Web 开发技术

Backend Development

Django

Web 后端技术栈

- 操作系统
  - Linux
- 服务器软件
  - Apache/Nginx
- 数据库软件
  - MySQL/MongoDB/PostgreSQL/Oracle/SQLite
- 编程语言
  - Python/Ruby/PHP, Java

内容提要

- 1 什么是 Django?
- 2 Why Django?
- 3 如何学好 Django?
- 4 预备知识
  - Virtualenv
  - MVC 和 MTV 设计模式
- 5 基础知识
  - 安装 Django
  - 创建一个 Django 项目
  - Django 项目的文件结构
  - 模型层、视图层和模板层
  - 首次运行 Django 项目
- 6 以工程视角切入：
  以创建投票系统项目为例

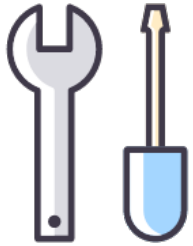
1 什么是 Django?

Django

- 是一款开放源代码的 Web 框架，由 Python 开发。
- 目标：使得开发复杂的、数据库驱动的网站变得简单。
- 注重组件的重用性和“可插拔性”，DRY (Don’t Repeat Yourself) 原则。

- 以法国吉他手 Django Reinhardt 命名。
- 最初是被开发来用于管理劳伦斯出版集团旗下的新闻内容网站，2005.7 开源在 BSD 许可证下发布。
- 2008 成立软件基金会。

2 Why Django?

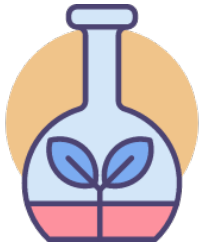


适合或有用

科研：成果展示平台
工程：Web 平台需求广泛

流行度

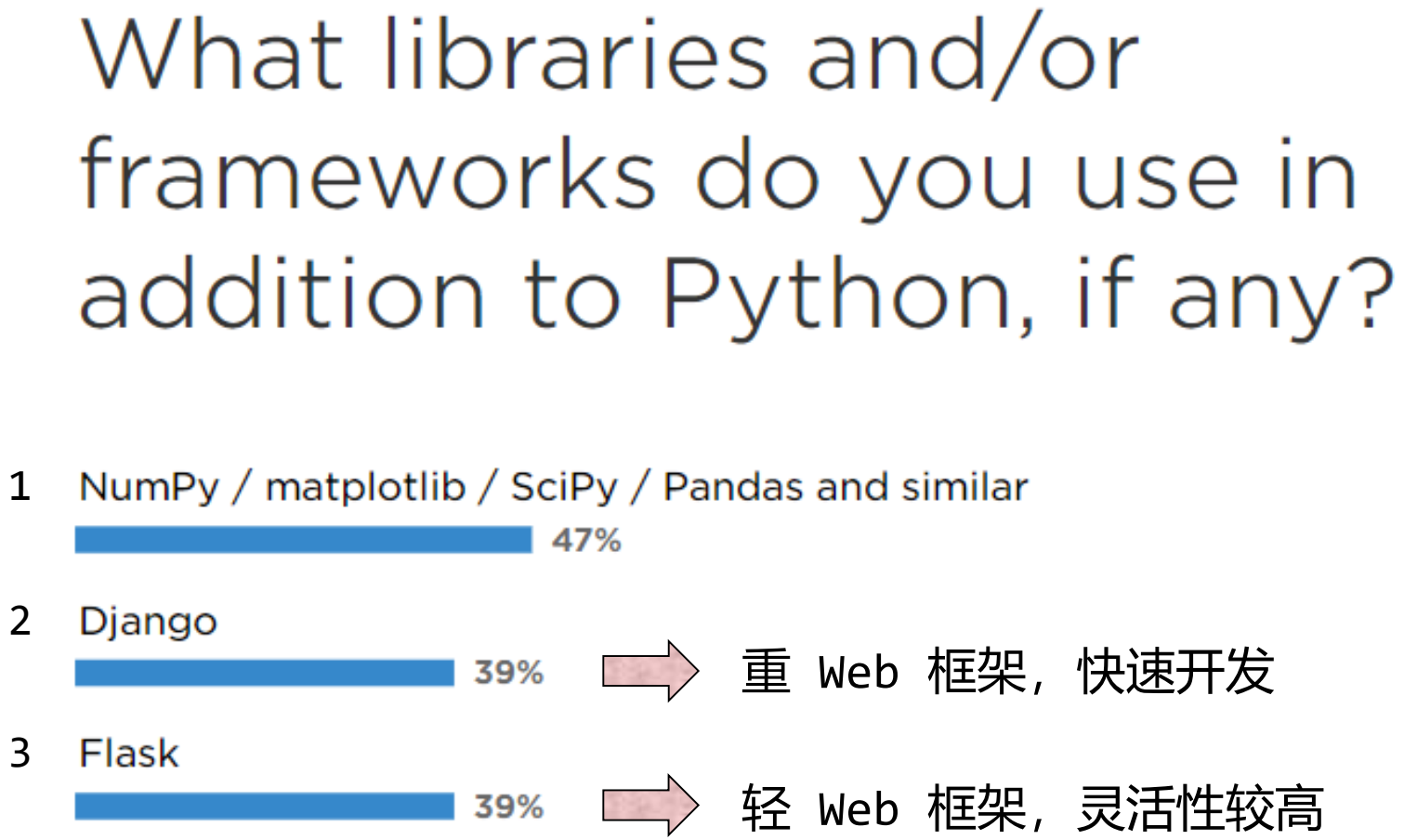
No.1 Python Web Framework
(JetBrains, 2018)



