JavaEE标准，搭建了最好的JavaEE开发框架，并通过IoC和AOP为应用服务器提供了低耦合度的应用框架。”—P.B.

现在的Spring通过多年发展，不仅包括了上面的特性还提供了更多：

1. 比如通过Spring Boot减少了非常多的配置，并且有利于CI（持续集成的开发模式）；
2. 比如通过Spring Cloud提供云式框架；
3. 比如通过Spring Security提供多样化的安全控制框架；
4. 更多可以参考官方网站https://spring.io/

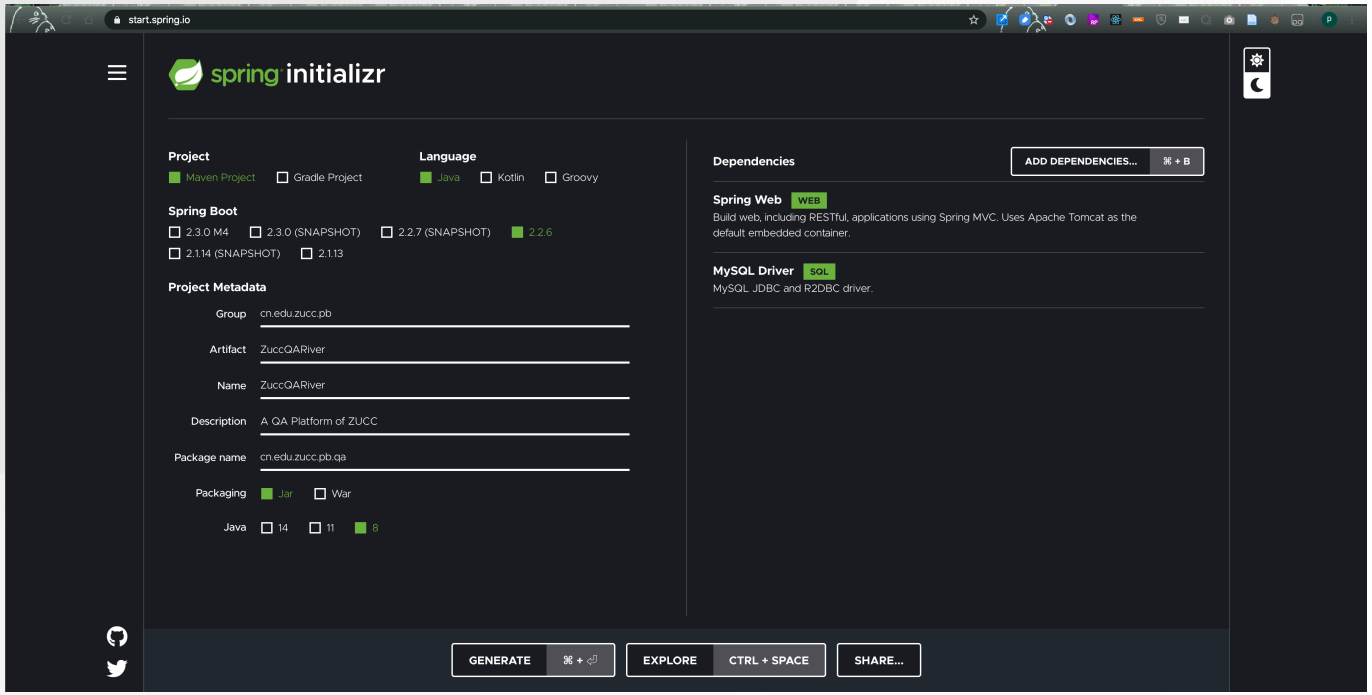
创建Spring项目

你可以用多种办法创建项目，最终只要是合适的项目结构和配置文件即可，比如常见的一些创建项目的方式：

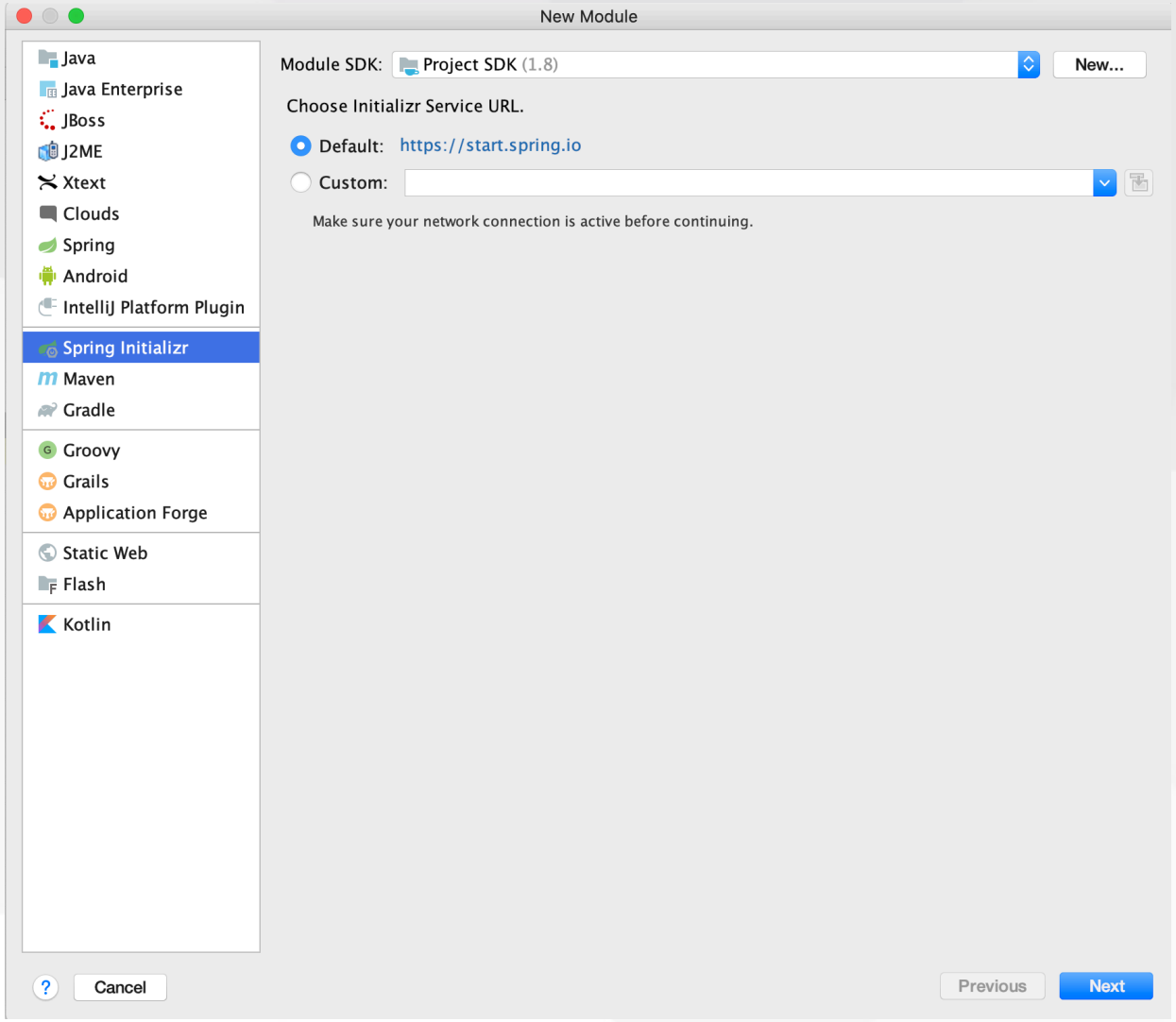
