

提纲

- Django简介
- Django环境搭建
- Django环境使用
- Django创建Blog网站
- 小结

提纲

- **Django简介**
- **Django环境搭建**
- **Django环境使用**
- **Django创建Blog网站**
- **小结**

Web框架

■ Django

- Django 的命名是在一个叫做Django Reinhardt的吉普赛吉他手之后，被认为是最好的吉他手。
- Django 的发音是“JANG-oh酱呕” 和“FANG-oh 放呕”的发硬押韵。首字母D不发音。

历史

- 许多年前，劳伦斯出版集团需要开发一个以**新闻**内容为主的网站。
- 众所周知，对于新闻网站来说，**需求变化很快，互动性也很高。**
- 于是，world online的三位工程师使用python开发了**Django**。



为什么要快速开发

- 如果我们只考虑**时间代价**，时间**短**往往意味着客户更**满意**。



Time

为什么要快速开发

- 并且你还可以有余力开发第二个版本



Time

设计哲学

- Django的主要目的是**简便、快速**地开发**数据库驱动**的网站——**动态**网站。
- Django强调**代码复用**，多个组件可以方便地以“插件”形式服务于整个框架，Django有许多功能强大的**第三方插件**。
- Django强调**快速开发**。
- 基于**MVC**（更确切的说是MTV）

为什么是python

- 可能是一下几点：
 - 面向对象编程（Object Oriented Programming）
 - module机制，松耦合，模块插入方便
 - 代码简洁
 - 功能强大，模块多
- 总结起来：python技术就是django技术

Django应用

- 最著名的当然就是**Google AppEngine**基于Django，Django应用可以很方便地在它上面部署。
- 可以用Eclipse、PyCharm等进行开发。

动态网站

- 用户向web服务器**请求**一个文档
- Web服务器随即**获取或生成**这个文档
- 服务器再把**结果返回**给浏览器
- 浏览器将这个文档**渲染**出来

Django

■ MVC 设计模式

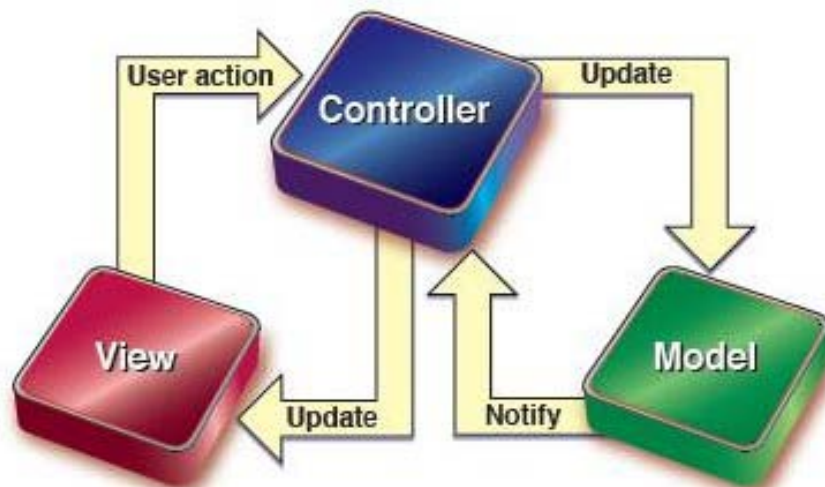
- 模型M
- 视图V
- 控制器C

Django 包含了很多应用在它的“contrib”包中，这些包括：

- 一个可扩展的认证系统
- 动态站点管理页面
- 一组产生RSS和Atom的工具
- 一个灵活的评论系统
- 产生Google站点地图（Google Sitemaps）的工具
- 防止跨站请求伪造（cross-site request forgery）的工具
- 一套支持轻量级标记语言（Textile和Markdown）的模板库
- 一套协助创建地理信息系统（GIS）的基础框架

MVC

- MVC把web分为数据模型，控制器和视图三层，可以使业务逻辑与数据表现分开；
- 说白了，美工搞美工的，后台搞后台的，互不干扰，发挥各自优势



Django是MTV分层

- Django遵循了MVC（model-view-controller）这个分层方式，但是确切的说
是MTV分层。
- M→model，数据模型
- T→template，模板
- V→view，视图

Django是MTV分层

■ Model层

- 使用的是（ORM对象关系映射, Object Relational Mapping)
- 我们所能控制的就是models.py文件
- 负责数据库管理

Django是MTV分层

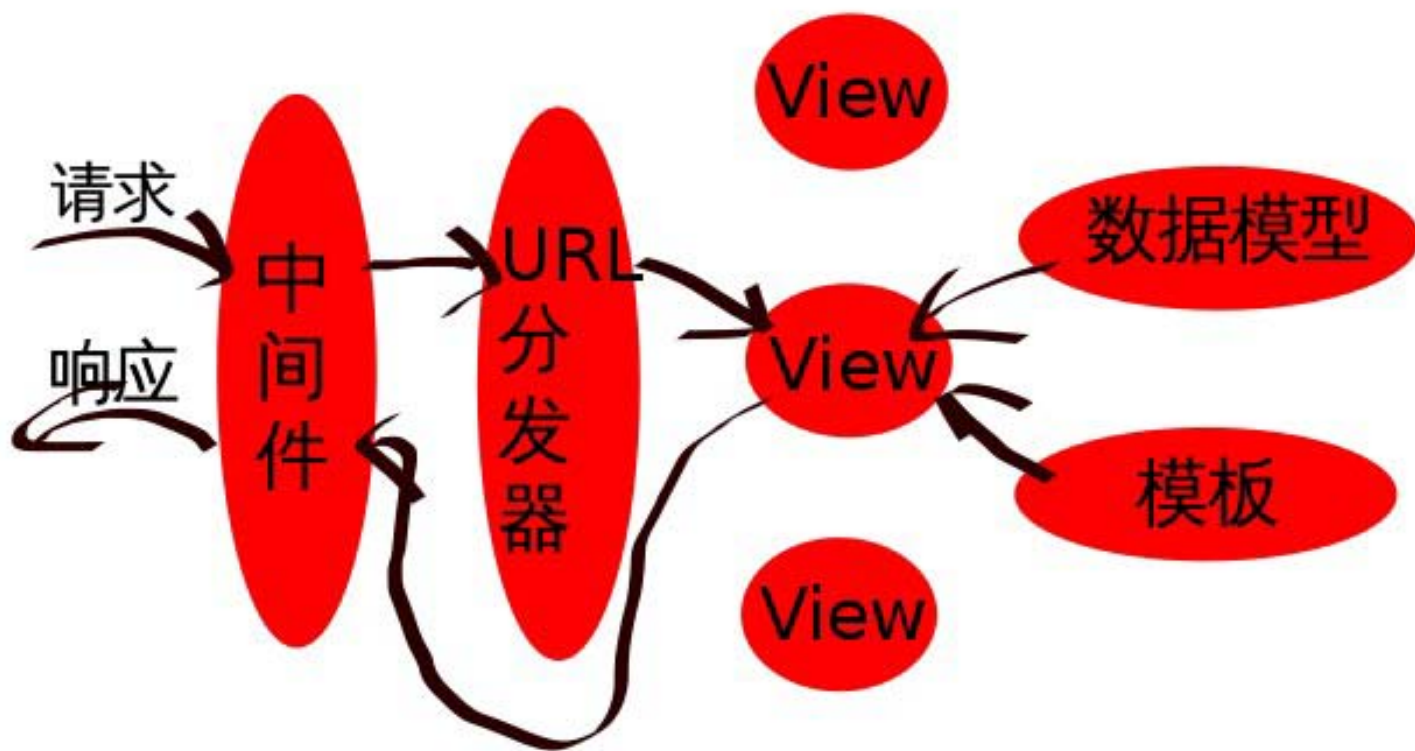
- **Template层负责怎么样显示数据**
 - 利用一些**格式化的html文件**，使数据按照要求显示（显示在哪里，怎么显示等等）；
 - 需要在工程目录下，建立**templates文件夹**，然后在**setting.py**中设置好**templates目录**的路径。然后在此目录中建立所需**html文件**。

Django是MTV分层

■ View层

- ❑ django中的view层是用于控制要显示什么数据
- ❑ 我们能看到的就是views.py文件
- ❑ views.py（可以是任意名字）

Django的过程



Django的过程

- Web服务器收到一个http请求;
- Django把web服务器传过来的请求转换成一个请求对象;
- Django在URLconf里查找正确的视图函数;
- 调用这个视图函数, 参数为请求对象以及任何捕捉到的URL参数;
- 然后视图会创建并返回一个响应对象;
- Django将这个响应对象转换成web服务器可以理解的格式;
- Web服务器将响应发送给客户端。

Django一些可重用的模块

- Django提供了很多可重用的模块;
- Django的modules机制是松耦合的, 也就是说你可以很方便的插入这些模块;
- 减少了多余的编写代码工作。

Django一些可重用的模块

django-ratings	django-ajax-validation	django-google-analytics	django-mailer
django-queue-service	django-announcements	django-email-confirmation	django-jits
django-liveblogging	django-atompub	django-discussion	django-galaxy
django-messages	django-audioplayer	django-db-log	django-evolution
django-authopenid	django-googlemap	django-compress	django-dynamic-media-serve
django-avatar	django-graphs	django-oembed	django-clevercss
django-basic-blog	django-microformats	django-object-view-tracking	django-chunks
django-basic-library	django-tagging	django-navbar	django-ads
django-basic-people	django-survey	django-orm-cache	django-rest-interface
django-basic-places	django-voting	django-page-cms	django-registration
django-cron	django-wiki	django-photologue	django-mobileadmin
django-favorites	satchmo	django-pingback	django-openid
django-forum	sorl-thumbnail	django-pressroom	django-oauth
django-gcal	django-mailfriend	django-mmo	django-recommender