生态系统

完整精细的官方文档
全面的第三方包
丰富案例

2 Why Django?



重 Web 框架，快速开发

轻 Web 框架，灵活性较高

3 如何学好 Django

（1）认清本质

- Web - 工程项目
- Django – Web 框架

（2）查阅官方文档

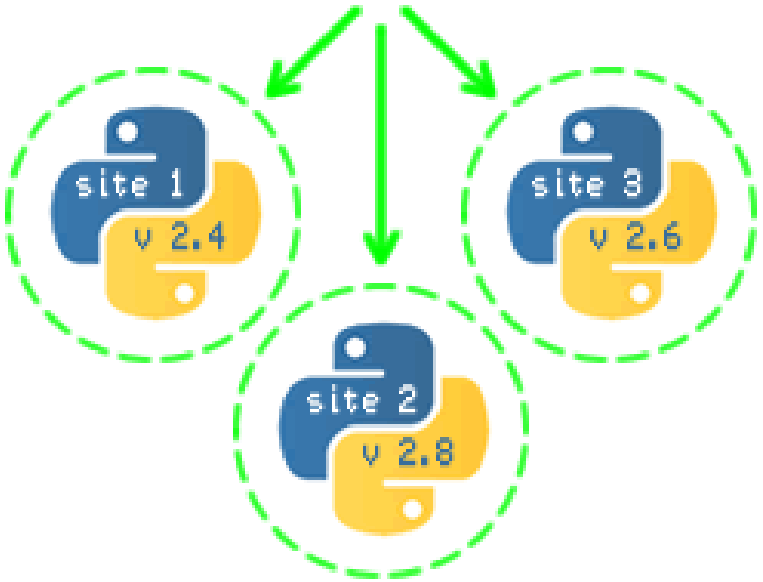
https://docs.djangoproject.com/

4.1 预备知识 - Virtualenv

Virtualenv

- 创建独立 Python 开发环境的工具，方便 Python 版本、包的管理。
- 解决不同开发项目的不同依赖、版本及间接权限的问题。
- 使用 Virtualenv 将创建一个独立的安装环境，不与其他环境共享。

```
> virtualenv
```



场景如：
项目一使用 Django 2.15 开发，
项目二使用 Django 1.70 开发，
在同一 Python 开发环境上无法同时开发和运行两个项目。
可使用 Virtualenv 隔离环境。

4.1 预备知识 - Virtualenv

安装并激活 Virtualenv

cmd/terminal

```
> pip install virtualenv          # 使用 pip 安装 Virtualenv
> virtualenv test                 # 新建名为 test 的虚拟环境目录，
                                  使用默认 Python 版本
```

Windows

```
> test\Scripts\activate
# 激活 Virtualenv

> test\Scripts\deactivate
# 关闭该 Virtualenv
```

Linux

```
> source test/bin/activate
# 激活 Virtualenv

> deactivate
# 关闭该 Virtualenv
```

4.1 预备知识 - Virtualenv

迁移模块

```
> pip freeze > requirements.txt          # 导出模块
```

```
> pip install -r requirements.txt        # 安装模块
```

Usage

```
pip freeze [options]
```

Description

Output installed packages in requirements format.