1）使用git clone复制现有项目作为项目模板来修改；

2）使用Idea的创建向导；

3）Spring项目可以通过Spring initializer初始化（这个Idea工具也支持）

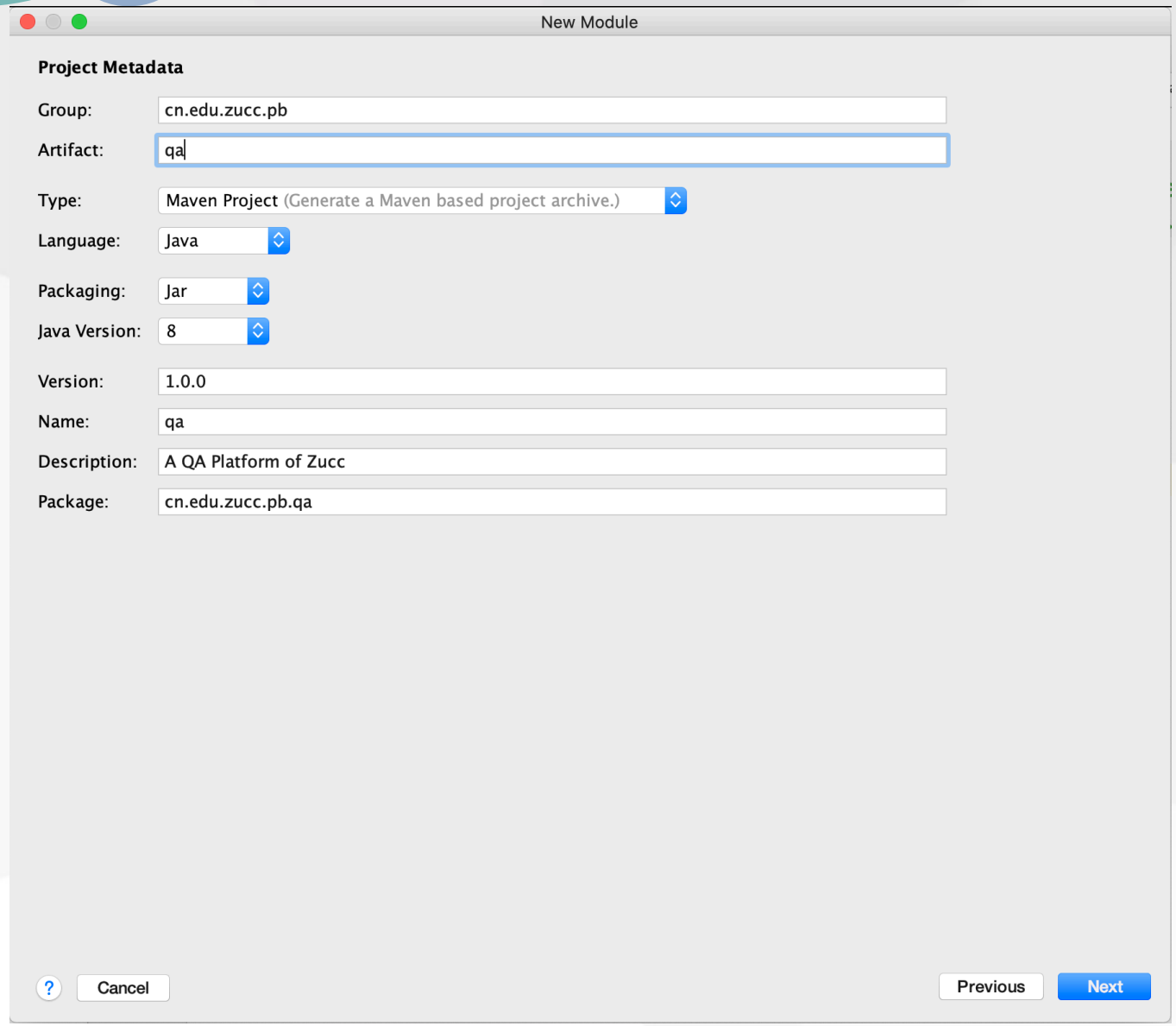


创建Spring项目



本次我们使用Idea的Spring Initializr工具进行项目初始化。