预备知识

- Python基础知识
- **Html和css**，其他前端技术知道更好
- **数据库**
- **正则表达式**

提纲

- Django简介
- Django环境搭建
- Django环境使用
- Django创建Blog网站
- 小结

Django Linux安装使用

■ 安装（以Linux为例）

□ rpm -ivh

http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm

□ yum install -y Django14

■ 使用步骤:

□ 创建项目 `django-admin startproject` **X**

□ 创建应用 `django-admin startapp` **Y**

Django Window安装

■ 安装MySQL

<http://dev.mysql.com/downloads/mysql/>

■ 安装Django

□ 下载地址:

<https://www.djangoproject.com/download/>

□ 直接将下载的Django-1.*.*.tar.gz解压，在cmd中进入该目录，输入：**python setup.py install**

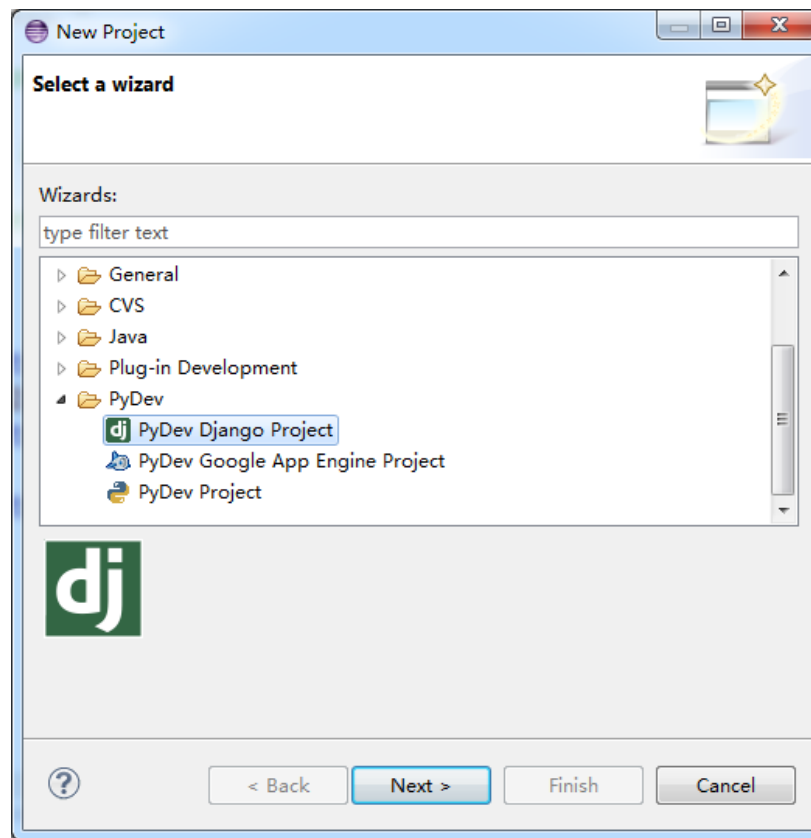
□ 安装完成后:

```
Installed c:\users\zhou\anaconda\lib\site-packages\django-1.8-py2.7.egg
Processing dependencies for Django==1.8
Finished processing dependencies for Django==1.8
D:\django>
半:
```

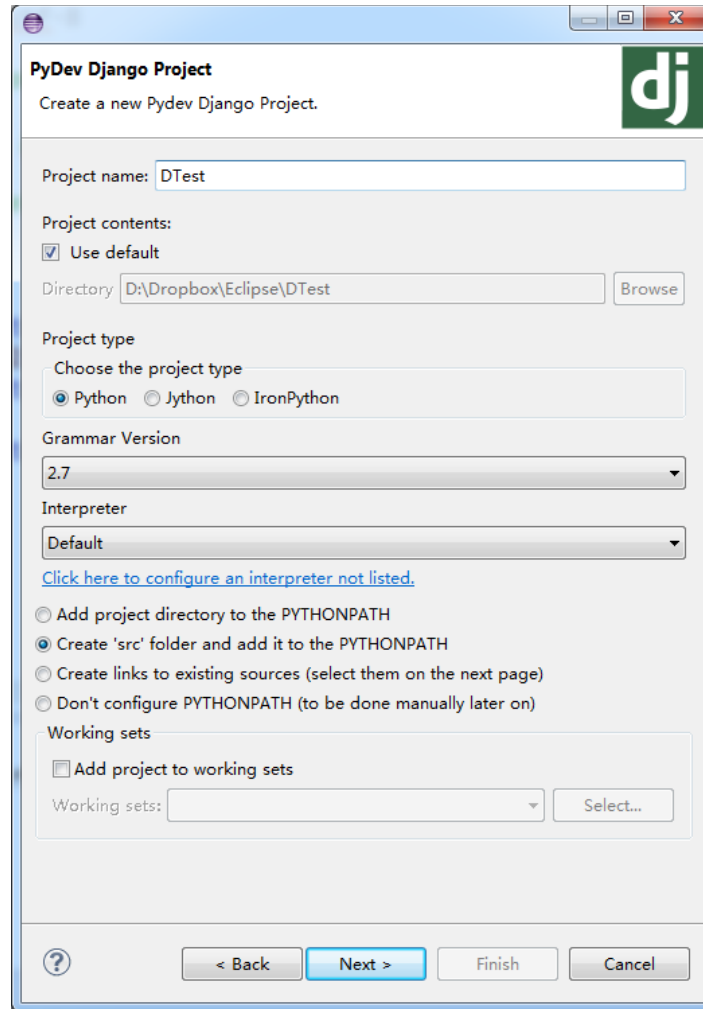
```
>>> import django
>>> django.get_version()
'1.8'
>>>
```


在eclipse中搭建Django

■ 创建项目 New->Project



在eclipse中搭建Django



The image shows the 'PyDev Django Project' dialog box in Eclipse. It is titled 'PyDev Django Project' and has a subtitle 'Create a new Pydev Django Project.' with a Django logo. The dialog contains several sections: 'Project name' with a text field containing 'DTest'; 'Project contents' with a checked 'Use default' checkbox and a 'Directory' field containing 'D:\Dropbox\Eclipse\DTest' with a 'Browse' button; 'Project type' with a 'Choose the project type' section containing three radio buttons: 'Python' (selected), 'Jython', and 'IronPython'; 'Grammar Version' with a dropdown menu set to '2.7'; 'Interpreter' with a dropdown menu set to 'Default' and a link 'Click here to configure an interpreter not listed.'; a section with four radio buttons for PYTHONPATH configuration, where 'Create 'src' folder and add it to the PYTHONPATH' is selected; and 'Working sets' with an unchecked 'Add project to working sets' checkbox and a 'Working sets:' dropdown with a 'Select...' button. At the bottom are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

PyDev Django Project
Create a new Pydev Django Project.

Project name: DTest

Project contents:
☒ Use default
Directory: D:\Dropbox\Eclipse\DTest Browse

Project type
Choose the project type
☒ Python ☐ Jython ☐ IronPython

Grammar Version
2.7

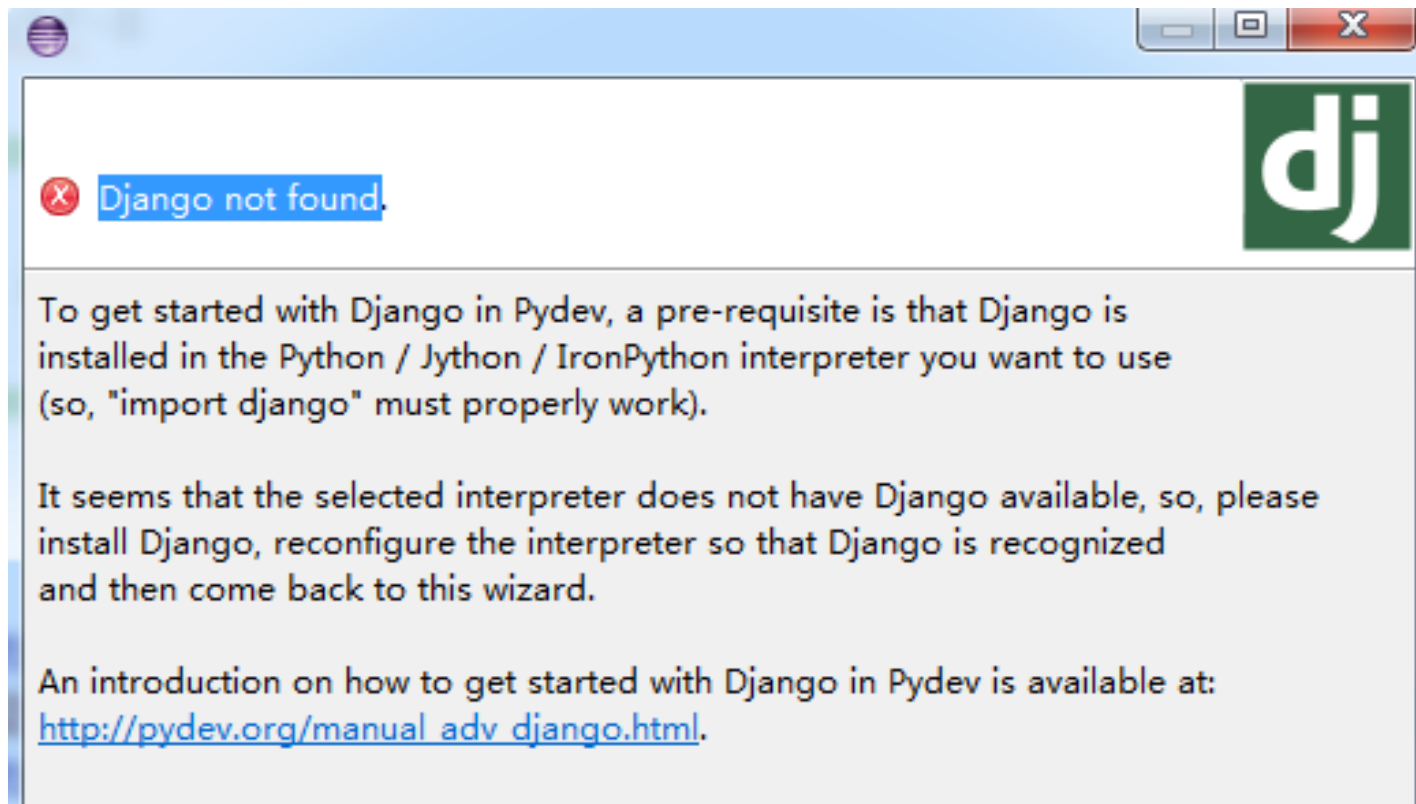
Interpreter
Default
[Click here to configure an interpreter not listed.](#)

