# 软件工程项目笔记

## 关于项目的部署

本软件项目使用Spring Boot作为底层启动框架，软件项目管理工具使用的是build.gradle。

其他有关的框架有：

1 使用Spring Data JPA作为持久层访问的框架，其特点是使用一系列@Annotaion作为实体的标记工具，用于控制数据库的同步生成。以及对数据持久层的访问。

2 使用Spring MVC作为web层的核心架构。页面模板引擎为Thymeleaf，其主要特点是可以快速显示Model中存储的信息，比较快速和方便。

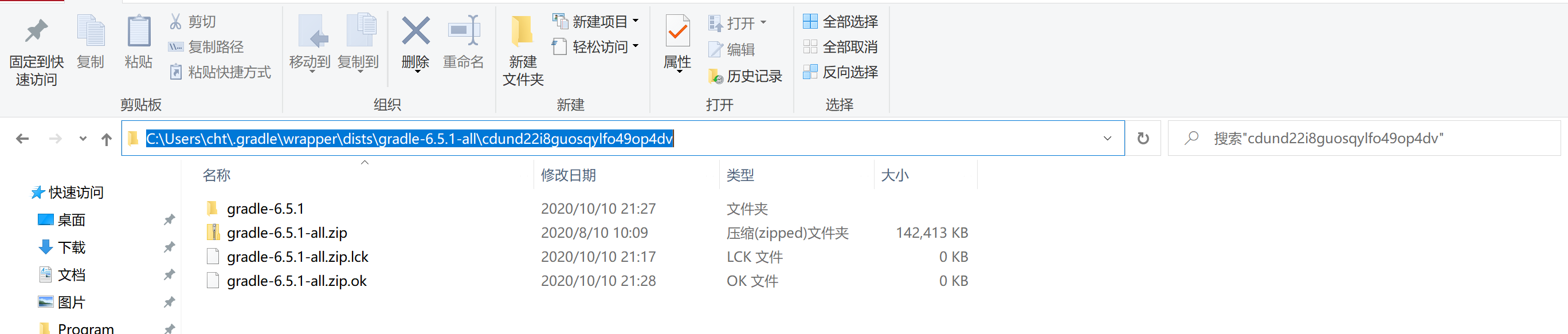
3 前端主要使用纯html和少量的js代码展示页面。

软件的部署分以下几部完成：

### 1 构建项目

用IntelliJ IDEA打开server文件夹，点击IDE右上角的build按钮进行项目构建。在构建过程中可能会出现下列问题。

a) 网络异常，无法下载gradle-6.5.1-all.zip。原因：gradle构建包有时候会由于网络问题无法下载，这个时候需要手动下载包到缓存目录。首先找到用户目录（常常为：c:/Users/${userName}），找到./gradle/wrapper/dists文件夹（如果没有则创建，然后），然后找到gradle-6.5.1-all/${base64}（为base64的一串码）/文件夹。然后将gradle-6.5.1-all.zip放到这个目录。



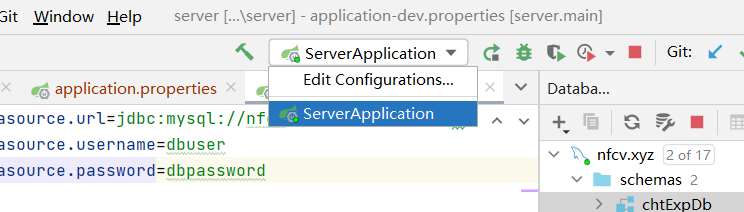
重新启动IDE，并开始测试。

### 2 配置数据库

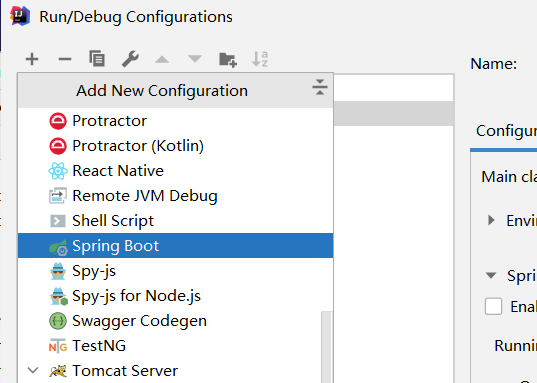
a) 在application.properties的同级目录中创建application-dev.properties，然后加入与数据库有关的配置项。例如：

|  |
| --- |
| spring.datasource.url=jdbc:mysql://nfcv.xyz:3306/chtLearning spring.datasource.username=dbuser spring.datasource.password=dbpassword |