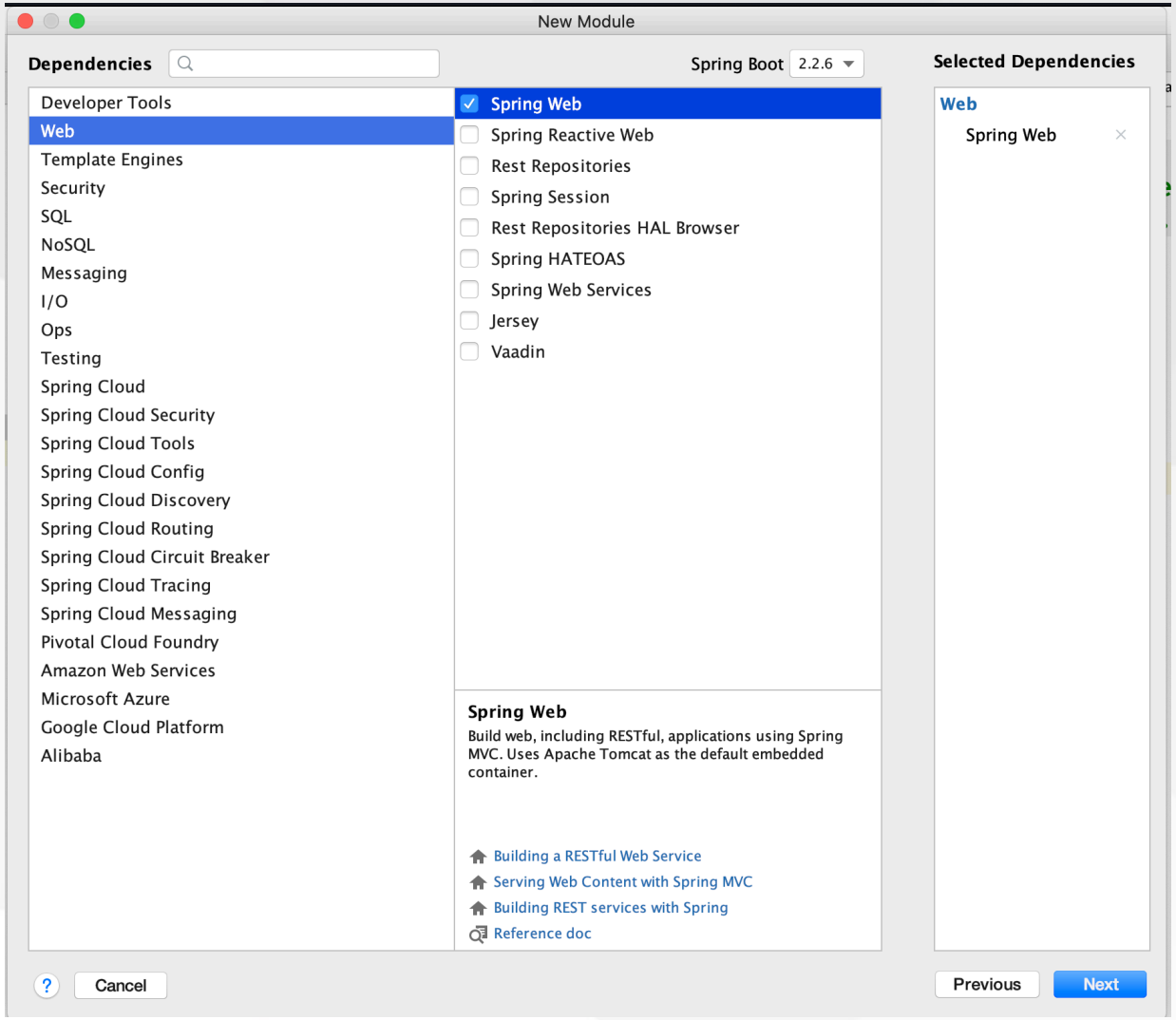
创建Spring项目



填写本次我们要完成项目的基本信息；

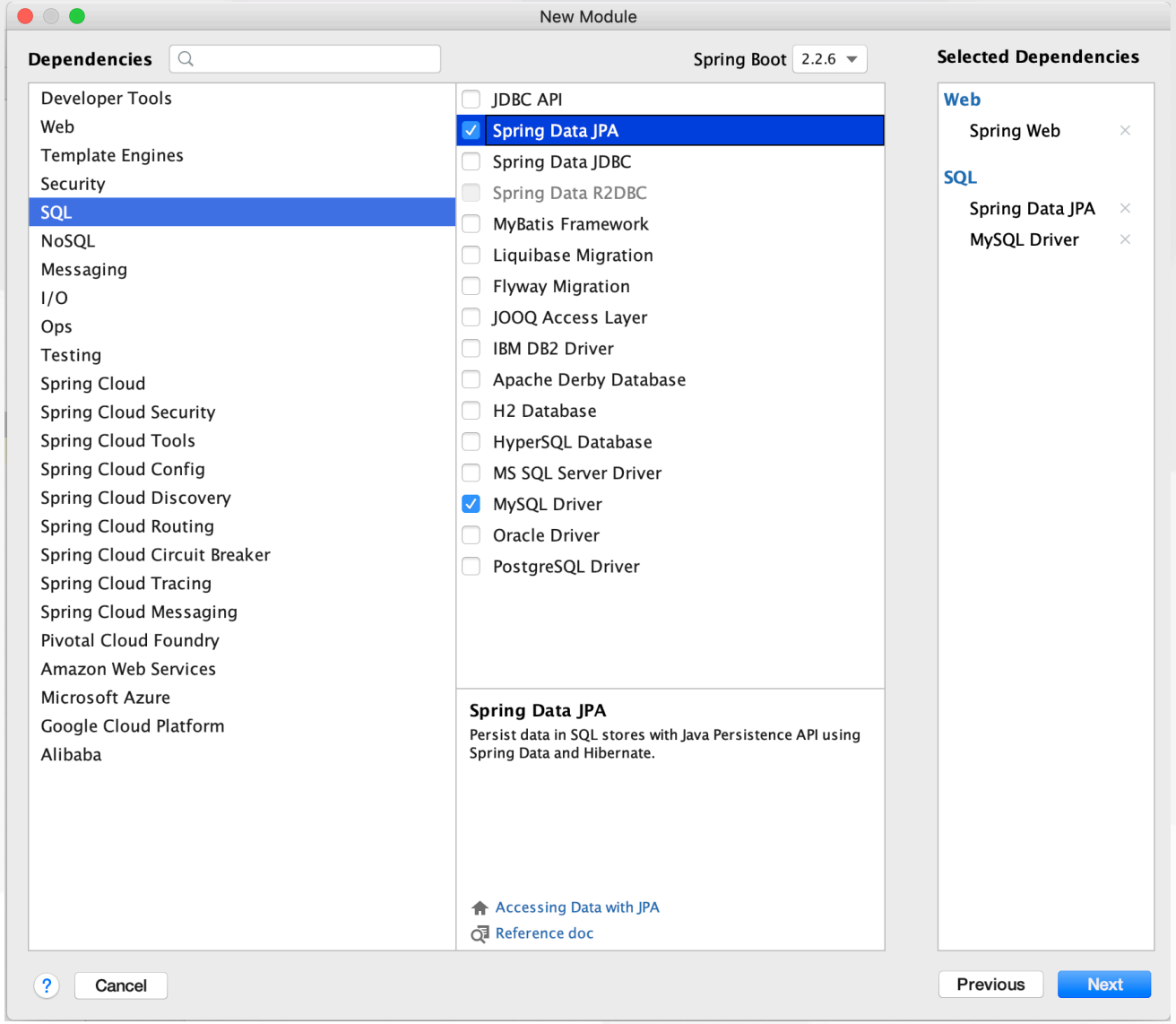
- Type是项目选择的工程管理工具（Maven或者Gradle）
- Java Version：8（免费版本）