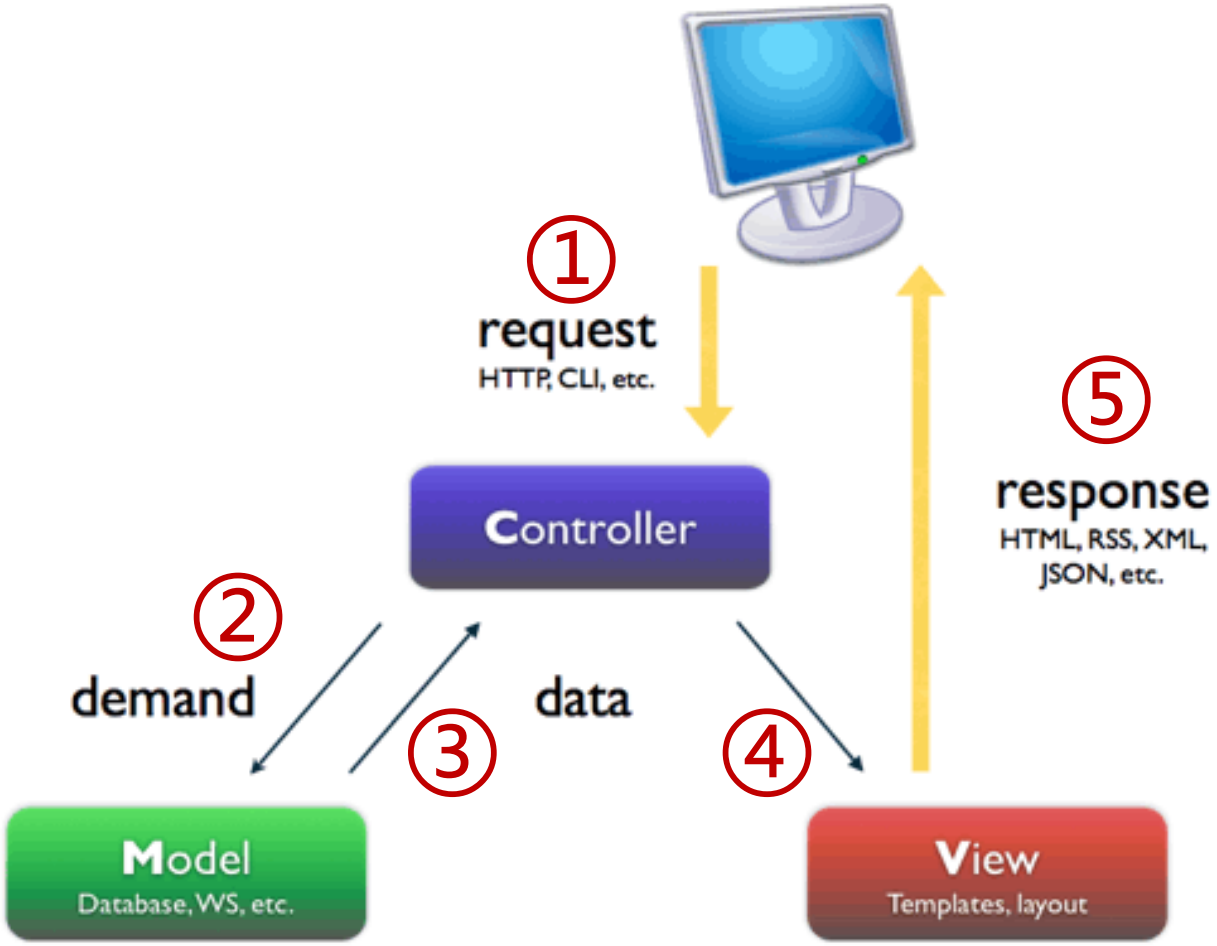
packages are listed in a case-insensitive sorted order.

4.2 预备知识 - MVC/MTV

经典的 MVC 模式

- Model-View-Controller 模式
- 广泛用于应用程序的分层开发

- Model（模型层）：代表一个存取数据的对象。
- View（视图层）：代表模型包含数据的可视化。
- Controller（控制器）：作用于模型和视图上。控制数据流向模型对象，在数据变化时更新视图。



4.2 预备知识 - MVC/MTV

Why MVC?

