Agenda

产品实战（第一天）：如何在 1 天之内交付面试评估系统

1. 产品背景，迭代思维与 MVP 产品规划
2. 数据建模 & 企业级应用数据库设计原则
3. 创建项目 & 实现候选人面试数据增、删、改
4. 实现候选人数据导入

极客时间

扫码试看/订阅

《 Django 快速开发实战》视频课程

产品背景，迭代思维与 MVP 产品规划

线下流程

准备简历 & 面试评估表

- HR：发出面试评估表模板（Word）到一面面试官 （邮箱发出来）
- 一面面试官：登录邮箱下载 Word 模板，每个学生拷贝一份
- 按学生名字命名文件， 录入学生名字，学校，电话，学历等

第一轮面试

- 一面官：每面完一个学生，填写 Word 格式的评估表中
- 一面官：面完一天的学生后，批量把 Word 文档 Email 到 HR
- HR：晚上查收下载评估表，汇总结果到 Excel，通知学生复试
- HR：同时把已经通知复试的学生信息，发送到技术二面复试官

第二轮面试和 HR 面试

- 二面官：查收 Email，下载 Word 格式的一面评估记录
- 二面官：复试后追加复试的评估到 Word 记录中，邮件到 HR
- 类似如上步骤的 HR 复试。

校园招聘面试评估表

应聘者姓名：_____ 手机号码：____ 应聘职位：_____

生源地：_____ 毕业学校：_____ 学历：_____

综合能力测评成绩：_____ 专业笔试成绩：_____

专业初试		专业初试得分：	
考核项目	评分		
学习能力			
专业能力			
评分标准： 极优秀：4.5-5，优秀：4-4.4，良好：3.5-3.9，一般：3-3.4，较差：3 分以下，不包括 3 分。			
专业初试结果		转荐部门：	面试官签名：

专业复试 1		专业复试得分：	
考核项目	评分	评语： 优势：	
学习能力			
专业能力			
追求卓越		顾虑和不足：	
沟通能力			
抗压能力			
评分标准： 极优秀：4.5-5，优秀：4-4.4，良好：3.5-3.9，一般：3-3.4，较差：3 分以下，不包括 3 分。 如果候选人在算法方面有特长，复试官可以对其有 0.1-0.3 的加分。			
专业复试 1 结果	☐ 建议录用 ☐ 待定 ☐ 放弃	建议方向 或 转荐部门：	面试官签名：

推荐专业复试 2		专业复试得分： （前三项分数按 4:4:2 的比例算出总分。后两项是筛选的标准，如果这两项上有明显缺陷，则可以否认该候选人。）	
考核项目	评分	评语： 优势：	
学习能力			
专业能力			
追求卓越		顾虑和不足：	
沟通能力			
抗压能力			
评分标准： 极优秀：4.5-5，优秀：4-4.4，良好：3.5-3.9，一般：3-3.4，较差：3 分以下，不包括 3 分。 如果候选人在算法方面有特长，复试官可以对其有 0.1-0.3 的加分。			
专业复试 2 结果	☐ 建议录用 ☐ 待定 ☐ 放弃	建议方向：	面试官签名：

HR 复试			评语： 优势：	HR 复试综合等级： ☐ ☐ ☐ ☐
考核项目	素质项	评级 （S，A，B，C）		
综合素质	责任心			
	坦诚沟通			
	逻辑思维			
	人际理解力		顾虑和不足：	
	发展潜力			
	稳定性			
HR 复试结果	☐ 建议录用 ☐ 待定 ☐ 放弃			HR 面试官签名：

迭代思维与 MVP 产品规划方法（OOPD）

- MVP：minimum viable product， 最小可用产品
- OOPD：Online & Offline Product Development， 线上线下相结合的产品开发方法
  - 内裤原则：MVP 包含了产品的轮廓，核心的功能，让业务可以运转
  - 优先线下：能够走线下的，优先走线下流程，让核心的功能先跑起来，快速做用户验证和方案验证
  - MVP 的核心：忽略掉一切的细枝末节，做合适的假设和简化，使用最短的时间开发出来
- 迭代思维是最强大的产品思维逻辑，互联网上唯快不破的秘诀
- 优秀的工程师和优秀的产品经理，善于找出产品 MVP 的功能范围