创建Spring项目



选择Spring Web支持

创建Spring项目



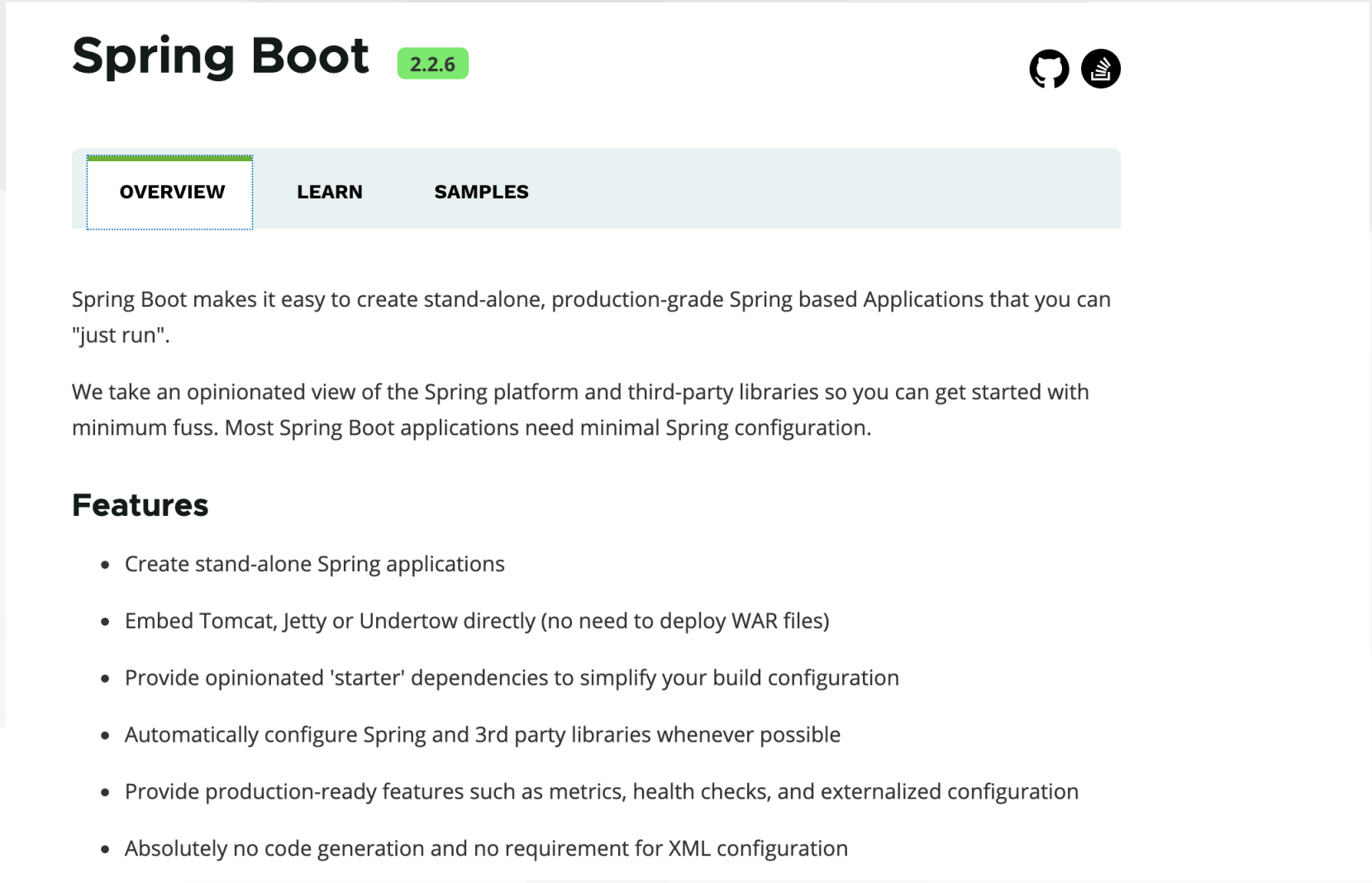
选择Spring Data支持；

选择创建工程后，Idea开始解析和下载依赖，然后根据网络情况，你就可以耐心等待了……

如果很慢请自行搜索切换到国内Maven镜像仓库。

一个Spring Boot项目

什么是Spring Boot



新创建的项目是一个Spring Boot进行支持的项目

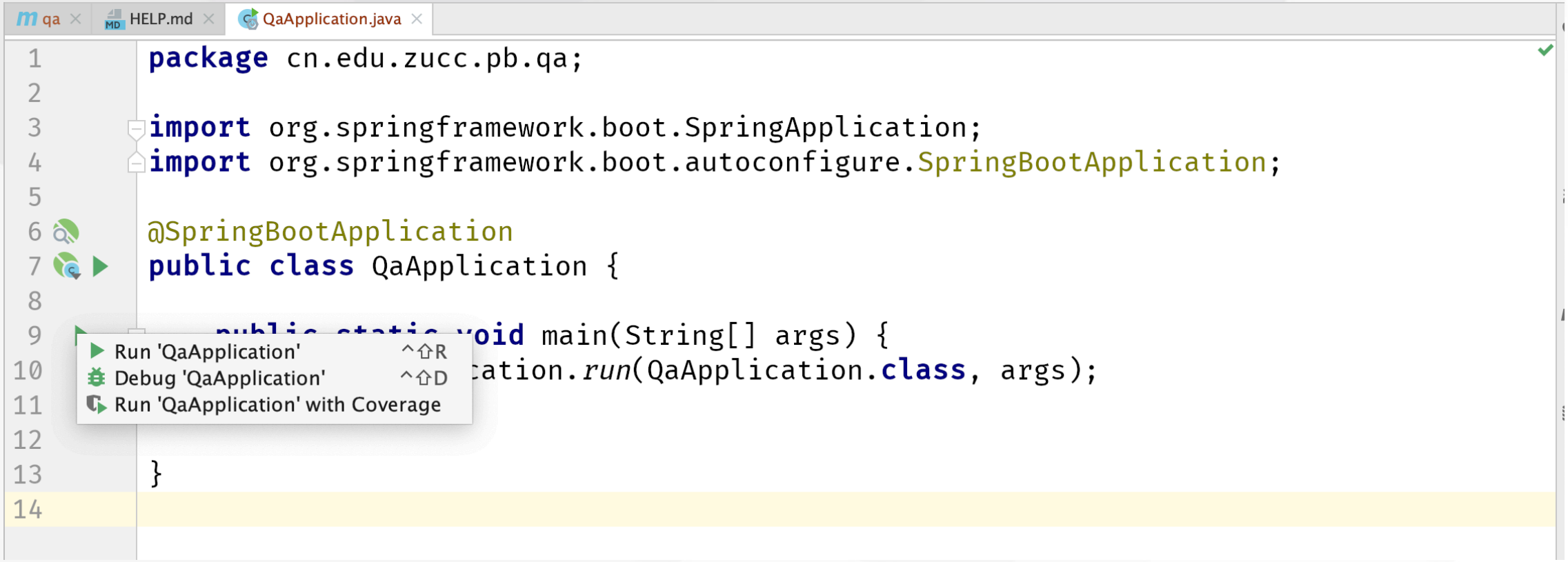
我们看见通过我们选择的想依赖，pom文件中看见了和spring boot有关的web和data两个starter