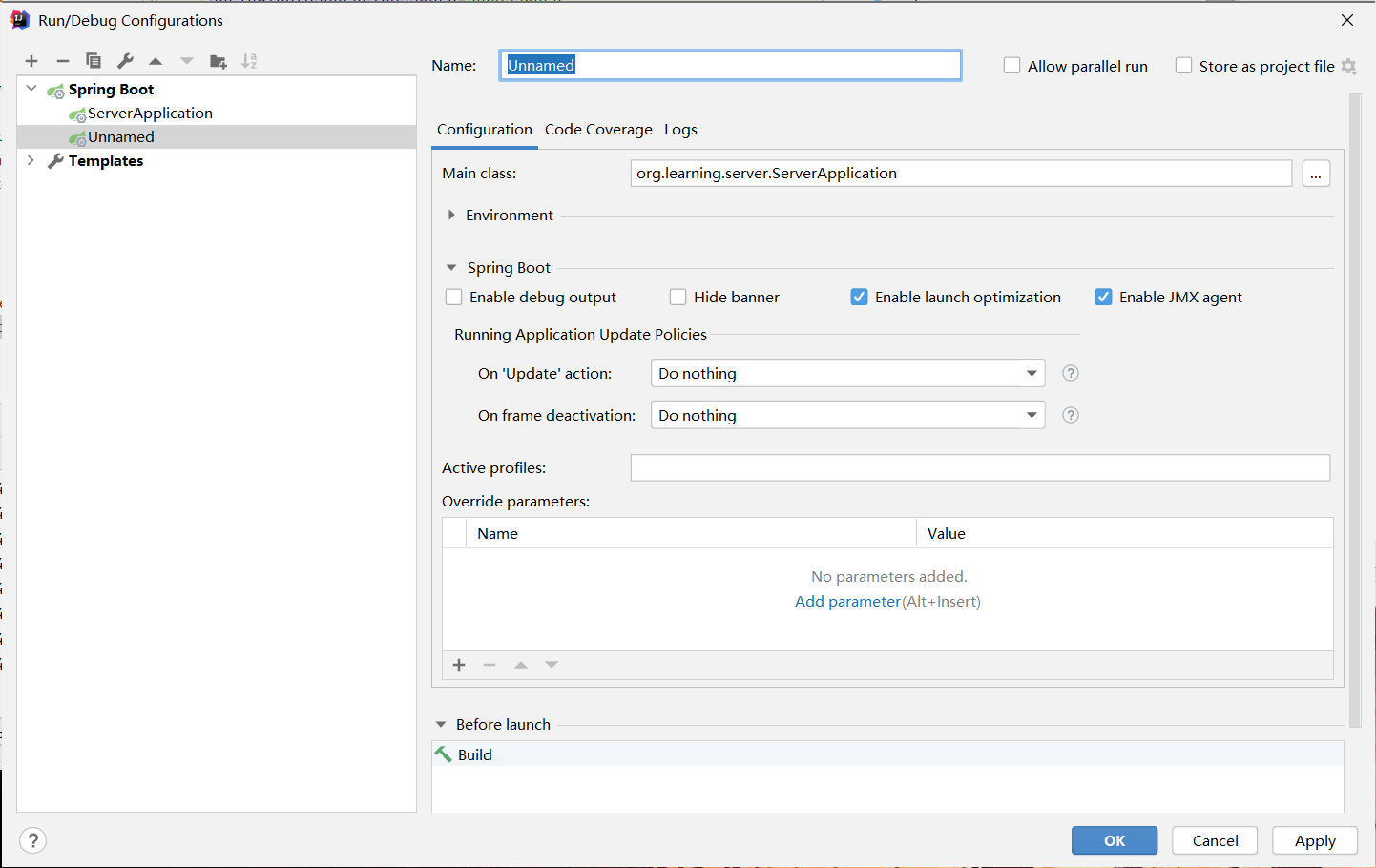
### 3 配置启动项



点击Edit Configuration…



找到Spring Boot



修改Name和Main class（org.learning.server.ServerApplication）。

## 关于build.gradle文件

build.gradle是部署与配置这个项目的核心。其中repositories指定软件依赖库的仓储。而dependencies指定软件的依赖。

在repositories中

|  |
| --- |
| maven **{** url 'https://maven.aliyun.com/repository/google' **}** maven **{** url 'https://maven.aliyun.com/repository/public' **}** |

两行制定了国内的maven源，主要是由于国际源常常由于网络的原因，连接不到。

dependencies中的各项即软件的依赖项，使用implementation/compile + 软件描述符的方式进行描述。其中软件描述符有3部分组成：groupName:archiveName:version。当软件需要依赖时，在其中添加依赖并build即可。