- Model 可重用：多个 View 共享一个 Model.
  - 场景：同一个 Web 应用程序提供多种用户界面，如 PC 大屏/手机小屏浏览器、App.
- Model 与 View 隔离：方便改变数据层和业务规则。
  - 场景：将数据库从 MySQL 迁移至 MongoDB，只需修改 Model，无需修改 View.

=> 提高了程序的灵活性和可配置性。

Controller 用于连接不同 Model 和 View 完成用户需求。

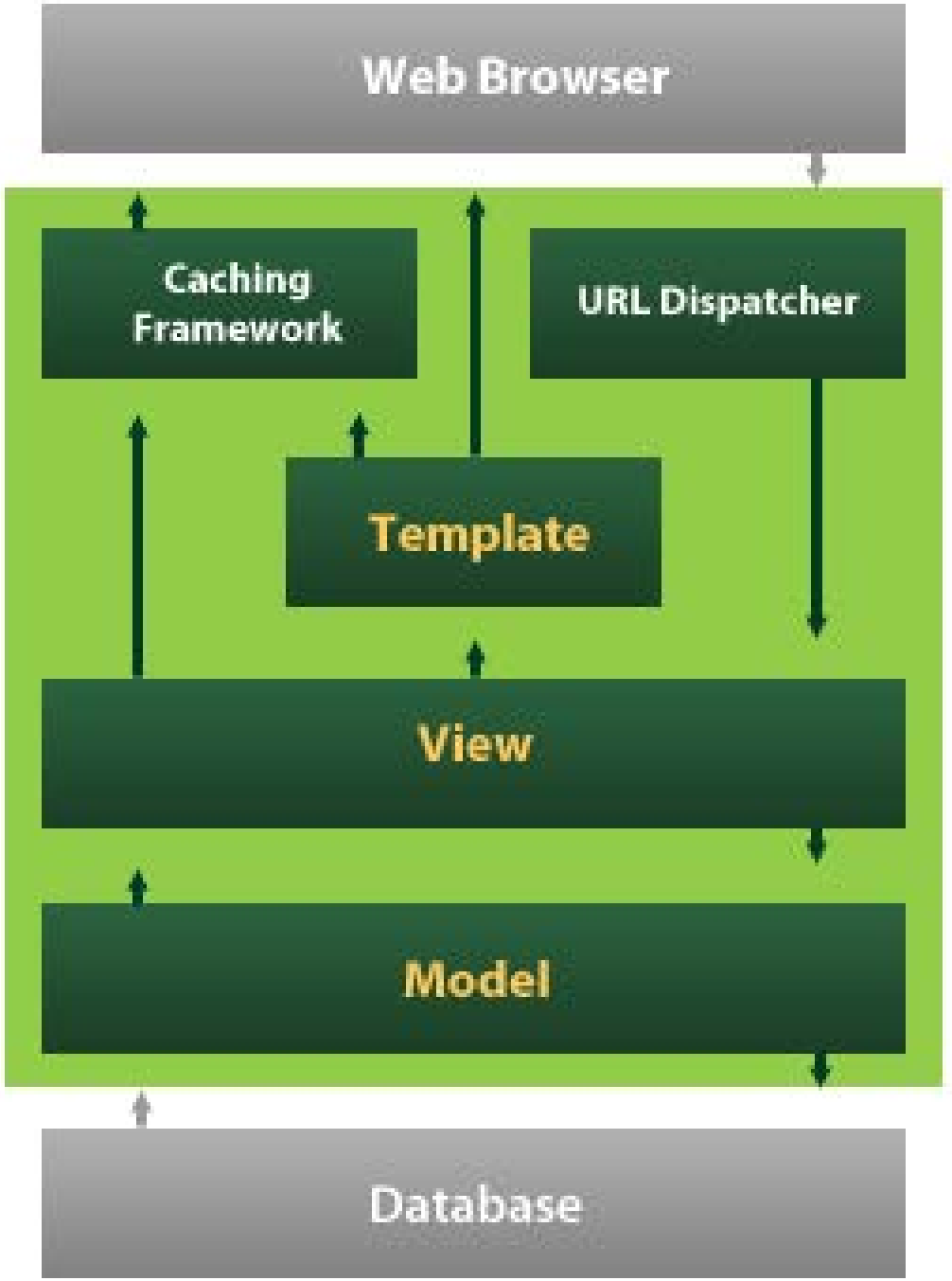
4.2 预备知识 – MVC/MTV

MTV：Django 实现的 MVC 模式

MVC 与 MTV 的对应关系

MVC	MTV
Model	Model
View	Template
Controller	View

官方问答：
Django 似乎就是个 MVC 框架。
但你们把 Controller 叫做 View，
View 叫做 Template。有何不同？



5 基础知识 - Virtualenv

基础知识

- 安装 Django　　`pip install django`
- 创建一个 Django 项目　　`django-admin startproject firstapp`
- 创建一个 Django 模块　　`python manage.py startapp polls`
- Django 项目的文件结构
- 模型层、视图层和模板层
- 首次运行 Django 项目　　`python manage.py runserver 8080`