☐ Add project directory to the PYTHONPATH
☒ Create 'src' folder and add it to the PYTHONPATH
☐ Create links to existing sources (select them on the next page)
☐ Don't configure PYTHONPATH (to be done manually later on)

Working sets
☐ Add project to working sets
Working sets: Select...

? < Back Next > Finish Cancel

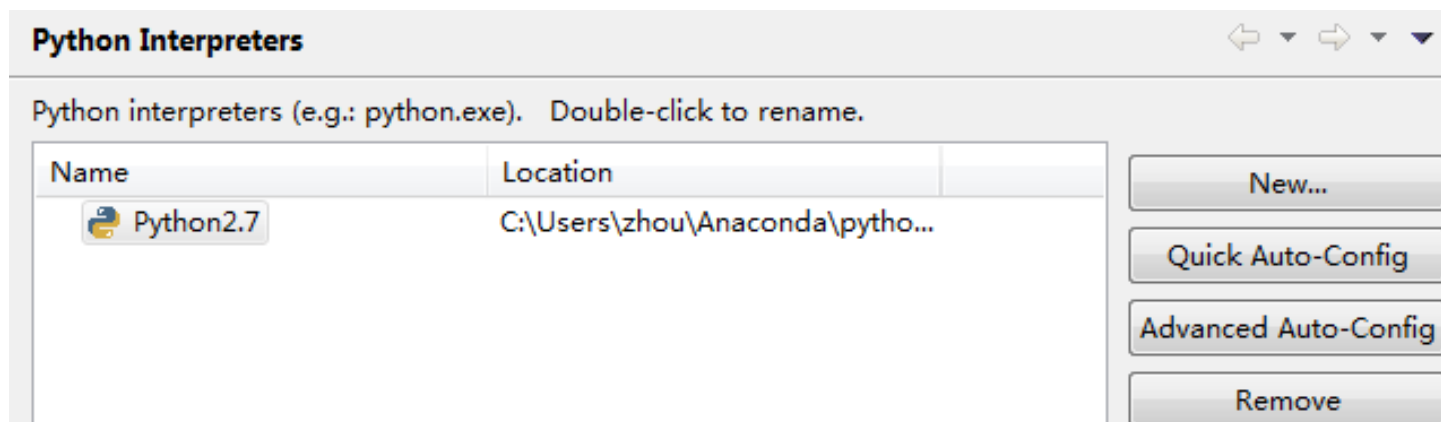
在eclipse中搭建Django



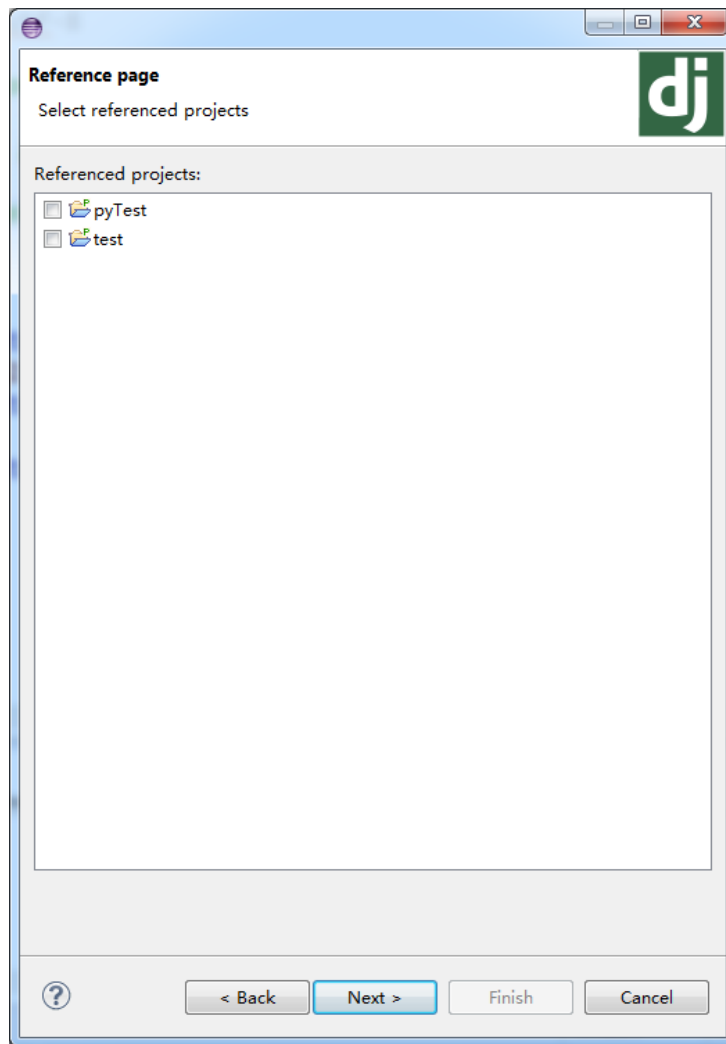
在eclipse中搭建Django

■ Django not found解决方案

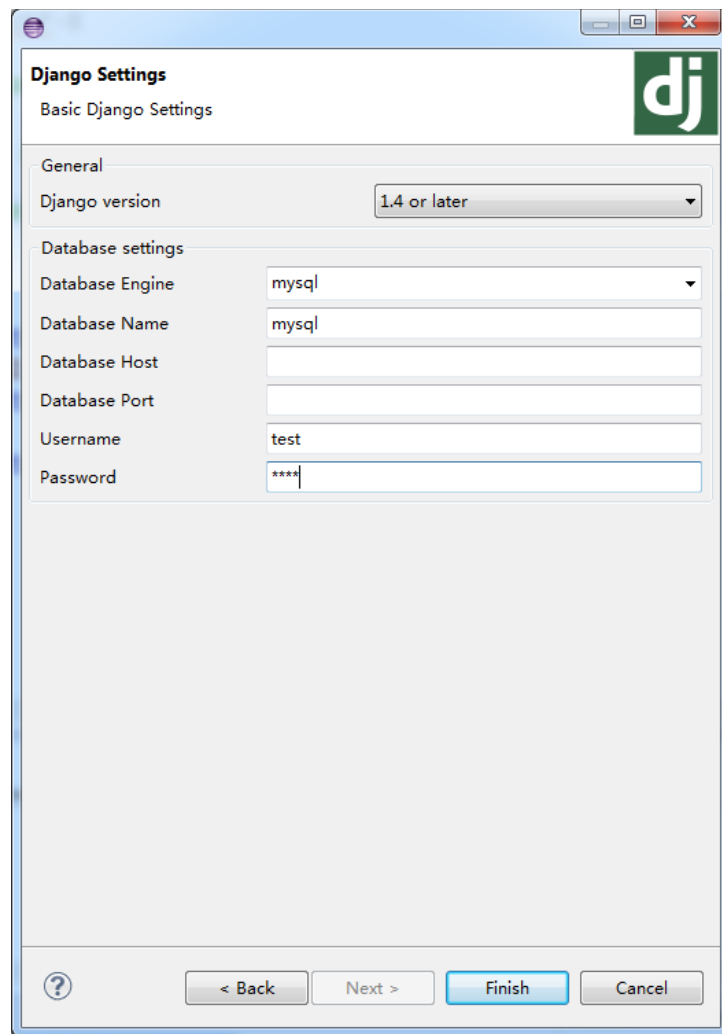
- ❑ 这是因为Python在Django安装之前已经安装，从而Pydev不能找到django所在的路径。
- ❑ 解决方法：点击Eclipse->window->首选项->Pydev->Interpreter-Python，把解释器删除重新配置。



在eclipse中搭建Django



在eclipse中搭建Django



The image shows a 'Django Settings' dialog box from the Eclipse IDE. The dialog has a title bar with standard window controls. Inside, the title 'Django Settings' is followed by a subtitle 'Basic Django Settings' and a green 'dj' logo. The settings are organized into two sections: 'General' and 'Database settings'. In the 'General' section, the 'Django version' is set to '1.4 or later'. The 'Database settings' section includes fields for 'Database Engine' (set to 'mysql'), 'Database Name' (set to 'mysql'), 'Database Host', 'Database Port', 'Username' (set to 'test'), and 'Password' (masked with '****'). At the bottom, there is a help icon, a '< Back' button, a 'Next >' button, a 'Finish' button, and a 'Cancel' button.