## 包的作用和说明

entity：存放与数据库实体相关的类。

repository：存放dao类型，其为一个接口，继承CrudRepository，从而实现数据库的Crud操作。

service：存放服务，业务的核心部分，一般能通过单元测试。其中实现类放在service.impl包中。

controller：控制器类，用于控制url->页面（数据）的跳转关系。

form：存放表单类型，一般用于存放可验证的表单类。

form.pattern：存放要用到的pattern信息

## 1.2 用户基础设置

使用到的教程：

1 如何进行表单验证（配合thymeleaf）<https://www.jianshu.com/p/48a4feab7541>

2 thymeleaf入门：<https://zhuanlan.zhihu.com/p/24988845?refer=dreawer>

数据库设计：请见实体entity.User

注册、登录：

使用到的Form：UserRegisterForm

使用到的其他类型：UserController，User，ActionResult，UserRepository，UserService

## 15周验收内容

1 组织使用树型结构进行管理。（使用回溯表节点进行管理）。分成多种树型的结构。

2 课程内容的设置参考Mooc。

3 体现组织之间管理的关系，例如上级的管理员可以对下级的节点进行管理。

4 课程资料库。课程。和课程开设的关系。

5 课程进度的记录，以及课程的管理等。

## 系统的结构设计

### 模块与功能

功能点：

（模块课程学习，课程查找等）

1、（查找课程学习）能够查找当前进行中的**课程学习(CourseLearning)**，课程学习与课程，学员，架构节点关联（其中架构节点可以为空，表示该课程是学员自己主动学习的进程）。存在一个学员同时在不同的架构节点学习同一门课程的情况，不同的架构节点使用不同的**课程学习进度**。但可以通过课程学习进度同步操作同步课程的学习进度。

返回的信息：课程学习列表（全），在主页(/)上显示，接口/api/course/current

2、（查找管理课程）用户能够在首页查找自己管理的课程，进入后进入课程管理界面。（管理课程界面能够进行课程内容的概览，但不记录学习记录）

接口：/api/course/manage\_current。

2、（智能查找）智能查找（现阶段由模糊匹配替换）该学员可以学习（已学习，公开的课程）的课程。并按照匹配度，热度（仅一个月内学习的人数），兴趣匹配度的函数进行排序。

返回的信息：课程列表（分页），在课程(/course)上显示，默认为按照热度，兴趣匹配度排序，接口/api/course/search?name=:{name}

3、（课程细节查询）用户从首页或者课程搜索界面进入，进行课程的学习，当没有学习的课程时，**显示订阅该课程按钮**。需要显示该课程的各个目录的信息，以及各个目录的完成度。

返回的信息：A、订阅的课程学习（如果没有则为空），以及对应的课程，还有**用户上次学习的位置**。接口/api/course/:{course\_id}&[course\_learn\_id=:{course\_learn\_id}]（这个为空表示用户是从课程搜索界面进入的）

B、如果用户已经订阅过该课程但从课程搜索界面中进入时，要求学员选择一个课程学习，**但是如果学员只有当前课程的课程学习实例，则自动跳转**。（CourseLearning）。可以通过查询/api/course /get?uid=:{uid}，点击后重新发送请求。

4、（课程章节查询）用户点击某一节（默认为第一个为学习的tab）或者某一节上某个资料的tab时，跳转到某一个资料的入口。并显示课程内容、课程学习进度等信息。

返回的信息：一个节内各个tab的信息，进度信息（如果是视频还需要携带上次最后观看的时间等），以及对应资源的信息（包括记录进度信息，以及讨论区的id，还有参加课程的人员信息），供后续调用。

接口/api/course/:{course\_id}/:{chapter\_id}?tabId=:{tabId}

5、（课程媒体获取）接4，通过媒体的查询获取对应的媒体资料。

返回的信息：媒体，接口/api/course/media/:{media\_id}

6、（记录进度信息）接4，通过对应的接口对应课程的进度信息，有前端来上传学习的时长，以及学习到的位置等信息。

返回的信息：操作是否成功，接口/api/course/report?course\_learn\_id=&during=&record=

7、（用户对课程进行订阅）用户仅能对**公开的课程**进行订阅，进行订阅后，将会创建一个架构节点为空的**课程学习（CourseLearning）**。

返回的信息：新的CourseLearning，接口/api/course/:{course\_id}/order

8、（用户收藏课程）用户可以收藏一门课程，并将此课程导入到收藏列表。（当用户收藏课程时，并不意味者用户订阅了该课程）。因此用户界面既有订阅的课程列表，也有收藏的列表。两者是分开的。

返回的信息：用户的课程列表，接口/api/course/star

~~8、（连接讨论区）接4，用户点击某一节的课程后，会在课程的右侧显示讨论区，将会返回讨论区的id，供之后讨论区相关的上传，同步api调用。~~

~~返回的信息：对应的讨论区，接口/api/course/:{course\_id}/comment~~

9、（获取讨论区最新的消息）前端系统定期轮询后端，后去讨论区的最新消息，使用时间戳来同步。

返回的信息：增加的聊天记录，接口/api/comment/:{comment\_id}?timeStamp=?