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>
```

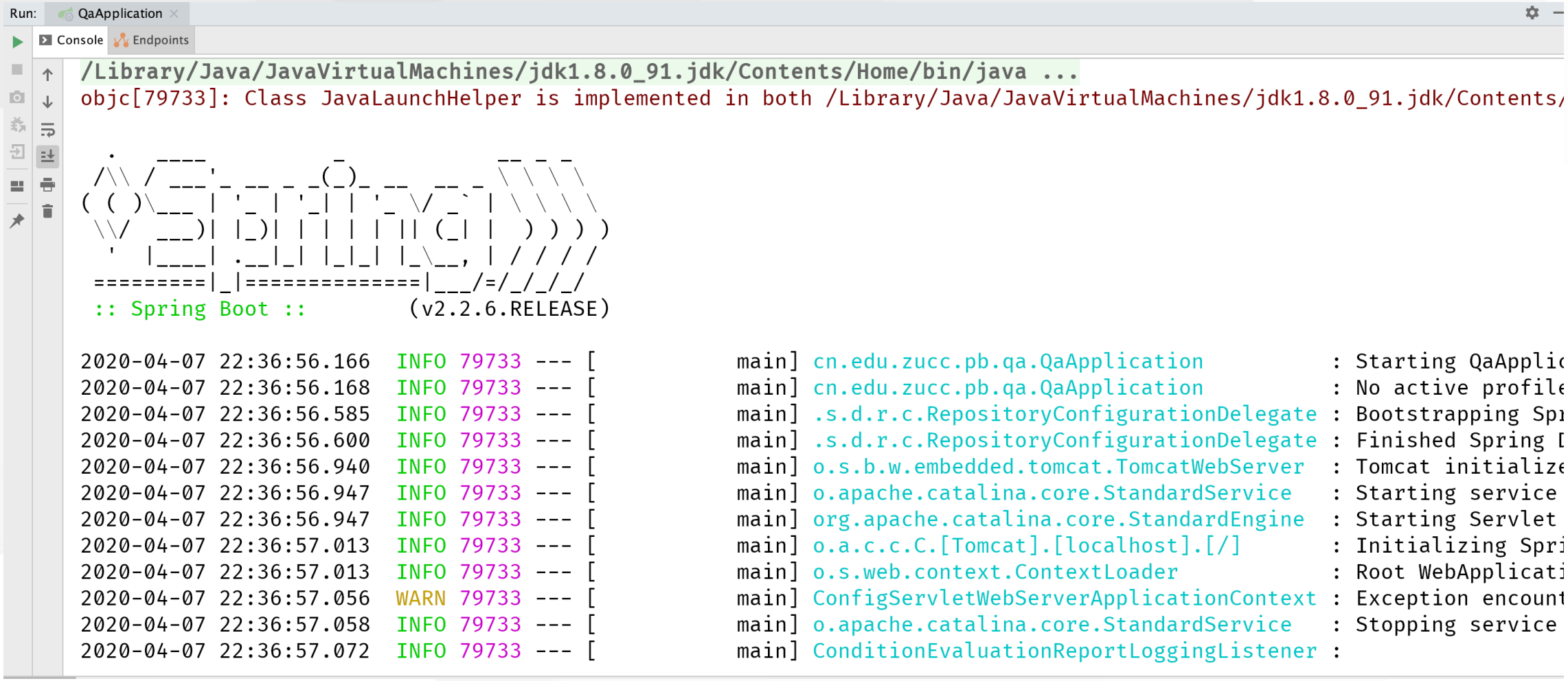
新建的项目包括了一个启动的Application

通过打开自动生成的QaApplication类，左边出现了绿色小三角，点击可以直接运行



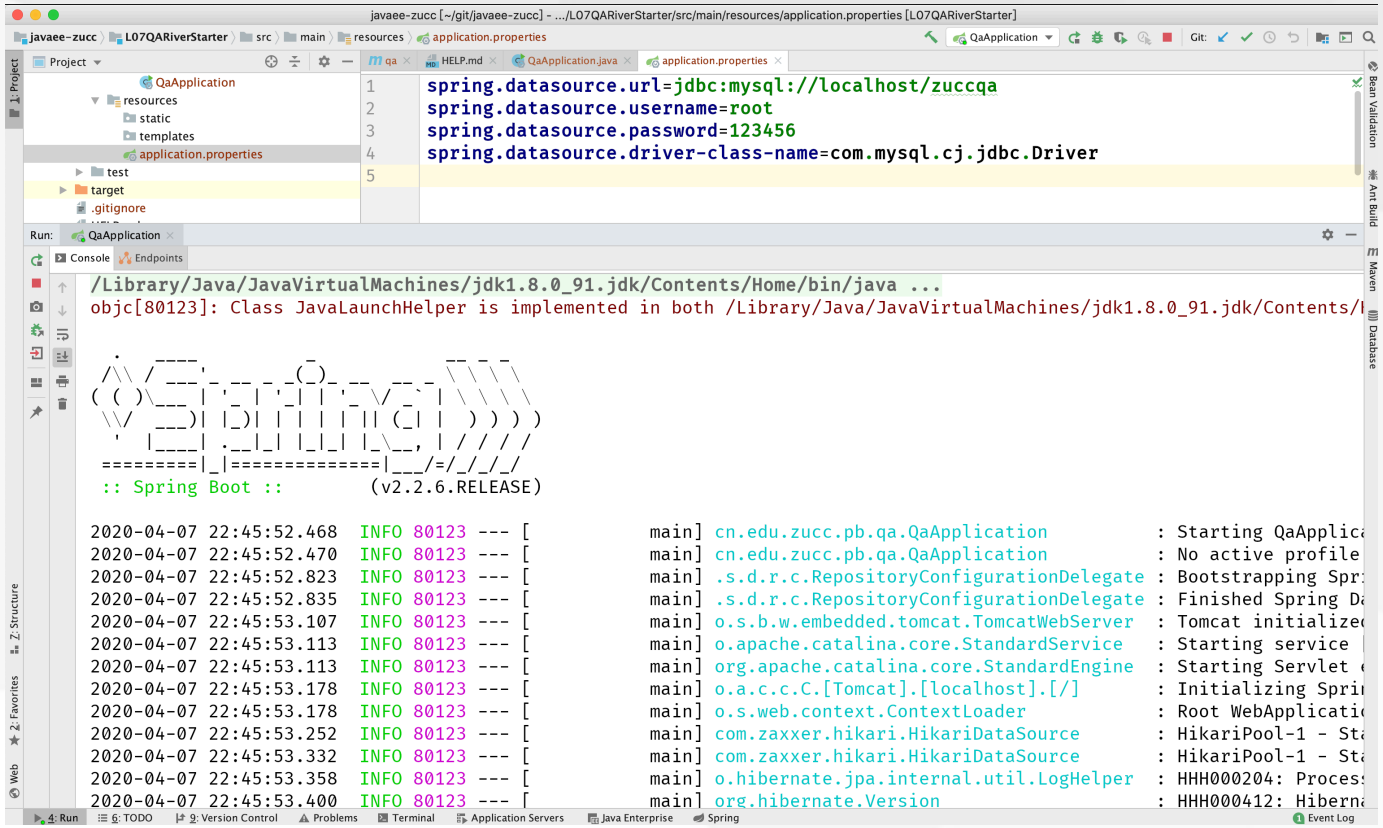
进一步配置

如果现在直接启动，我们可以看见“Spring Boot”启动的标志，但是由于还没有配置数据源，所以还不会启动成功。



配置数据源

通过spring boot配置数据源特别简单，在application.properties文件中输入下面的信息即可。所需要的包刚才根据向导创建的时候已经通过maven下载了。完成配置后再次启动，一个“没有”功能的web应用就启动了。