Django Settings
Basic Django Settings

General

Django version: 1.4 or later

Database settings

Database Engine: mysql

Database Name: mysql

Database Host:

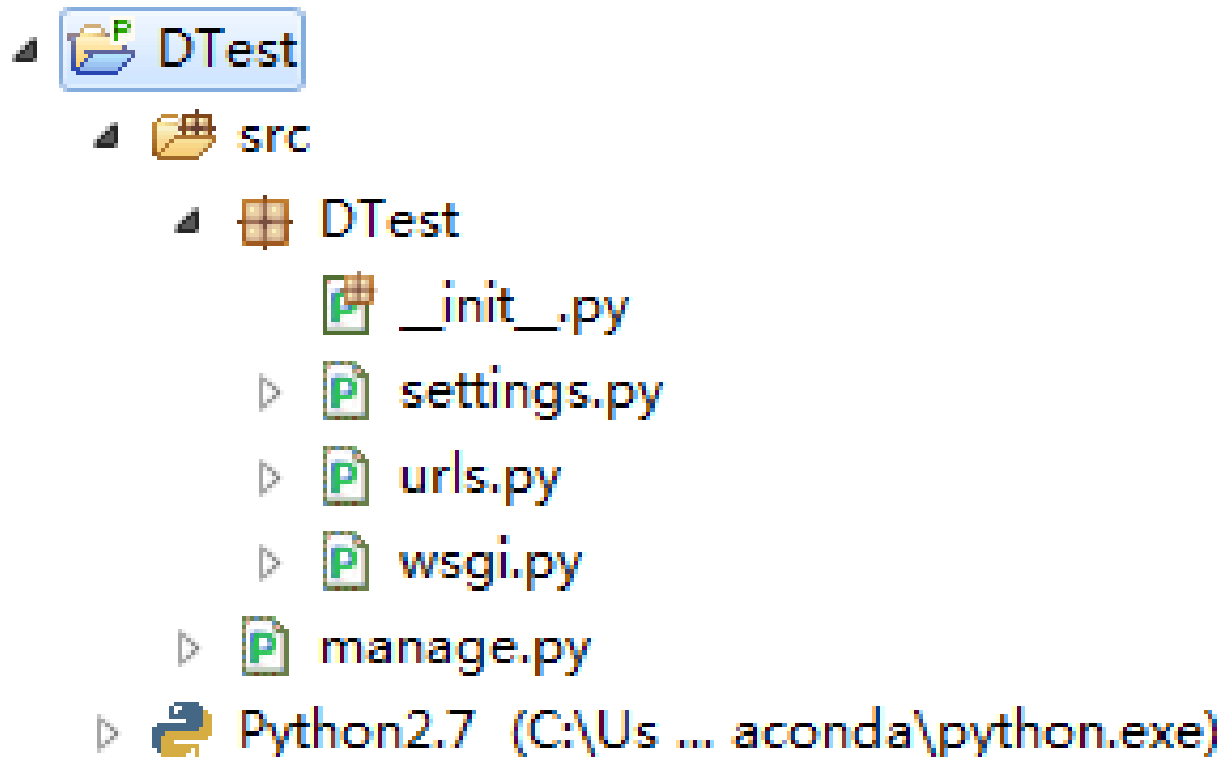
Database Port:

Username: test

Password: ****

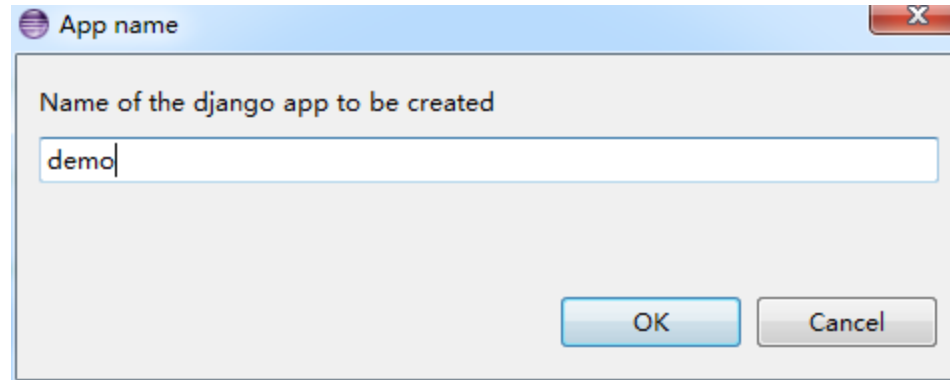
? < Back Next > Finish Cancel

在eclipse中搭建Django

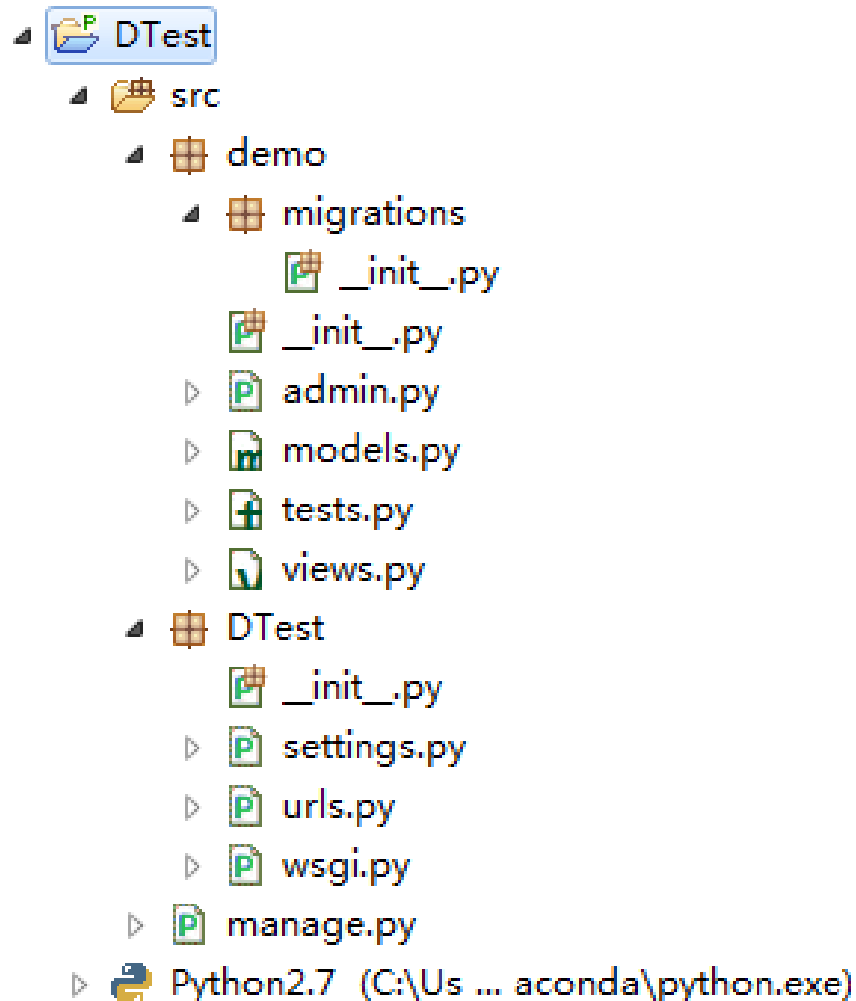


在eclipse中搭建Django

- 使用eclipse创建一个app, 在项目名字那里点击右键, 选到Django那一项-->
Create application(manage.py start app)
- 设置名字为demo



在eclipse中搭建Django



在eclipse中搭建Django

- 在views.py中写下面几行代码

```
3 # Create your views here.  
4 from django.http import HttpResponse  
5  
6 def hello(request):  
7     return HttpResponse('<h1>Hello Django!!!</h1>')
```

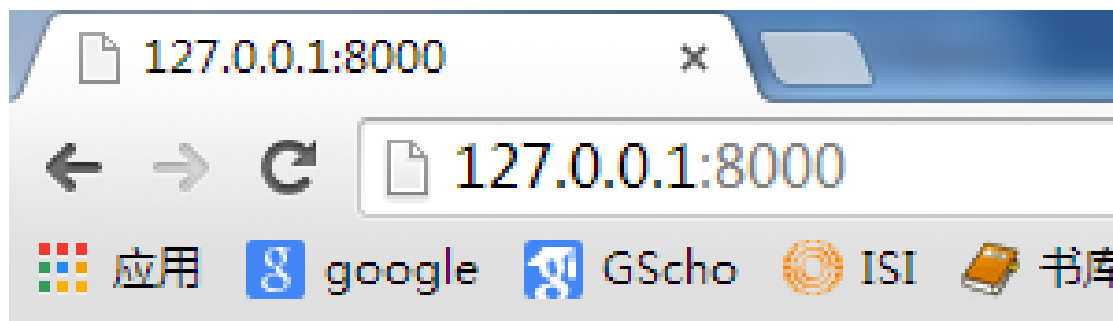
在eclipse中搭建Django

■ urls.py里面加一句代码

```
4 urlpatterns = [  
5     # Examples:  
6     # url(r'^$', 'DTest.views.home', name='home'),  
7     # url(r'^blog/', include('blog.urls')),  
8  
9     url(r'^admin/', include(admin.site.urls)),  
10    url(r'^$', 'demo.views.hello'),  
11 ]
```

在eclipse中搭建Django

- 最后运行项目（Run As-->Pydev :Django）
- 运行效果为：



Hello Django!!!