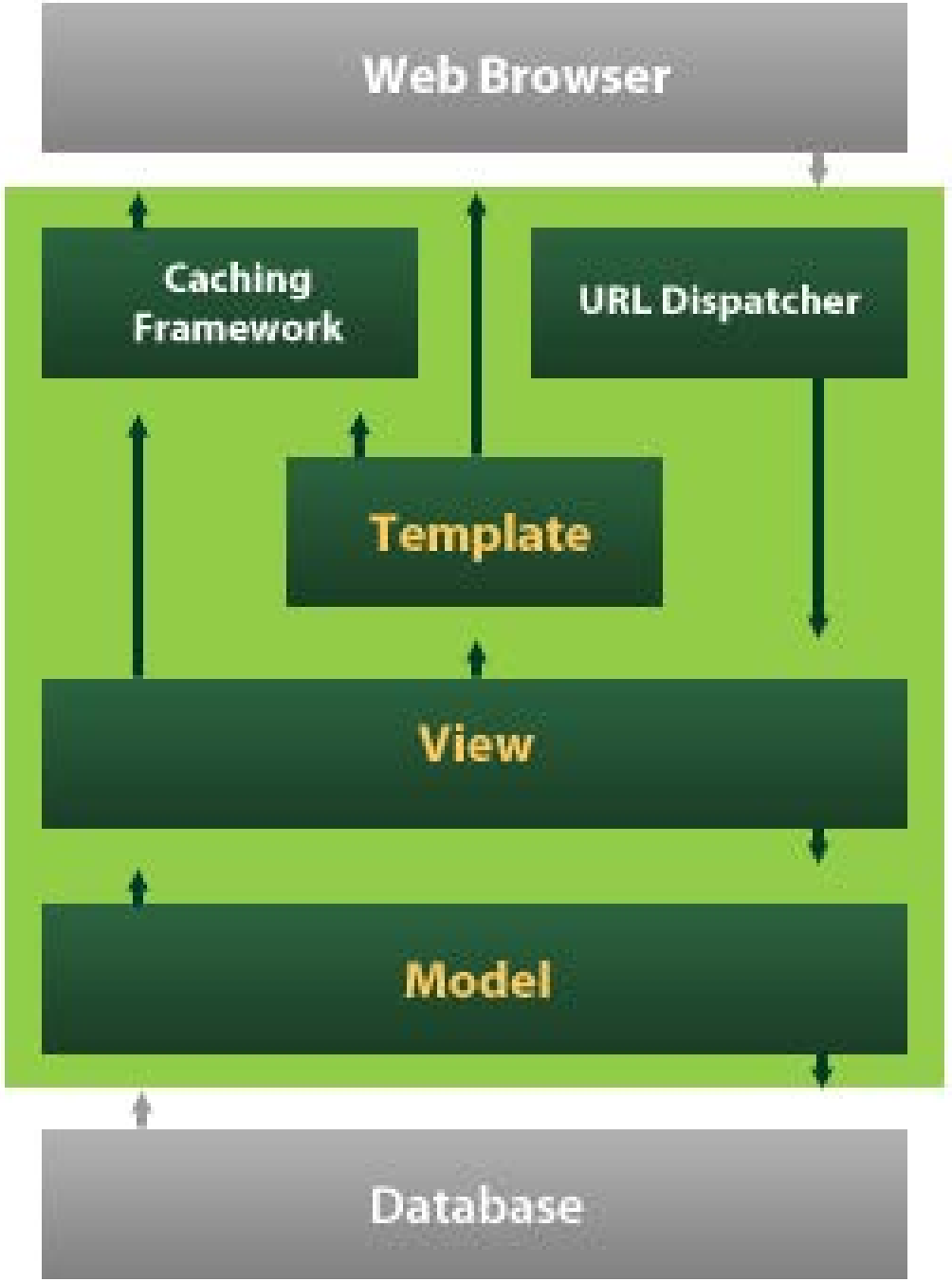
6 以工程视角切入

-> 浏览器访问 URL

-> urls.py 将访问的 URL 映射
至 views.py 的函数中