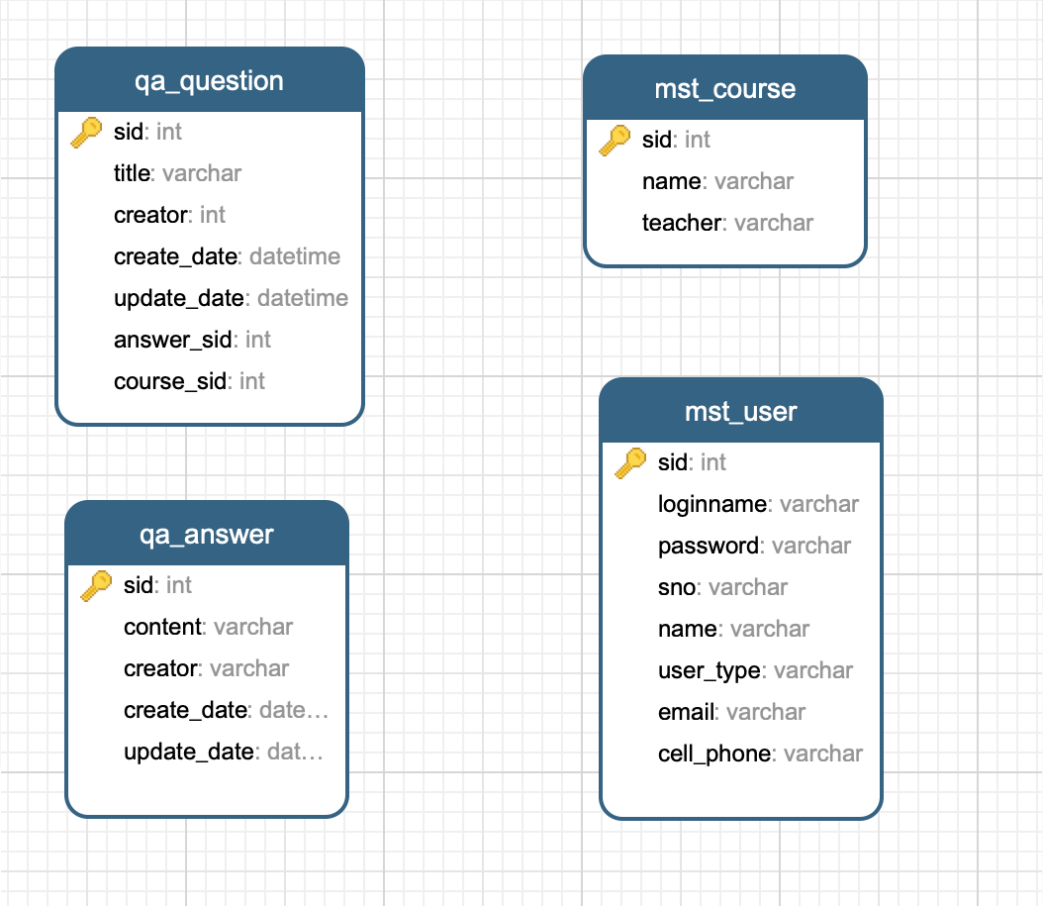


https://spring.io/guides/gs/accessing-data-mysql/

接下来我们设计我们的数据库

我们先看看我们接下来几周要做的系列作业的需求。

作为示例我们设计了两个简单的表格（自己的作业请理解需求和仔细设计，以更好的满足需求）



我们为什么不设计外键？

来一个接口

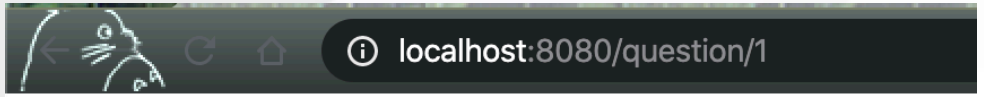
创建一个接口非常简单，类似我们之前的作业，只要标注

RestController建立一个控制器，然后在里面各个方法上标注各种Mapping就可以被url触发了。

qa | HELP.md | QaApplication.java | application.properties | QuestionController.java | Question.java