在 Django 中使用 MySQL

- Django要求MySQL4.0或更高的版本。3.X版本不支持嵌套子查询和一些其它相当标准的SQL语句。
- 下载安装 MySQLdb
 - <http://www.codegood.com/archives/129>
 - <http://www.djangoproject.com/r/python-mysql/>
。

数据库设置

- 编辑 **DTest/settings.py** 。
 - 这是一个普通的Python模块，包含了代表Django设置的模块级变量。更改**DATABASES**中‘**default**’下的以下键的值，以匹配您的数据库连接设置。
 - **ENGINE** – 从
 - ‘django.db.backends.postgresql_psycopg2’,
 - ‘django.db.backends.mysql’,
 - ‘django.db.backends.sqlite3’,
 - ‘django.db.backends.oracle’ 中选一个

数据库设置

- **NAME** – 你的数据库名。当指定路径时，总是使用正斜杠，即使是在Windows下(例如：``C:/homes/user/mysite/sqlite3.db``)。
- **USER** – 你的数据库用户名 (SQLite 下不需要)。
- **PASSWORD** – 你的数据库密码 (SQLite 下不需要)
- HOST** – 你的数据库主机地址。如果和你的数据库服务器是同一台物理机器，请将此处**保留为空** (或者设置为 `127.0.0.1`) (SQLite 下不需要)。
- **SQLite** 是内置在 Python 中的，因此你不需要安装任何东西来支持你的数据库。

数据库设置

```
61 DATABASES = {
62     'default': {
63         'ENGINE': 'django.db.backends.mysql',
64         'NAME': 'dtest', # os.path.join(BASE_DIR, 'db.sqlite3'),
65         # The following settings are not used with sqlite3:
66         'USER': 'root',
67         'PASSWORD': 'test',
68         'HOST': '127.0.0.1', # '127.0.0.1' for localhost through TCP.
69         'PORT': '3306', # Set to empty string for default.
70     }
71 }
```


提纲

- Django简介
- Django环境搭建
- Django环境使用
- Django创建Blog网站
- 小结

Django环境使用

■ 所需编程知识

- 需要理解基本的**面向过程和面向对象编程**:
 - 流程控制（if，while 和 for），数据结构（列表，哈希表/字典），变量，类和对象。
- **Web开发经验**，非常有帮助，但是不是必须的。
- **Python所需知识**
 - Django 只不过是用 **Python 编写的一组类库**。用 Django 开发站点就是使用这些类库编写 Python 代码。
 - 学习 Django 的关键就是**学习如何进行 Python 编程并理解 Django 类库的运作方式**。
 - 对你来说，学习Django就是学习她的**命名规则和API**。

第一份视图

- Django对于view.py的文件命名没有特别的要求，它不在乎这个文件叫什么。但是根据约定，把它命名成view.py，这样有利于其他开发者读懂你的代码。

- 我们的视图非常简单。这些是完整的函数和导入声明，你需要输入到views.py文件：

```
from django.http import HttpResponse
def hello(request):
    return HttpResponse("Hello Django!!!")
```

第一份视图

- 从 `django.http` 模块导入 `HttpResponse` 类

```
4 from django.http import HttpResponse
5
6 def hello(request):
7     return HttpResponse('<h1>Hello Django!!!</h1>')
```

- 每个视图函数至少要有有一个参数，通常被叫作 **request**。这是一个触发这个视图、包含当前Web请求信息的对象，是类 `django.http.HttpRequest` 的一个实例。
- 在这个示例中，我们虽然不用 `request` 做任何事情，然而它仍必须是这个视图的第一个参数。

第一份视图

- 将urls.py修改为:

```
4 urlpatterns = [  
5     # Examples:  
6     # url(r'^$', 'DTest.views.home', name='home')  
7     # url(r'^blog/', include('blog.urls')),  
8  
9     url(r'^admin/', include(admin.site.urls)),  
10    url(r'^$', 'demo.views.first_page'),  
11 ]
```

- 修改了最后一行。它将根目录的URL分配给一个对象进行处理，这个对象是 **mysite.views.first_page**。

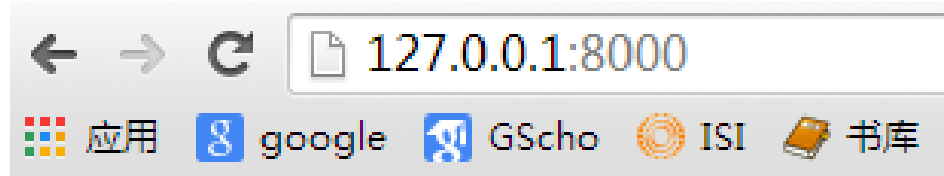
第一份视图

- 用以处理HTTP请求的**这一对象还不存在**，我们在views.py中定义first_page函数：

```
1 #coding=utf-8
2 from django.shortcuts import render
3
4 # Create your views here.
5 from django.http import HttpResponse
6
7 def hello(request):
8     return HttpResponse('<h1>Hello Django!!!</h1>')
9
10 def first_page(request):
11     return HttpResponse("<p>第一个网页</p>")
```

第一份视图

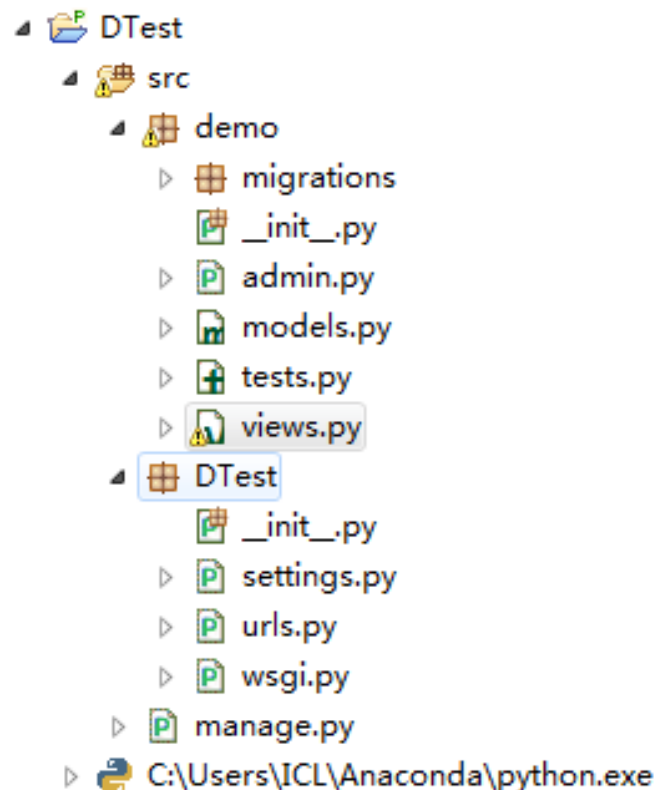
- 第一行说明字符编码为utf-8，为下面使用中文做准备。
- `first_page`函数的功能，是返回http回复，即这里的<p>第一个网页</p>。
- `first_page`有一个参数`request`，该参数包含有请求的具体信息，比如请求的类型等，这里并没有用到。



第一个网页

工程目录解析

- 外层demo目录只是你项目的一个**容器**。对于Django来说该目录名并不重要;你可以**重命名**为你喜欢的。
- **manage.py**: 一个实用的**命令行工具**, 可让你以各种方式与该Django项目进行**交互**。
- 内层demo目录是你项目中的实际Python包。



工程目录解析

- 内层demo目录名就是Python包名，通过它可以导入它里面的任何东西。
(e.g. `import demo.settings`).
- `demo /__init__.py`: 一个空文件，告诉Python该目录是一个Python包。
- `demo /settings.py`: 该Django项目的设置/配置。
- `demo /urls.py`: 该Django项目的URL声明; 一份由Django驱动的网站“目录”。
- `demo /wsgi.py`: 一个WSGI兼容的Web服务器的入口，以便运行你的项目。

数据库设置

- 编辑 `mysite/settings.py` 。
 - 这是一个普通的Python模块，包含了代表 Django 设置的**模块级变量**。更改 `DATABASES` 中‘**default**’下的以下键的值，以匹配您的数据库连接设置。
 - a、ENGINE – 从
 - ‘`django.db.backends.postgresql_psycopg2`’,
 - ‘**`django.db.backends.mysql`**’,
 - ‘`django.db.backends.sqlite3`’,
 - ‘`django.db.backends.oracle`’ 中选一个

数据库设置

- ❑ **NAME** – 你的数据库名。
 - 如果你使用 SQLite，该数据库将是你计算机上的一个文件；在这种情况下，`:setting:NAME` 将是一个完整的绝对路径，而且还包含该文件的名称。如果该文件不存在，它会在第一次同步数据库时自动创建。当指定路径时，总是使用正斜杠，即使是在 Windows 下(例如：`C:/homes/user/mysite/sqlite3.db`)
- ❑ **USER** – 你的数据库用户名 (SQLite 下不需要)
- ❑ **PASSWORD** – 你的数据库密码 (SQLite 下不需要)。

数据库设置

- ❑ **HOST** – 你的数据库主机地址。如果和你的数据库服务器是同一台**物理机器**，请将此处保留为空 (或者设置为 127.0.0.1) (SQLite 下不需要)
- ❑ 如果你是新建数据库，我们建议只**使用 SQLite**，将 **ENGINE** 改为 `'django.db.backends.sqlite3'` 并且将 **NAME** 设置为你想存放**数据库**的地方。
- ❑ **SQLite** 是内置在 **Python** 中的，因此你**不需要**安装任何东西来支持你的数据库。

