混合微服务 vs Django微服务

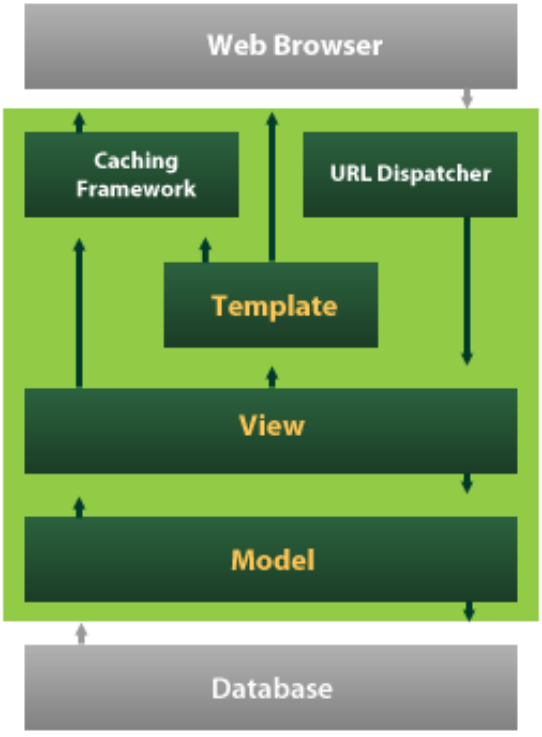
Posted by: Phodal Huang (/blog/author/root/) May 25, 2015, 11 p.m.

在设计所谓的"Next-Generation CMS"，即Echoes CMS (https://github.com/phodal/echoes)的时候，对于我这种懒得自己写Django App的人来说，通过我会去复制别人的代码，于是我继续在Github上漫游。接着找到了DjangoProject.com的源码，又看了看Mezzanine(ps: 我博客用的就是这个CMS)。于是从DjangoProject复制了Blog的代码，从Mezzanine复制了conf的代码，然后就有了Echoes (https://github.com/phodal/echoes)的codebase。然后，继之前的文章(《微服务的小思考》 (http://www.phodal.com/blog/think-about-microservices/)我想了想， 这不就是我想要的模型么?

微服务与Django (http://www.phodal.com/blog/django-application-with-hybrid-microservices/)

Django 应用架构

Django MVC结构如下如示:

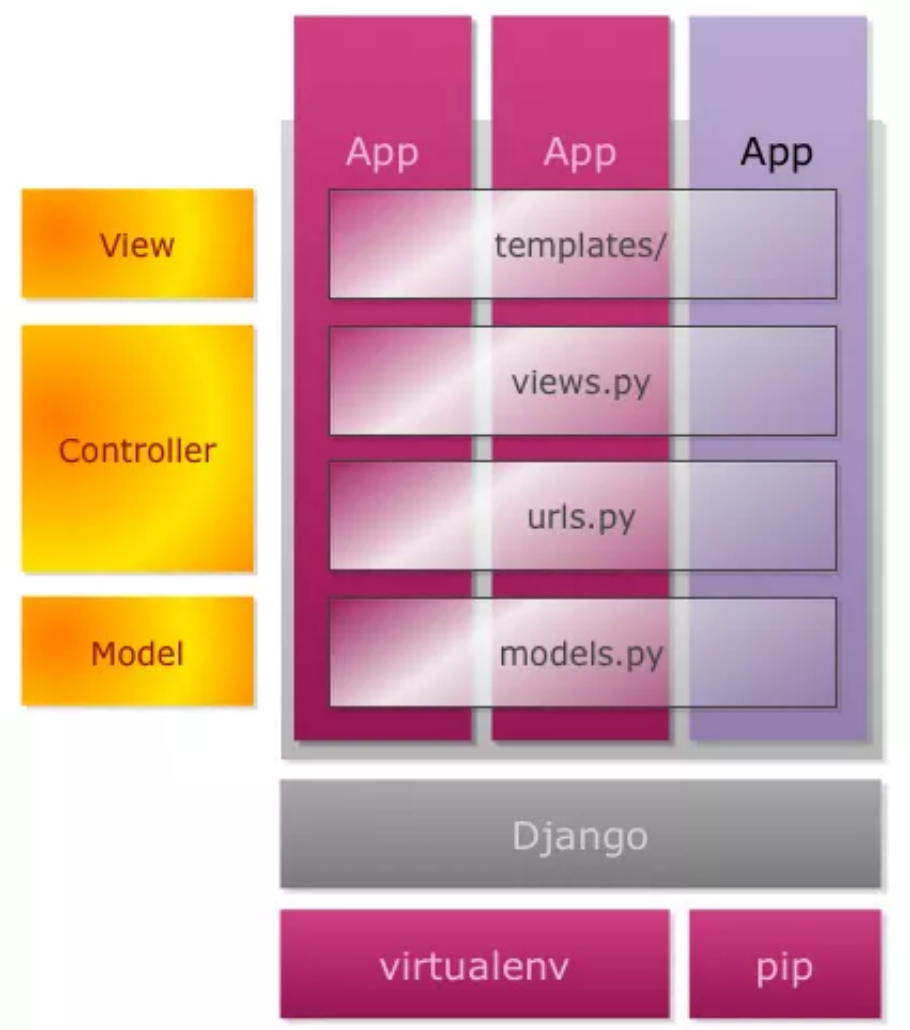


然后，记住这张图，忘记上面的MVC，Django实际上是一个MTV

- Model
- Template
- View

主要是Django中的views.py通常是在做Controller的事。

然而对于一个Django的应用来说，他的架构如下所示:



Django的每个App就代表着程序的一个功能。每个App有自己的models、views、urls、templates所以对于一个app来说他的结构如下:

```
.
|_____init__.py
|____models.py
|____tests.py
|____views.py
```

如果是新版的Django那么它的结构如下:

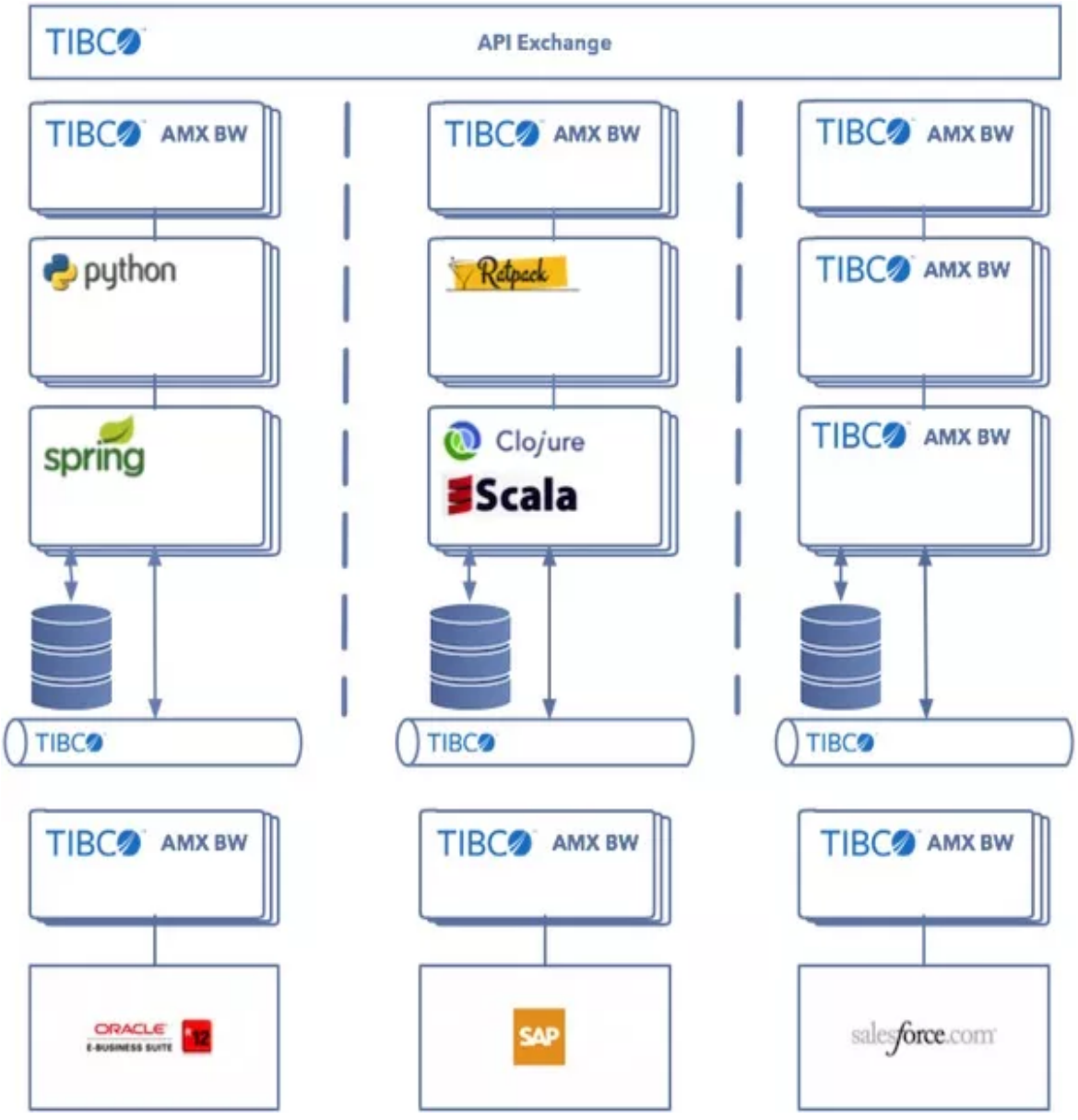
```
.
|      init   py
```

```
|_____init__.py
|____admin.py
|____migrations
| |_____init__.py
|____models.py
|____tests.py
|____views.py
```