微信 1.0 的 MVP 迭代

- 只有 3 个功能
  - 聊天发文本消息
  - 发送图片
  - 自定义头像
- 没有更改用户昵称的功能
- 产品的目标
  - 替换掉短信的免费聊天工具

微信 1.0 for iPhone(测试版) 全新发布

发布日期：2011-1-21

发布版本：微信1.0 for iPhone(测试版) 点击下载

基于iPhone、iPod Touch平台的腾讯微信服务，带给您全新的消息体验，您可以使用微信快速收发消息，即时拍照分享，随时随地联系身边的朋友。支持iPhone4、iPhone3G/3GS、iPod Touch等多种设备。

快速消息：极速轻快的楼层式对话，带给您飞一般的聊天体验。

如何找出产品的 MVP 功能范围？

使用这些问题来帮助确定范围

- 产品的核心目标是什么？核心用户是谁？核心的场景是什么？
- 产品目标都需要在线上完成或者呈现吗？
- 最小 MVP 产品要做哪些事情，能够达到业务目标？
- 哪些功能不是在用户流程的核心路径上的？
- 做哪些简化，和假设，能够在最短的时间交付产品，并且可以让业务流程跑起来？

用户场景和功能清单：找出必须的功能

角色	功能	是否必须
HR	可以管理职位	
候选人	可以浏览职位列表，详情	
候选人	可以在线投递简历	
HR	查看候选人投递的简历，审核简历	
HR	导入候选人	
HR	**添加，修改候选人，查看候选人列表**	
管理员	可以添加面试官	
面试官	**可以进行一面，二面；HR可以进行终面**	
管理员	能够管理HR，面试官的角色	
HR/面试官	HR和面试官只能看到有权限的内容	

- **定义最小可用的面试评估系统**
- **哪些是可以线下人肉做的事情**
- **可以做出哪些假设来简化产品**

数据建模 & 企业级数据库设计原则

- 姓名,城市,本科学校,研究生学校,手机号码,邮箱,学历,专业, 应聘职位,测评成绩,笔试成绩,生源地
- 初试评估结果
  - 初试-学习能力得分
  - 初试-专业能力得分
  - 初试-优势,初试-顾虑和不足
  - 初试结果
  - 初试-推荐部门
- 1面面试官, 2面面试官
- 复试评估结果
  - 复试1得分,复试1-学习能力得分
  - 复试1-专业能力得分
  - 复试1-追求卓越得分
  - 复试1-沟通能力得分
  - 复试1-抗压能力得分
  - ……

企业级数据库设计十个原则

3 个基础原则，4 个扩展性原则，3 个完备性原则

3 个基础原则

- 结构清晰：表名、字段命名没有歧义，能一眼看懂
- 唯一职责：一表一用，领域定义清晰，不存储无关信息，相关数据在一张表中
- 主键原则：设计不带物理意义的主键；有唯一约束，确保幂等

企业级数据库设计十个原则

4 个扩展性原则（影响系统的性能和容量）

- 长短分离：可以扩展，长文本独立存储；有合适的容量设计
- 冷热分离：当前数据与历史数据分离
- 索引完备：有合适索引方便查询
- 不使用关联查询：不使用一切的 SQL Join 操作，不做 2 个表或者更多表的关联查询
  - 示例：查询商家每一个订单的金额
  - select s.shop_name, o.id as order_id, o.total_amount from shop s, order o where s.id = o.shop_id

企业级数据库设计十个原则

3 个完备性原则

- 完整性：保证数据的准确性和完整性，重要的内容都有记录
- 可追溯：可追溯创建时间，修改时间，可以逻辑删除
- 一致性原则：数据之间保持一致，尽可能避免同样的数据存储在不同表中

创建应用和模型，分组展示页面内容

- 创建应用:

  python ./manage.py startapp interview

- 注册应用

  在 settings.py 中添加 interview 应用

- 添加模型

  在 interview/models.py 里面定义 Candidate 类

参考资料

- 使用 Model 模型

  https://developer.mozilla.org/zh-CN/docs/Learn/Server-side/Django/Models

  https://docs.djangoproject.com/en/3.1/topics/db/models/

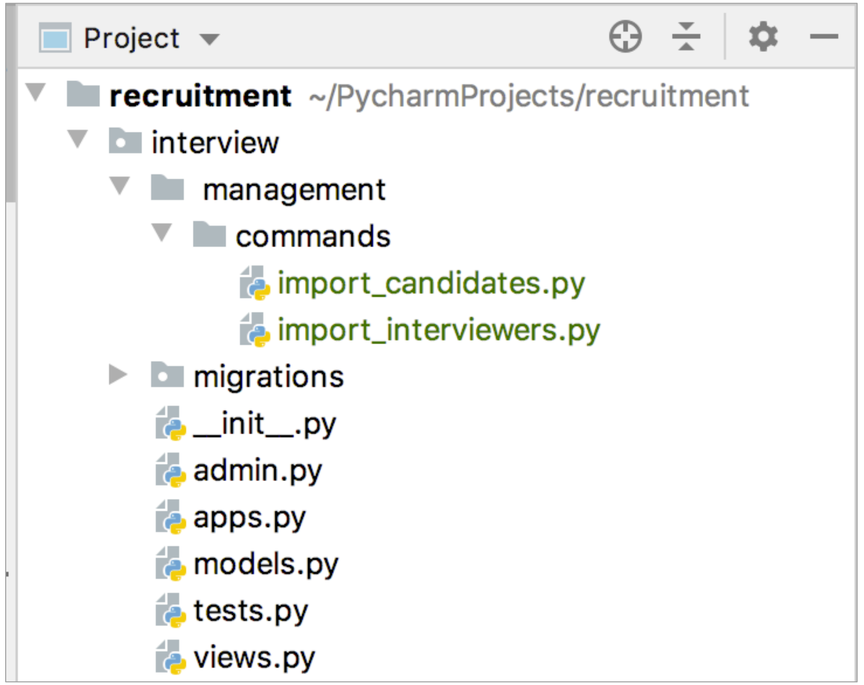
- 使用 Admin 管理类

  https://developer.mozilla.org/zh-CN/docs/learn/Server-side/Django/%E7%AE%A1%E7%90%86%E7%AB%99%E7%82%B9

  https://docs.djangoproject.com/en/3.1/ref/contrib/admin/

实现候选人数据导入

- 怎么样实现一个数据导入的功能最简洁
  - 开发一个自定义的 Web 页面，让用户能够上传 excel/csv 文件
  - 开发一个命令行工具，读取 excel/csv，再访问数据库写入 DB
  - 从数据库的客户端，比如 MySQL 的客户端里面导入数据
- Django 框架已经考虑到（需要使用到命令行的场景）
  - 使用自定义的 django management 命令来导入数据
  - 应用下面创建 management/commands 目录，
  - commands 目录下添加脚本，创建类，继承自 BaseCommand，实现命令行逻辑