MySQL数据库设置

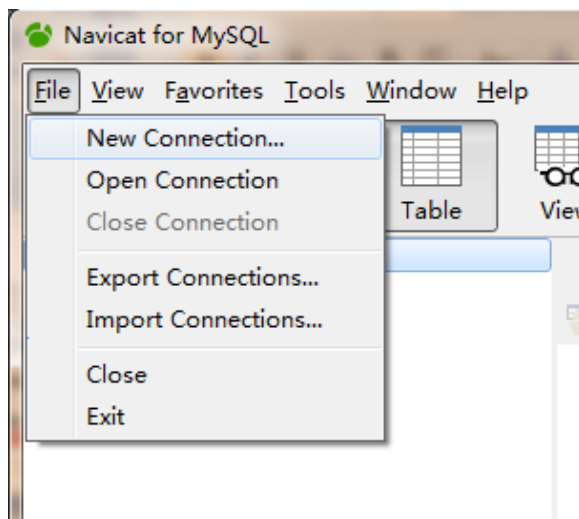
```
61 DATABASES = {
62     'default': {
63         'ENGINE': 'django.db.backends.mysql',
64         'NAME': 'dtest', # os.path.join(BASE_DIR, 'db.sqlite3'),
65         # The following settings are not used with sqlite3:
66         'USER': 'root',
67         'PASSWORD': 'test',
68         'HOST': '127.0.0.1',# '127.0.0.1' for localhost through TCP.
69         'PORT': '3306', # Set to empty string for default.
70     }
71 }
```

MySQL数据库设置

- 如果你使用MySQL，确保你已经**创建了一个数据库**。
 - 可以通过你的数据库交互接口中的“**CREATE DATABASE database_name;**”命令做到这一点的。
 - 如果你使用SQLite，你不需要事先创建任何东西。在需要的时候，将自动创建数据库文件。

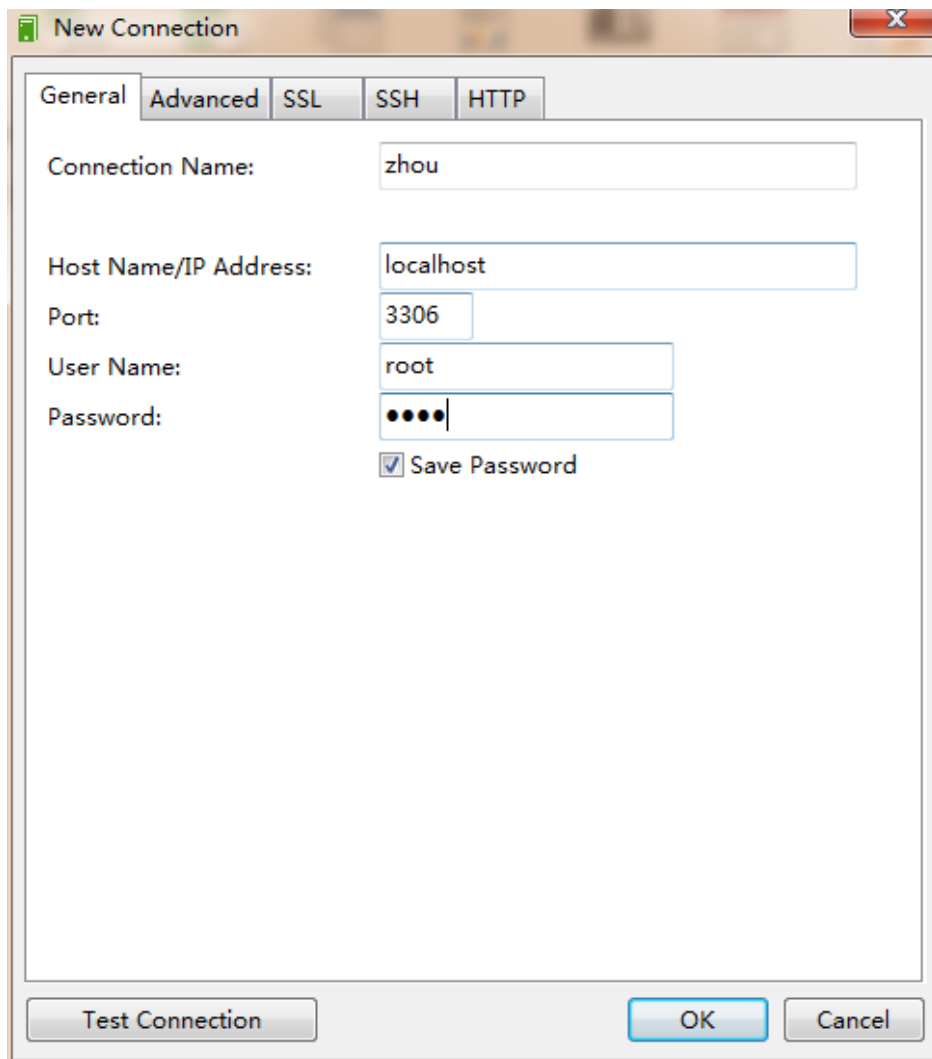
Navicat for MySQL创建数据库

- 下载安装Navicat for MySQL
 - ❑ <http://www.navicat.com.cn/products/navicat-for-mysql>
 - ❑ 可以方便的查看操作MySQL数据库中的数据。



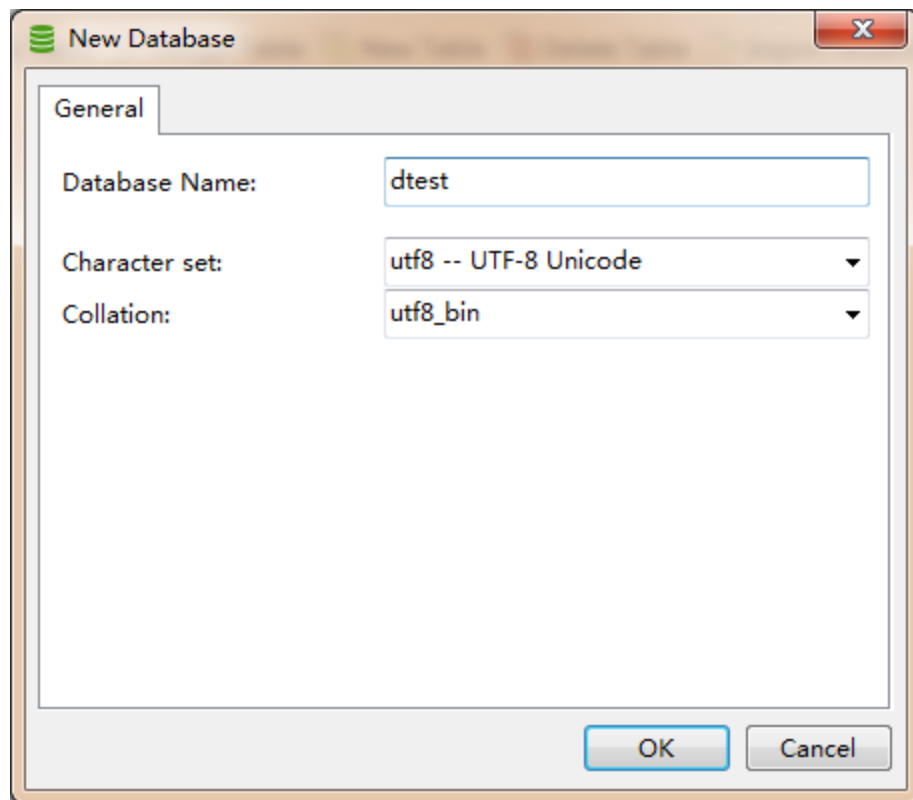
Navicat for MySQL创建数据库

- 输入**连接名**和安装MySQL时设置的**用户名和密码**，剩下的默认设置。



Navicat for MySQL创建数据库

- 鼠标右键点击连接名，选择“New database”；
- 输入数据库名称，选择字符集和校对规则。



编辑 settings.py

- 将 **TIME_ZONE** 修改为你所在的时区。

- 默认值是**美国中央时区**（芝加哥）。
- 可以改成北京时间：

```
87 TIME_ZONE = 'Asia/Shanghai'
```

- 注意文件底部的 **INSTALLED_APPS** 设置

- 它保存了当前 Django 实例已**激活**的所有 Django **应用**。
- 每个应用可以被**多个项目**使用，而且你可以**打包和分发**给其他人在他们的项目中使用。

INSTALLED_APPS应用

```
33 INSTALLED_APPS = (  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles'  
40 )
```

INSTALLED_APPS应用

- 默认情况下，INSTALLED_APPS 包含以下应用，这些都是由 Django 提供的：
 - ❑ django.contrib.admin – 数据库管理框架。
 - ❑ django.contrib.auth – 身份验证系统。
 - ❑ django.contrib.contenttypes – 内容类型框架。
 - ❑ django.contrib.sessions – session 框架。
 - ❑ django.contrib.messages – 消息框架。
 - ❑ django.contrib.staticfiles – 静态文件管理框架。
 - ❑ 这些应用在一般情况下是默认包含的。