10、（讨论区提交留言）向讨论区提交一份留言，接口/api/comment/:{comment}/submit?content=?

11、（获取对应的学习记录信息）

（管理课程内容相关）

1、（创建课程/编辑课程）一个用户创建一个课程，填写基础信息后创建课程，进入创建课程编辑面板。（编辑状态指此课程无法发布，且对编辑组的所有其他用户都处于不可见状态。）填写的基础信息有：课程名称，课程描述，课程节点，课程是否处于编辑状态，课程对外界的可见性。

注意：用户切换发布状态/或可见性将会对相关的用户（关注、收藏、订阅的用户）**进行通知的操作**。

返回的信息：当前课程的基础信息，接口/api/course/create?...

2、（创建课程中的章节部分）用户在创建课程编辑模板上进行添加章节、删除章节、编辑章节内容、移动章节的操作。

返回的信息：更新后的目录部分。接口/api/edit\_course/update\_chapter

/api/edit\_course/move\_chapter，/api/edit\_course/delete\_chapter，/api/edit\_course/add\_chapter等

3、（创建课程中的内容部分）用户在章节内添加内容，删除内容，修改内容，移动内容等操作，具体的实现方式和课程中的章节部分类似。

4、（邀请用户参加课程编辑）创建课程者可以邀请用户参加课程，这将导致系统给被邀请的**用户发送一则通知**。邀请用户的功能使用按用户名（uid/name）模糊查询（一个接口）。

返回的信息：操作的状态，返回最近协作者的状态。（具体协作者的实现方式可以参考Github的Collobraters）

5、（用户模糊查询）按照用户名（uid/name）进行查询，并返回用户的基础信息。供4调用。

返回的信息：用户的记录

接口/api/user/search?query=?

6、（用户上传视频资料/PPt资料/文本资料）创建课程后指定内容时，可以选择上传新的资料，也可以使用已经使用的资料。这部分分成两个部分，一个是编辑的用户查看当前的各种资料区，以及各种修改操作，另一个是使用资料到对应的课程。**资料库和课程进行绑定**。

接口：/api/edit\_course/:{course\_id}/resources（获取）

/api/edit\_course/:{course\_id}/rename（对资料重命名）

/api/edit\_course/:{course\_id}/upload（资料上传）

/api/edit\_course/:{course\_id}/delete（删除资料）

/api/edit\_course/:{course\_id}/bind?media\_id=&resource\_id=等

（开设课程管理）

1、（开设一门课程）组织的管理员可以对一个架构节点发布一项课程，称为开设课程，进行发布课程时需要指定整个课程的开始时间和结束时间，在这之后，将会开设一门开课。同时创建与之对应的**章节管理表**。

返回的信息：创建的开课信息，接口/api/open\_course/?id=...（此接口如果没有id则进行创建的操作，否则进行修改的操作等），当然创建的开课也是具有编辑状态的，只有关闭编辑状态才视为发布，同时会对**对应的用户进行通知操作**（包括管理的课程，以及学员）。

2、（管理员对章节表进行管理）管理员可以对每一个章节管理表进行修改课程开始时间以及结束时间，需要注意的是，所有课程的开始时间不能早于整个开设课程的开始时间，不能万余整个开设课程的结束时间等。

返回的信息：

3、（管理员获取管理的开课列表）**管理员可以获取在这个组织内管理的课程**，也可以在首页中获取管理的课程。

返回的信息：当前节点管理的课程列表 接口：/api/org/open\_course?node\_id=

4、（获取学员的学习统计情况）（与之对应的，学员也需要知道自己的学习进度）管理员可以在开设的课程内获取学员的统计情况，

返回的信息：当前开设的课程的各个学员的学习进度。

5、（催促进度）管理员可以对学员进行进度的催促，这个是扩展功能，当前不考虑。

（组织管理）

1、**（已经完成，组织的创建）**组织的创建者可以创建一个组织，创建组织后，当前用户将会成为该组织的主管理员。需要注意的是组织可以设置对外公开状态，如果对外不公开，则只能组织邀请用户登录。接口/api/org/create

2、（组织中部门节点的增删改查）

3、（批量创建用户）组织可以批量创建账户，创建的用户则默认加入该组织，**且不能退出该组织**。

4、（获取组织中某个节点下的用户列表）

5、（进行用户的移出）

6、（用户主动退出组织）

（其他杂项）

1、（用户登录）用户登录