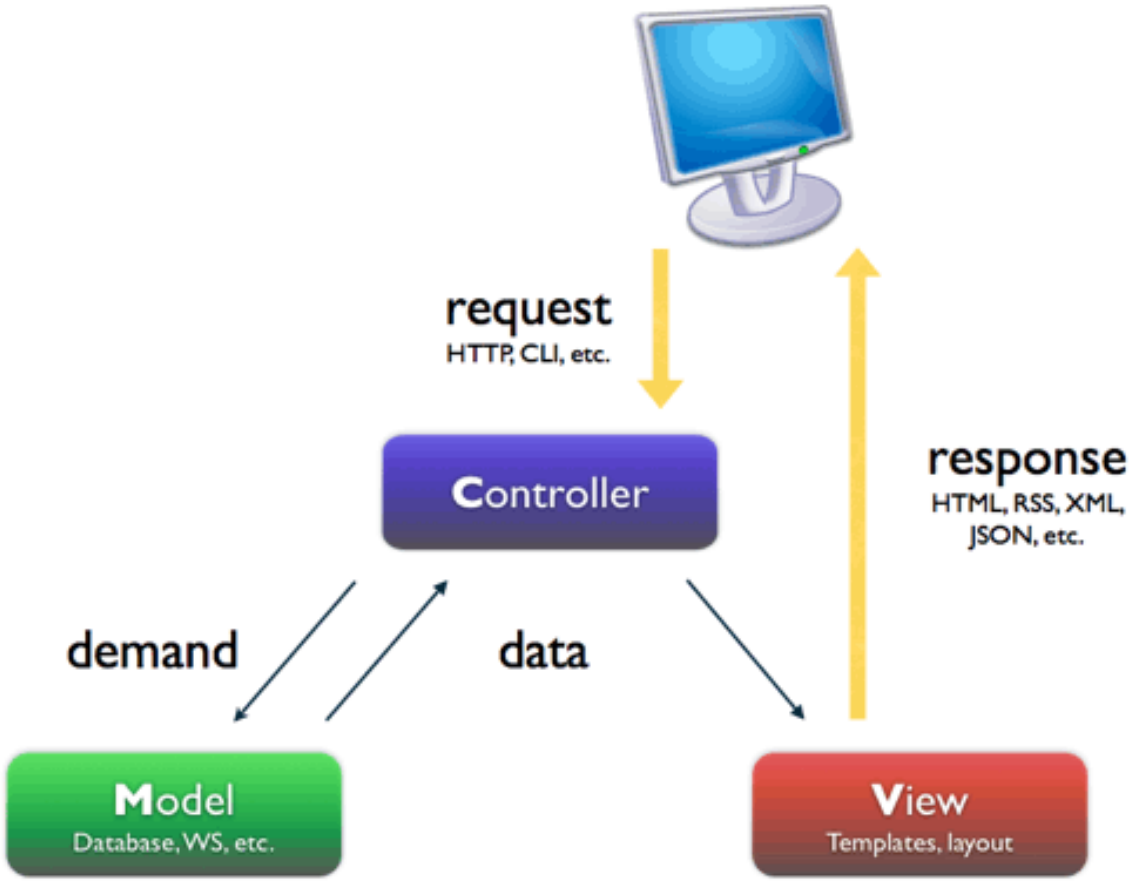
-> views.py 函数处理数据，
并将数据渲染至 templates

-> templates 返回 HTTP Response



6 以工程视角切入

还缺少哪部分?