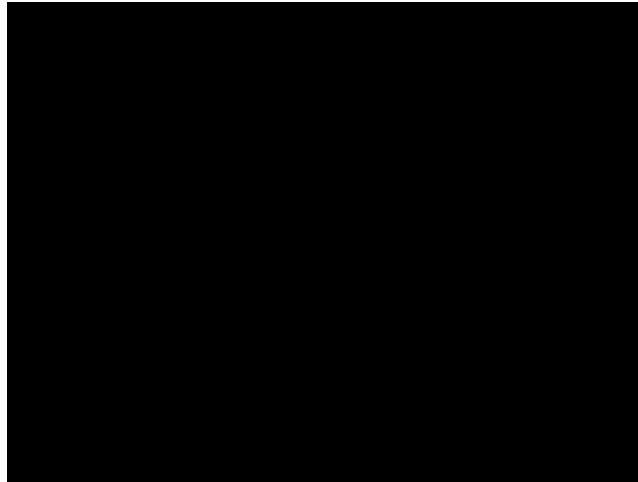
INSTALLED_APPS应用

- 所有这些应用中每个应用至少使用一个数据库表，所以在使用它们之前我们需要创建数据库中的表。
 - 要做到这一点，请运行以下命令：**python manage.py syncdb**，具体操作后面会阐述。

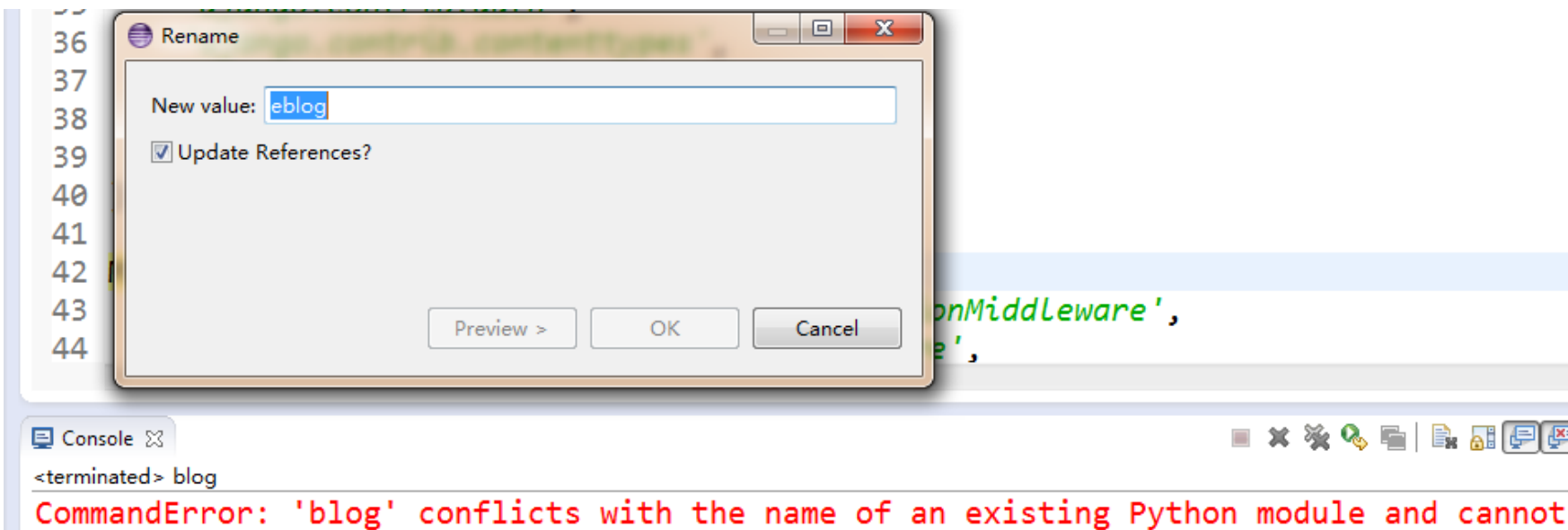
提纲

- Django简介
- Django环境搭建
- Django环境使用
- Django创建Blog网站
- 小结

创建网站模块app



创建网站模块app



修改eblog.models.py

```
models ✕  
1 from django.db import models  
2 from django.contrib import admin  
3  
4 # Create your models here.  
5 class BlogPost(models.Model):  
6     title = models.CharField(max_length = 150)  
7     content = models.TextField()  
8     timestamp = models.DateTimeField()  
9  
10 class BlogPostAdmin(admin.ModelAdmin):  
11     list_display = ('title', 'content', 'timestamp')  
12  
13 admin.site.register(BlogPost, BlogPostAdmin)
```

修改eblog.models.py

- 我们将创建一个**BlogPost模型**
 - 包含title、content、timestamp三个字段。
 - 每个模型都由继承自**django.db.models.Model**子类的类来描述。
 - 每个模型都有一些**类变量**，每一个类变量都代表了一个**数据库字段**。
 - 每个字段由一个**Field**的实例来表现。
 - 比如 CharField 表示字符类型的字段；
 - DateTimeField 表示日期时间型的字段；
 - 这会告诉 Django 每个 字段都保存什么类型的数据。

修改eblog.models.py

- 每一个 Field 实例的名字就是**字段的名称**
 - 如： title、content、timestamp。
- 在你的 Python 的**代码**中会使用这个值，而你的**数据库**会将这个值作为表的**列名**。

修改eblog.views.py

■ 获取网页要显示的内容

```
views ✕  
1 # Create your views here.  
2 from django.template import loader, Context  
3 from django.http import HttpResponse  
4 from eblog.models import BlogPost  
5  
6 def archive(request):  
7     posts = BlogPost.objects.all() # @UndefinedVariable  
8     t = loader.get_template('archive.html')  
9     c = Context({'posts': posts})  
10    return HttpResponse(t.render(c))
```

修改DTest.setting.py

- 找到下面部分进行修改

```
33 INSTALLED_APPS = (  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles',  
40     'ebLog',  
41 )
```

修改DTest.urls.py

urls

```
1 from django.conf.urls import patterns, include, url
2
3 from django.contrib import admin
4 admin.autodiscover()
5
6 from eblog.views import archive
7
8 urlpatterns = patterns('',
9     # Examples:
10     # url(r'^$', 'mysite.views.home', name='home'),
11     # url(r'^blog/', include('blog.urls')),
12
13     url(r'^admin/', include(admin.site.urls)),
14     url(r'^eblog/', archive),
15 )
```

建立样式网页模板

- 在包eblog下添加templates文件夹;
- 并在templates下建立两个网页文件:
archive.html和base.html
- archive.html用来显示每条博客的标题、内容和时间。
- base.html用来显示整个博客的布局和博客名称。

编辑archive.html

archive.html

```
1 {% extends "base.html" %}
2 {% block content %}
3 {% for post in posts %}
4 <h1>{{ post.title }}</h1>
5 <p>{{ post.content }}</p>
6 <p>{{ post.timestamp|date:"1, F jS"}}</p>
7 {% endfor %}
8 {% endblock %}
```


编辑base.html

base.html

```
1 <html>
2   <style type="text/css">
3     body { color: #edf; background: #453; padding: 0 5em; margin:0 }
4     h1 { padding: 2em 1em; background:#675 }
5     h2 { color: #bf8; border-top: 1px dotted #fff; margin-top: 2em }
6     p { margin: 1em 0 }
7   </style>
8   <body>
9     <h1><center>Shusen Zhou's Blog</center></h1>
10    {% block content %}
11    {% endblock %}
12  </body>
13</html>
```