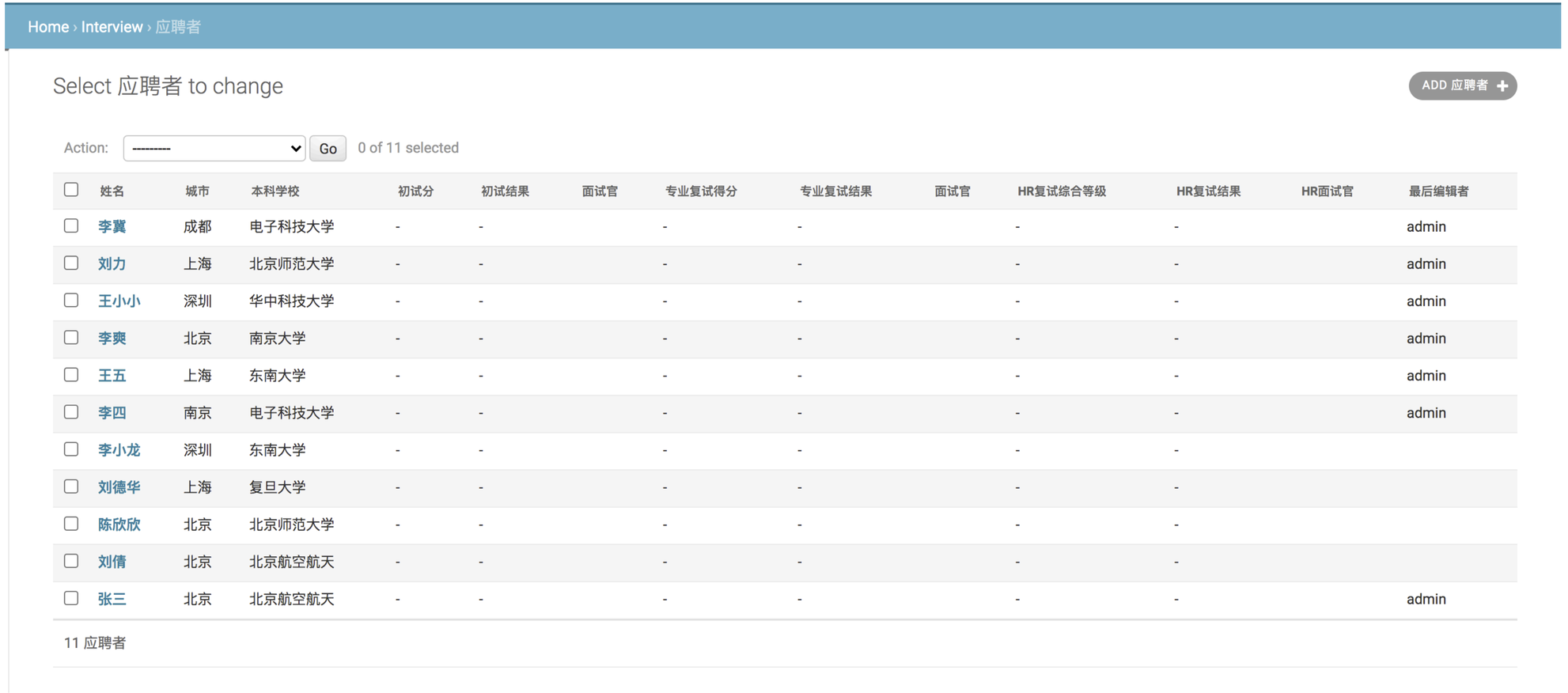
候选人数据导入

- 命令行导入

```
python manage.py import_candidates --path /path/to/your/file.csv
```