6.1 Django ORM

Django ORM(Object-Relational Mapper)

- ORM：对象关系映射，用于实现面向对象编程里用于不同类型系统的数据之间的转换。
- Django ORM 用于 Django 与任一关系型数据库如 MySQL, MongoDB, PostgreSQL, SQLite 等进行数据交互。
- “虚拟对象数据库” => “一种封装”

6.1 Django ORM

关系型数据库

- 关系型数据库是由多张能互相联接的二维行列表格组成的数据库。

如：用户表

手机号	用户名	密码	性别	注册时间
13031001616	Jia	******	男	2019-05-21
13031001717	Wang	********	男	2019-02-04
13031001818	Ma	******	女	2019-05-13

6.1 Django ORM

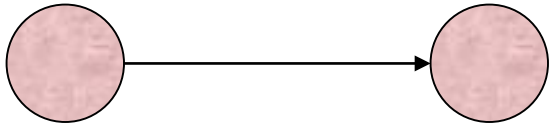
关系型数据库

- 关系型数据库是由多张能**互相联接**的二维行列表格组成的数据库。

如：数据库拥有用户表和问题表，每个用户可以提出多个问题。如何设计表格?

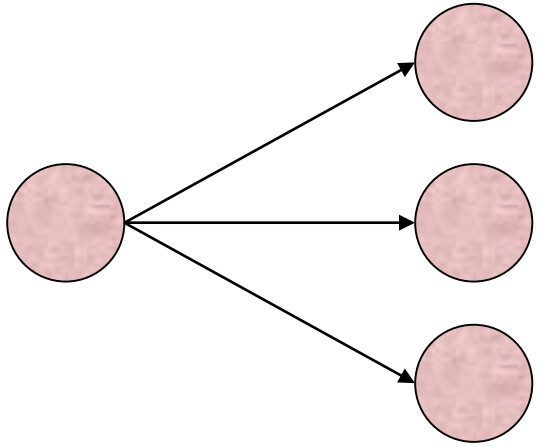
6.1 Django ORM

数据库的三种实体关系



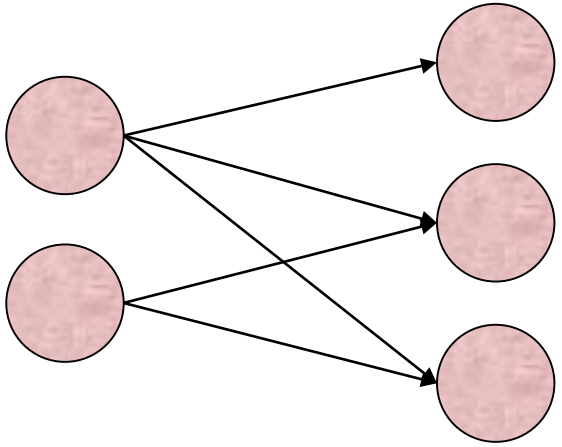
一对一关系

每位中国人民
对应一个身份证号



一对多关系

一个选择题目
对应多个选项

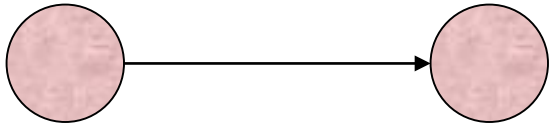


多对多关系

每位学生选择多门课程
每门课程对应多位学生

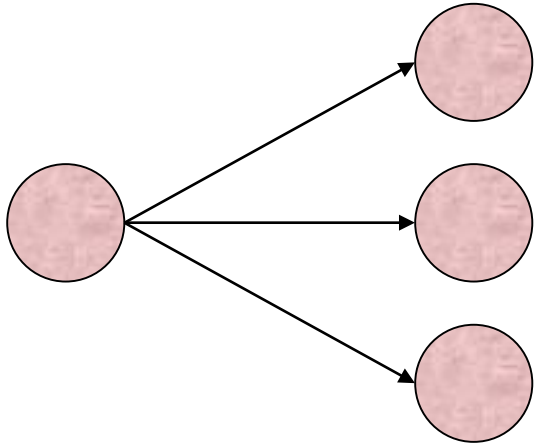
6.1 Django ORM

数据库的三种实体关系



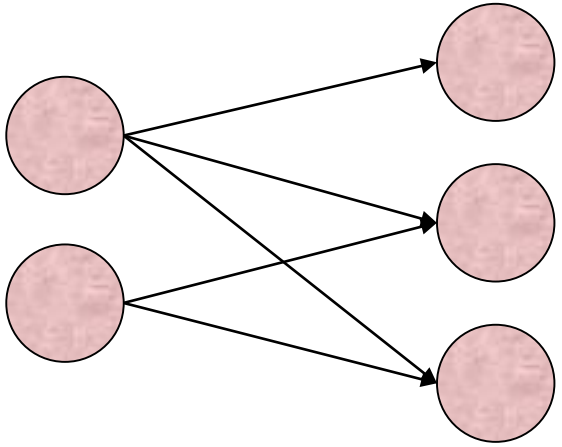
一对一关系

OneToOneField



一对多关系

ForeignKey



多对多关系

ManyToManyField

6.1 Django ORM

关系型数据库

- 关系型数据库是由多张能**互相联接**的二维行列表格组成的数据库。

如：数据库拥有用户表和问题表，每个用户可以提出多个问题。如何设计表格?