上面少了templates，最后会有一个总的URL，即第一张图的URL Dispatcher。接着，让我们看看微服务是怎样的。

微服务

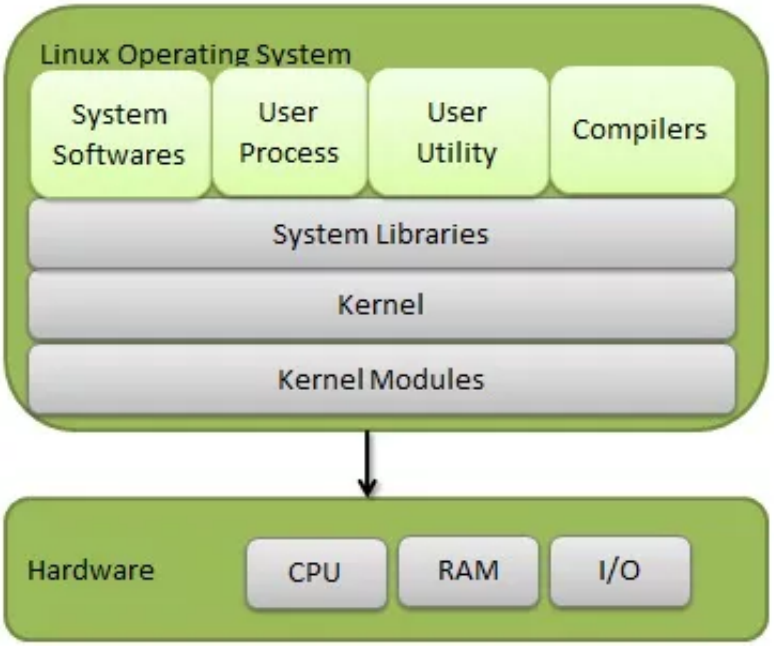
一个典型的微服务如下所示:



有不同的技术栈python、spring、scala，但是他们看上去和Django应用的图差不多，除了数据库不一样。

混合微服务

与其将复杂的测试、逻辑部分变得不可测，不如把这些部分放置于系统内部。



当我们在我们的服务器上部署微服务的时候，也就意味着实现所以的服务都是在我们系统的内部，我们有一个Kernel以及他们的Kernel Moduels，即微服务群们。他们调用DB，或者某些第三方服务。

System Libraries相当于我们的URL Dispatcher。而我们的URL Dispatcher实际上所做的便是将各自调用的服务指向各自的app。

这样我们即可以解决部署的问题，又可以减少内部耦合。

其他

> 我猜，微服务的流行是因为程序员可以欢乐地使用自己的语言，哪怕是Logo。