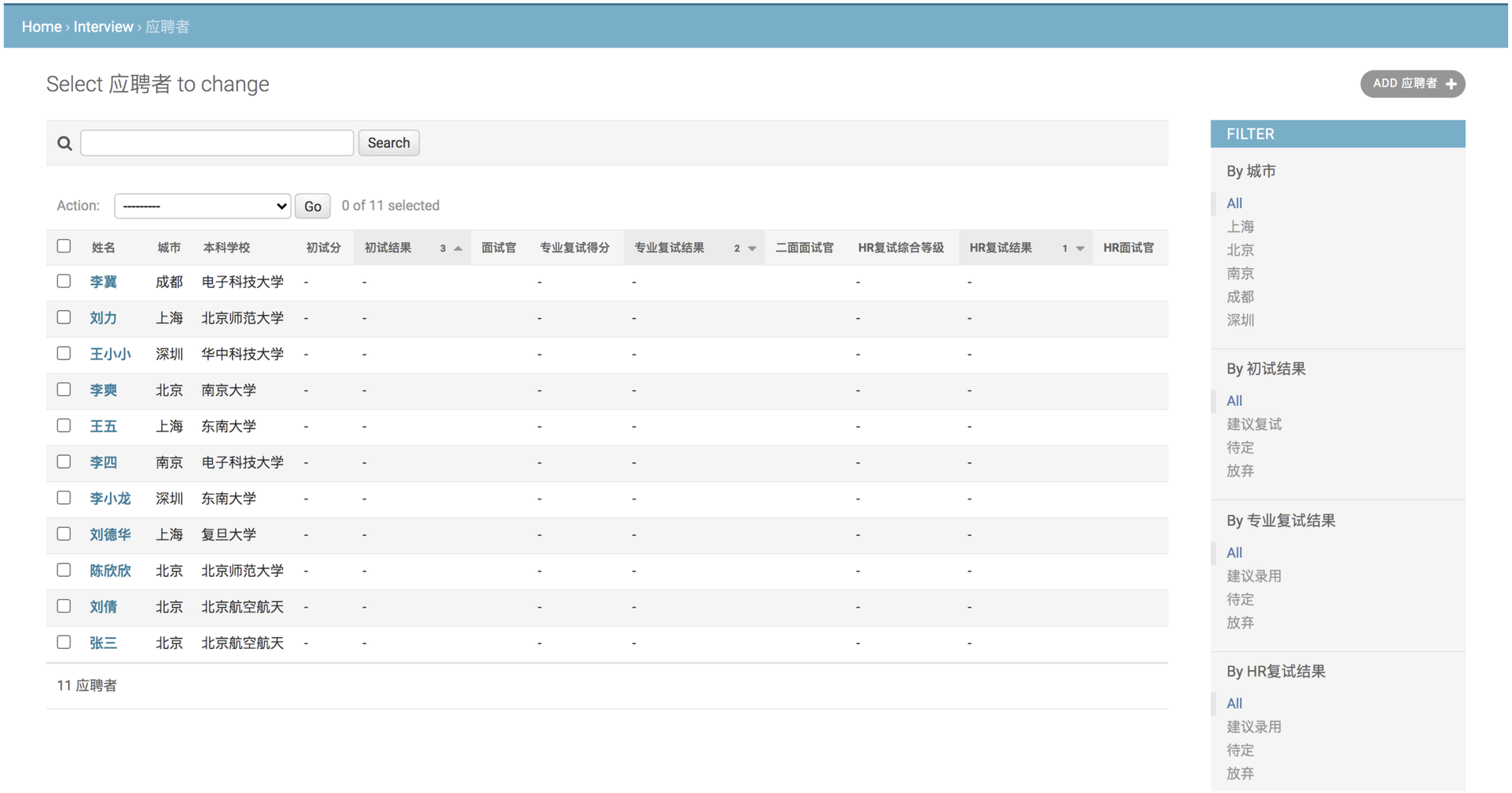
候选人列表筛选和查询

- 能够按照名字、手机号码、学校来查询候选人信息
- 能够按照初试结果，复试结果，HR复试结果，面试官来筛选；能按照复试结果来排序



候选人列表筛选和查询

- 期望效果



企业域账号集成

- 什么是目录服务 Directory Service ?
- 可以直接使用域账号登陆
- 不用手工添加账号，维护独立密码
- 可以集成 OpenLDAP/ActiveDirecotry
- 以 Open LDAP 为例

```
144  # The URL of the LDAP server.
145  LDAP_AUTH_URL = "ldap://localhost:389"
146  # Initiate TLS on connection.
147  LDAP_AUTH_USE_TLS = False
148
149  # The LDAP search base for looking up users.
150  LDAP_AUTH_SEARCH_BASE = "dc=ihopeit,dc=com"
151  # The LDAP class that represents a user.
152  LDAP_AUTH_OBJECT_CLASS = "inetOrgPerson"
153
154  # User model fields mapped to the LDAP
155  # attributes that represent them.
156  LDAP_AUTH_USER_FIELDS = {
157      "username": "cn",
158      "first_name": "givenName",
159      "last_name": "sn",
160      "email": "mail",
161  }
162
163  # A tuple of django model fields used to uniquely identify a user.
164  LDAP_AUTH_USER_LOOKUP_FIELDS = ("username",)
165
166  # Path to a callable that takes a dict of {model_field_name: value},
167  # returning a dict of clean model data.
168  # Use this to customize how data loaded from LDAP is saved to the User model.
169  LDAP_AUTH_CLEAN_USER_DATA = "django_python3_ldap.utils.clean_user_data"
170
171  # The LDAP username and password of a user for querying the LDAP database for user
172  # details. If None, then the authenticated user will be used for querying, and
173  # the `ldap_sync_users` command will perform an anonymous query.
174  LDAP_AUTH_CONNECTION_USERNAME = None
175  LDAP_AUTH_CONNECTION_PASSWORD = None
176
177  AUTHENTICATION_BACKENDS = {"django_python3_ldap.auth.LDAPBackend",'django.contrib.auth.backends.ModelBackend',}
178
```

- DN: 目录服务中的一个唯一的对象

CN=David,OU=Shanghai,DC=ihopeit,DC=com

Open LDAP 服务搭建

- Docker 启动 OpenLDAP 服务 & phpldapadmin-service
- docker run -p 389:389 -p 636:636 --name my-openldap-container --env LDAP_ORGANISATION="ihopeit" --env LDAP_DOMAIN="ihopeit.com" --env LDAP_ADMIN_PASSWORD="admin_passwd_4_ldap" --detach osixia/openldap:1.4.0
- docker run -p 80:80 -p 443:443 --name phpldapadmin-service --hostname phpldapadmin-service --link my-openldap-container:ldap-host --env PHPLDAPADMIN_LDAP_HOSTS=ldap-host --detach osixia/phpldapadmin:0.9.0
- 注意： 以上命令开放了 389 端口， 以及 443 端口到外网
- 阿里云服务器，需要在安全组里面开放 389, 443 端口； 不建议 80 端口开放出来