```java
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author pengbin
 * @version 1.0
 * @date 2020-04-07 23:04
 */
@RestController
public class QuestionController {
    @GetMapping("/question/{id}")
    Question one(@PathVariable Long id) {
        return new Question();
    }
}
```




```
{
    sid: 0,
    title: null,
    createDate: null,
    creator: 0
}
```

来一个数据库对象

创建一个数据库对象非常简单，对着数据库表输入就可以

QaApplication.java | application.properties | QuestionController.java | QuestionEntity.java | QuestionRepository.java | JpaRepository.class | Question.java

```java
@Entity
@Table(name = "qa_question")
public class QuestionEntity {
    private @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    int sid;

    private String title;
    private int creator;
    @CreationTimestamp
    @Column(name = "create_date")
    private Date createDate;
    @UpdateTimestamp
    @Column(name = "update_date")
    private Date updateDate;

    public int getSid() {
        return sid;
    }

    public void setSid(int sid) {
        this.sid = sid;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

```

- 标注entity对象及表名
- 标注id及自动增长
- 标注字段和属性名不同的字段
- 标注创建时间字段
- 标注更新时间字段

来一个数据库操作

来一个repository，我们通过接口继承JpaRepository就具备的主要的增删改查接口，不用写任何SQL语句。

```java
package cn.edu.zucc.pb.qa.repositories;

import cn.edu.zucc.pb.qa.entity.QuestionEntity;
import org.springframework.data.jpa.repository.JpaRepository;

/**
 * @author pengbin
 * @version 1.0
 * @date 2020-04-07 23:25
 */
public interface QuestionRepository extends JpaRepository<QuestionEntity, Integer>
}
```

*具体的SQL操作实现由spring框架帮我们完成了。

依赖注入

Controller中需要repository，那我们就注入一个（构造函数注入方式）

```
@RestController
public class QuestionController {

    private final QuestionRepository repository;
    QuestionController(QuestionRepository repository){
        this.repository = repository;
    }
```

一个创建接口就完成了

最后书写一下下面的代码，直接获取提交的json数据（自动填充到Question对象），然后调用repository的save方法，数据库里就有了数据（时间是自动生成的，insert语句是不用写的）

```
    @PostMapping("/question")
    Question saveOrUpdate(@RequestBody Question newQuestion) {
        QuestionEntity entity = new QuestionEntity();
        BeanUtils.copyProperties(newQuestion, entity);
        QuestionEntity retEntity = repository.save(entity);

        Question ret = new Question();
        BeanUtils.copyProperties(retEntity, ret);
        return ret;
    }
```

为啥我们要创建FormBean对象和Entity对象，然后复制来复制去的呢？

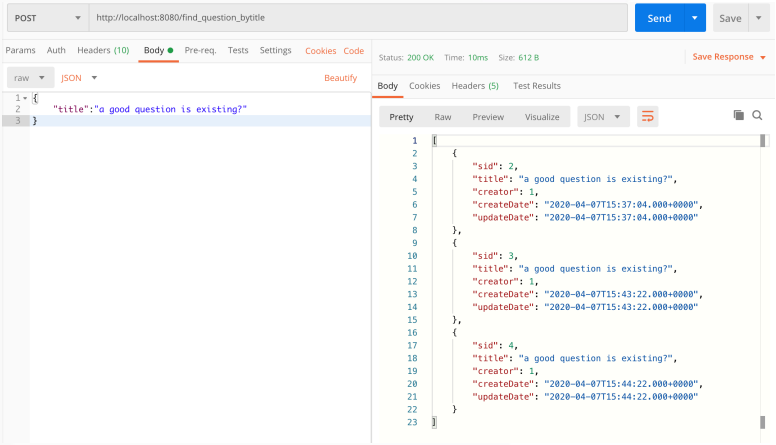
神奇的“约定”

我们通过Spring Data JPA的“名称约定”来完成一个查询接口，同样我们没有编写任何查询的实现代码，依赖Spring自动帮我们完成。我们只需要做：

```
public interface QuestionRepository extends JpaRepository<QuestionEntity, Integer>
    List<QuestionEntity> findByTitle(String title);
}
```

在刚才的接口中声明一个方法，这个方法满足一个约定，“find” + “By” + “字段名”，就可以依赖Spring帮我们自动生成实现方法。直接调用就可以获取返回结果。

```
@PostMapping("/find_question_bytitle")
List<QuestionEntity> findByTitle(@RequestBody Question queryExample) {
    return repository.findByTitle(queryExample.getTitle());
}
```



进一步探索

Spring Data支持非常多常用的命名模式，以支持常见的快速查询，非常方便

Table 3. Supported keywords inside method names

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is, Equals	findByFirstname, findByFirstnameIs, findByFirstnameEquals	… where x.firstname = ?1
Between	findByStartDateBetween	… where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull, Null	findByAge(Is)Null	… where x.age is null
IsNotNull, NotNull	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods

进一步探索~书写SQL语句

Spring Data支持使用Query标注进行SQL语句书写，而且支持表达式模式用于绑定变量。

Example 61. Declare query at the query method using `@Query`

```java
public interface UserRepository extends JpaRepository<User, Long> {

  @Query("select u from User u where u.emailAddress = ?1")
  User findByEmailAddress(String emailAddress);
}
```

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.at-query
https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query.spel-expressions

END

Pb&Lois