手机号	用户名	密码	性别	注册时间
18518677664	Wang	******	男	2019-05-21

1:n

问题名	提问时间
你对教学内容满意吗?	2019-05-22 10:20
你对课程有何建议?	2019-05-22 10:05

6.1 Django ORM

Demo

- 投票系统的**选择题目**简单数据库设计

问题表

字段名（列）	Field Name	类型	最大长度	能否为空	缺省值
问题名称	question_text	Char	100	否	-
创建时间	time_created	DateTime	-	否	当前时间

选项表

字段名（列）	Field Name	类型	最大长度	能否为空	缺省值
对应问题	question	ForeignKey	-	否	-
选项内容	answer_text	Char	100	否	-
投票数量	votes	Integer	-	否	0

6.1 Django ORM

Demo Django ORM Fields: https://docs.djangoproject.com/en/2.2/ref/models/fields/

- 投票系统的**选择题目**简单数据库设计 **=> Django ORM Coding**

Fields

- Django ORM 提供很多作为类实现的字段类型。如字符串：CharField，时间：DateTimeField，日期：DateField.

问题表

字段名（列）	Field Name	类型	最大长度	能否为空	缺省值
问题名称	question_text	Char	100	否	-
创建时间	time_created	DateTime	-	否	当前时间

6.1 Django Database API

Demo Django Database API: https://docs.djangoproject.com/en/2.2/topics/db/queries/

- **Django ORM Coding => 填充并操作数据库? Database API**

增删改查

- 查询: Question.objects.filter(question_text='你对课程有何建议? ')
  all()
- 增加: Question.objects.create(…)
- 修改: question = Question.objects.filter(…).first()
  question.question_text = '今天中午吃什么? '
  question.save()
- 删除: Question.objects.filter(…).delete()

6.2 Django Views 响应并处理请求

Demo Handling HTTP Response: https://docs.djangoproject.com/en/2.2/topics/http/views/

- Database API 操纵数据库 **=> 在 Web 中发送、响应请求，在 View 中操纵数据库**

① 编写 Templates 前端样式

- 编写表单并发送 HTTP Request 至 Django.

② 编写 Views 业务逻辑

- 在 views.py 中编写对应 HTTP Response 业务逻辑。操纵数据库并返回消息。

③ 编写 URLs 映射关系

- urls.py -> views.py.

6.3 Django Forms

- **在 Web 中发送和响应请求，在 View 中操纵数据库**

问题表

字段名（列）	Field Name	类型	最大长度	能否为空	缺省值
问题名称	question_text	Char	100	否	-
创建时间	time_created	DateTime	-	否	当前时间

数据库已经定义两种字段

- 需要用户通过表单输入的字段（如问题名称）
- 其他说明用户输入字段的字段（如创建时间）

Templates 又定义一遍用户表单输入字段

Views 又完整处理一遍用户表单输入字段

6.3 Django Forms

Demo Forms API: https://docs.djangoproject.com/en/2.2/ref/forms/api/

- **在 Web 中发送和响应请求，在 View 中操纵数据库**

问题表

字段名（列）	Field Name	类型	最大长度	能否为空	缺省值
问题名称	question_text	Char	100	否	-
创建时间	time_created	DateTime	-	否	当前时间

使用 Django Form 封装，直接实现：

- 模板中表单的渲染。
- 数据的验证工作，某一些输入不合法也不会丢失已经输入的数据。
- Formset 可定义多个表单集合

6.4 Generic Views

Demo Generic Views API: https://docs.djangoproject.com/en/2.2/ref/class-based-views/

- **在 Web 中发送和响应请求，在 View 中操纵数据库**

使用Generic Views 轻松处理特定功能的逻辑，主要包括四类

- 基础类：TemplateView/RedirectView
- 展示类：ListView/DetailView
- 编辑类：FormView/UpdateView/CreateView/DeleteView
- 日期类：ArchiveIndexView/YearArchiveView…

6.4 Generic Views

FormView 处理表单