面试官的导入、授权

- 从 Open LDAP/AD 中导入面试官信息
  - 同步账号到Django
  - 设置面试官群组，授予群组权限：查看应聘者、修改应聘者（评估）
  - 设置用户属性 is_staff 为true：允许登陆 Django Admin
  - 添加用户到群组： 使得面试官登陆后，可以填写反馈

面试官填写面试评估反馈

第一轮面试

初试分:	5	学习能力得分:	4	专业能力得分:	4

优势:

有项目实践经验，能够针对日常遇到的问题，找到解决方案，使用软件来解决实际问题。
熟悉 Java 的日常开发，了解常用中间件的使用；
有不错的数据结构，算法，网络，操作系统的基础；
有快速学习的能力

顾虑和不足:

沟通表达略有不足

初试结果: 建议复试

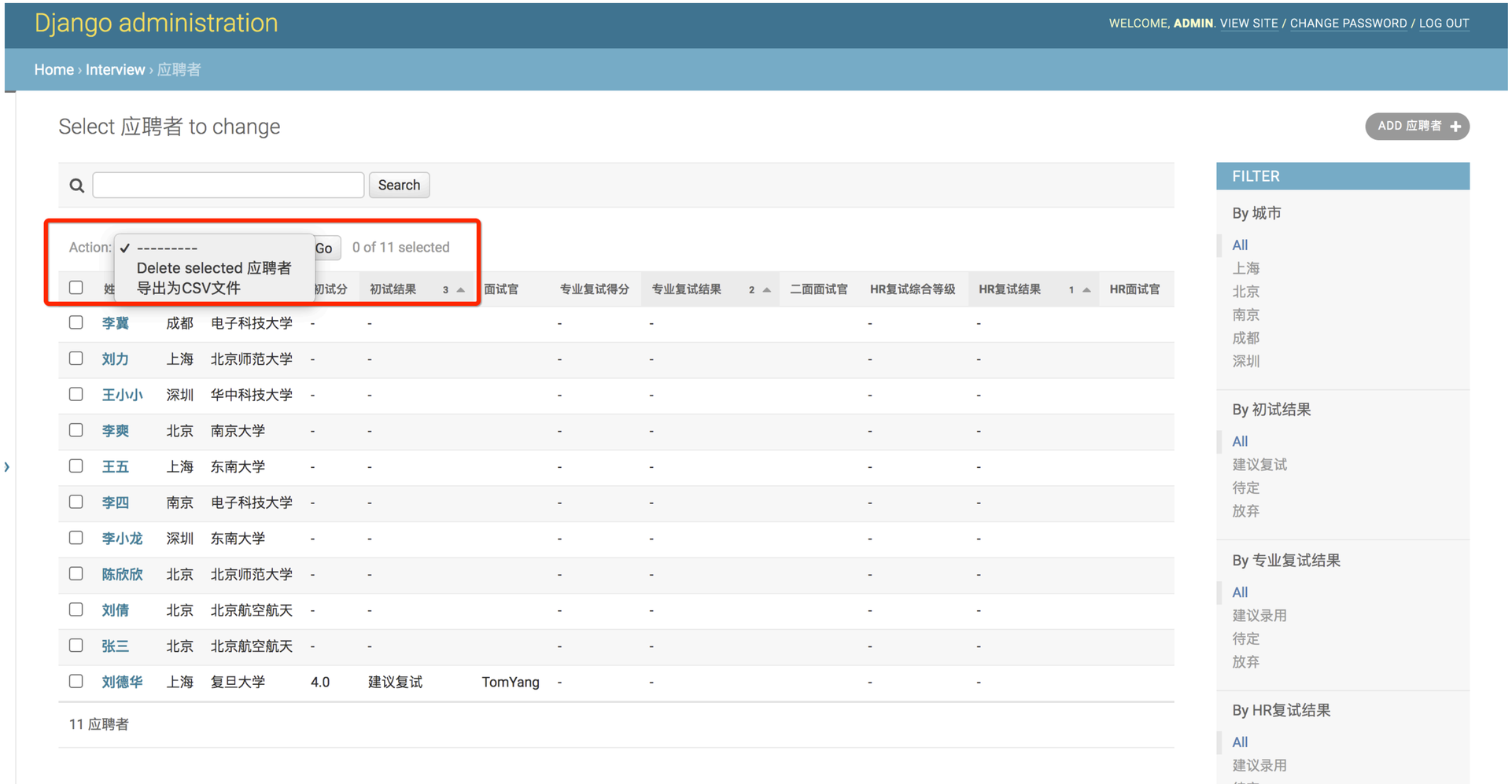
推荐部门:

面试官: davidullua

初试备注:

增加自定义的数据操作菜单 （数据导出为 CSV）

- 场景：需要对数据进行操作，比如导出，状态变更 （如 标记候选人为 “待面试”）
- 定义按钮的实现逻辑（处理函数） & 在 ModelAdmin 中注册函数到 actions



自定义按钮： 数据导出为 CSV

production.py | README.md | admin.py | response.py | apps.py | models.py | base.py | start.local.bat

```
11
12  ### define export action
13  def export_model_as_csv(modeladmin, request, queryset):
14      response = HttpResponse(content_type='text/csv')
15      field_list = exportable_fields
16      response['Content-Disposition'] = 'attachment; filename=%s-list-%s.csv' % (
17          'recruitment-candidates',
18          datetime.now().strftime('%Y-%m-%d-%H-%M-%S'),
19      )
20
21      ### 写入表头
22      writer = csv.writer(response)
23      writer.writerow(
24          [queryset.model._meta.get_field(f).verbose_name.title() for f in field_list],
25      )
26
27      for obj in queryset:
28          ## 单行 的记录（各个字段的值），  根据字段对象，从当前实例（obj）中获取字段值
29          csv_line_values = []
30          for field in field_list:
31              field_object = queryset.model._meta.get_field(field)
32              field_value = field_object.value_from_object(obj)
33              csv_line_values.append(field_value)
34          writer.writerow(csv_line_values)
35
36      return response
37
38
39  export_model_as_csv.short_description = u'导出为CSV文件'
40
41  # 候选人管理类
42  class CandidateAdmin(admin.ModelAdmin):
43      exclude = ('creator', 'created_date', 'modified_date')
44
45      actions = (export_model_as_csv,)
```

日志记录

- **四个组件**
  - Loggers：日志记录的处理类/对象，一个 Logger 可以有多个 Handlers
  - Handlers：对于每一条日志消息如何处理，记录到文件，控制台，还是网络
  - Filters： 定义过滤器，用于 Logger/Handler 之上
  - Formmaters: 定义日志文本记录的格式
- **四个日志级别**
  - DEBUG: 调试
  - INFO：常用的系统信息
  - WARNING: 小的告警，不影响主要功能
  - ERROR：系统出现不可忽视的错误
  - CRITICAL: 非常严重的错误

使用日志记录

- 记录 debug, info, warning, error, critical 不同级别日志

```python
# import the logging library
import logging

# Get an instance of a logger
logger = logging.getLogger(__name__)

def my_view(request, arg1, arg):
    ...
    if bad_mojo:
        # Log an error message
        logger.error('Something went wrong!')
```

配置日志记录

- Django 里面使用 dictConfig 格式来配置日志
- Dictionary 对象，至少包含如下内容:
  - version，目前唯一有效的值是 1
  - Handler, logger 是可选内容，通常需要自己定义
  - Filter, formatter 是可选内容，可以不用定义

配置日志记录

- 定义日志输出格式， 分别定义 全局日志记错， 错误日志处理， 自定义的 日志处理器

```python
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'simple': { # exact format is not important, this is the minimum information
            'format': '%(asctime)s %(name)-12s %(lineno)d %(levelname)-8s %(message)s',
        },
    },
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
            'formatter': 'simple',
        },

        'mail_admins': { # Add Handler for mail_admins for `warning` and above
            'level': 'ERROR',
            'class': 'django.utils.log.AdminEmailHandler',
        },
        'file': {
            #'level': 'INFO',
            'class': 'logging.FileHandler',
            'formatter': 'simple',
            'filename': os.path.join(os.path.dirname(BASE_DIR), 'recruitment.admin.log'),
        },
    },

    'root': {
        'handlers': ['console', 'file'],
        'level': 'INFO',
    },

    'loggers': {
        "django_python3_ldap": {
            "handlers": ["console", "file"],
            "level": "DEBUG",
```

```
[28/Aug/2020 16:54:33] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2198
[28/Aug/2020 16:54:33] "GET /admin/ HTTP/1.1" 302 0
[28/Aug/2020 16:54:33] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2198
LDAP connect succeeded
LDAP connect succeeded
LDAP user lookup succeeded
LDAP user lookup succeeded
[28/Aug/2020 16:54:39] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[28/Aug/2020 16:54:39] "GET /admin/ HTTP/1.1" 200 3379
```

定义日志格式后记录了更多内容

```
[28/Aug/2020 17:03:24] "GET /admin/logout/ HTTP/1.1" 200 1513
[28/Aug/2020 17:03:26] "GET /admin/ HTTP/1.1" 302 0
[28/Aug/2020 17:03:26] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2198
2020-08-28 17:03:30,862 django_python3_ldap.ldap 181 INFO     LDAP connect succeeded
2020-08-28 17:03:30,862 django_python3_ldap.ldap 181 INFO     LDAP connect succeeded
2020-08-28 17:03:30,902 django_python3_ldap.ldap 76 INFO     LDAP user lookup succeeded
2020-08-28 17:03:30,902 django_python3_ldap.ldap 76 INFO     LDAP user lookup succeeded
[28/Aug/2020 17:03:30] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[28/Aug/2020 17:03:30] "GET /admin/ HTTP/1.1" 200 3073
```

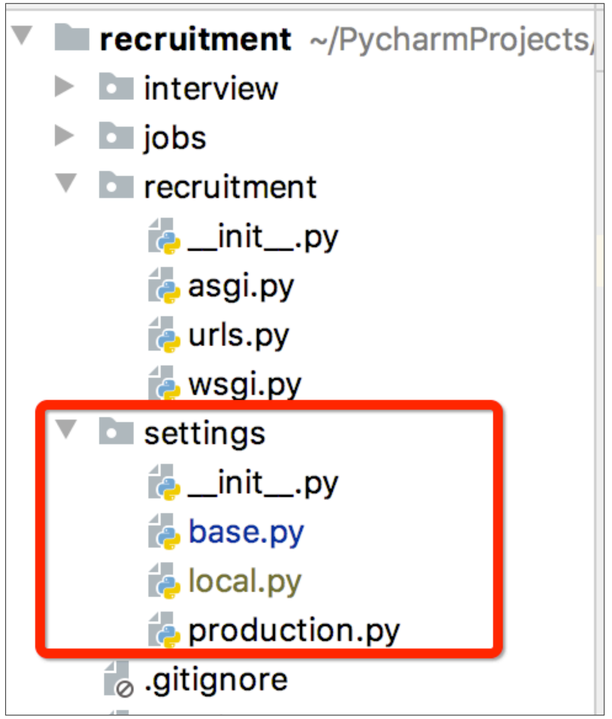
生产环境与开发环境配置分离

- 问题
  - 生产环境的配置与开发环境配置隔离开， 开发环境允许 Debugging
  - 敏感信息不提交到代码库中，比如数据库连接，secret key, LDAP连接信息等
  - 生产、开发环境使用的配置可能不一样，比如 分别使用 MySQL/Sqlite 数据库

生产环境与开发环境配置分离

- 推荐方案
  - 把 settings.py 抽出来，创建3个配置文件
    - base.py 基础配置
    - local.py 本地开发环境配置，允许 Debug
    - production.py 生产环境配置， 不进到 代码库版本控制
  - 命令行启动时指定环境配置
    - python ./manage.py runserver 127.0.0.1:8000 --settings=settings.local
    - 使得 manage.py 中的如下代码失效：

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'settings.base')
```



产品细节完善

- 修改站点标题
- 自动保存面试官信息
- 设置只读字段: readonly_fields = ('**first_interviewer**','**second_interviewer**',)
- 设置字段提示 help_text： 初试分，学习能力得分，专业能力得分范围 0-5 分
- 生成项目依赖: pip freeze > requirements.txt

参考

- Django document
  - ✓ https://docs.djangoproject.com/en/3.1/ref/contrib/admin/actions/
  - ✓ https://docs.djangoproject.com/en/3.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_filter
  - ✓ https://github.com/osixia/docker-openldap

极客时间

扫码试看/订阅

《 Django 快速开发实战》视频课程