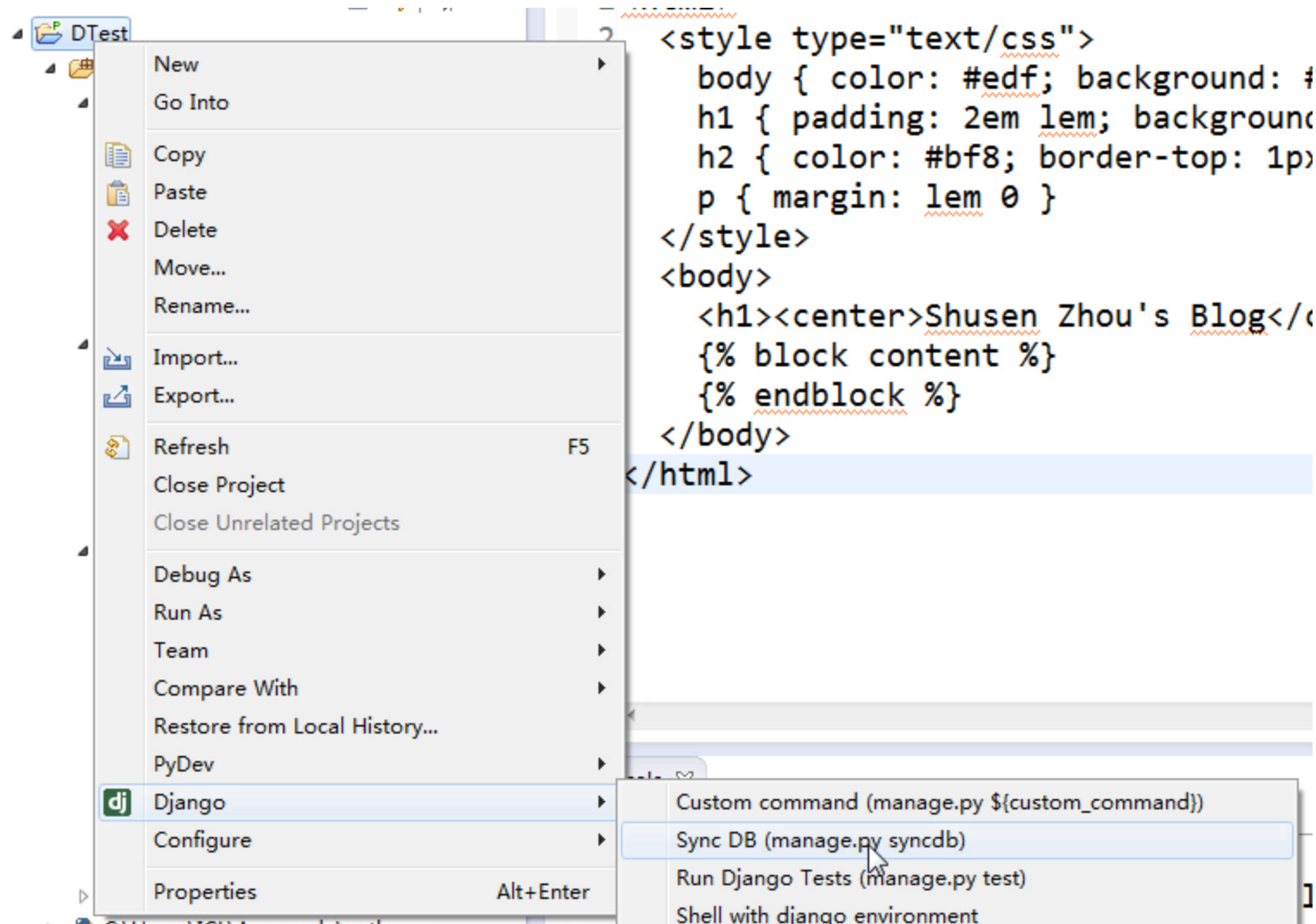
同步数据库

- 用来创建多个相关应用的表格；

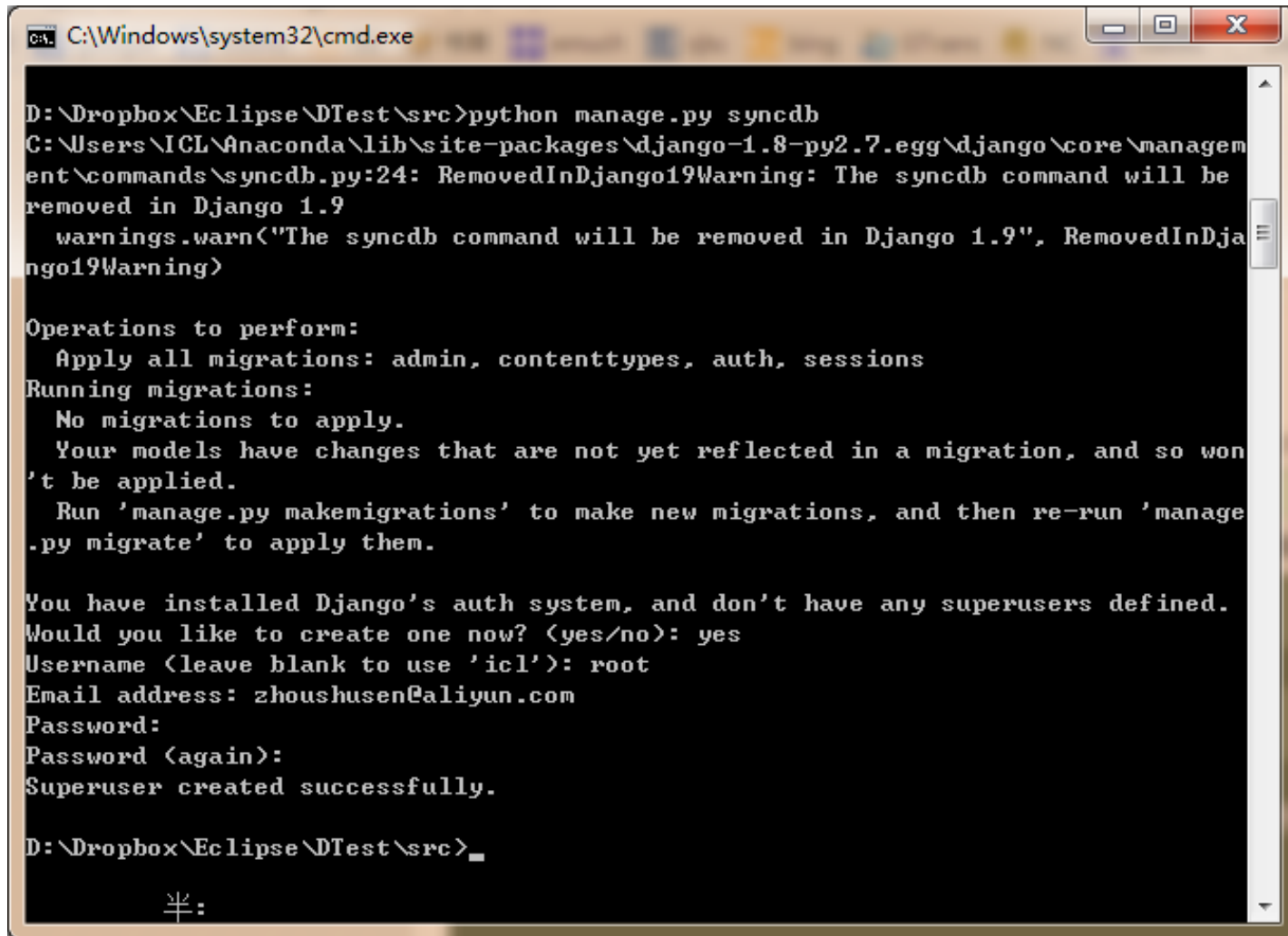
```
33 INSTALLED_APPS = (  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles',  
40     'eblog',  
41 )
```

- 然后设置你的账号和密码，为登陆blog的管理后台作准备。

在Eclipse界面同步数据库



用命令行同步数据库



```
C:\Windows\system32\cmd.exe

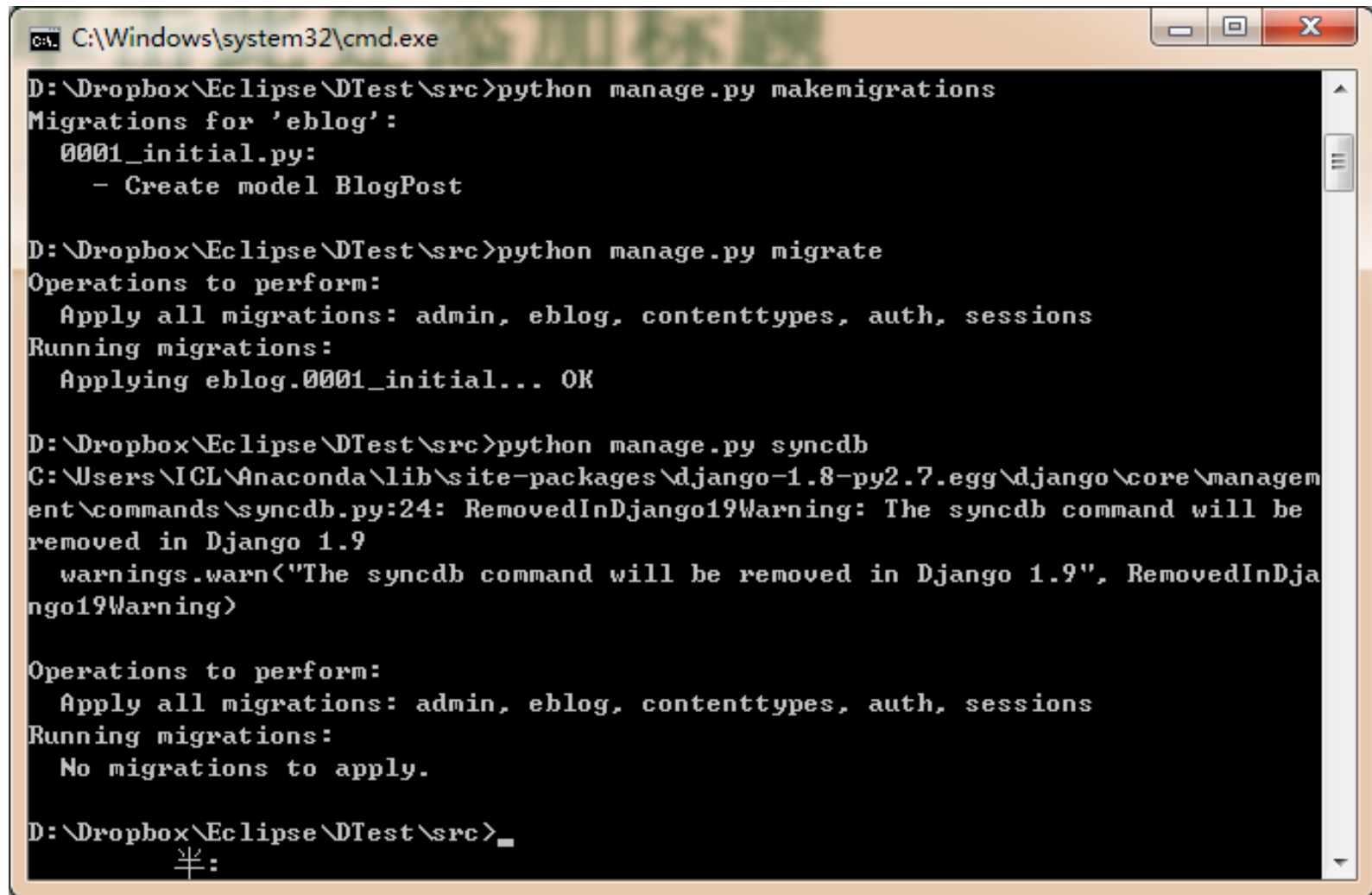
D:\Dropbox\Eclipse\DTest\src>python manage.py syncdb
C:\Users\ICL\Anaconda\lib\site-packages\django-1.8-py2.7.egg\django\core\managem
ent\commands\syncdb.py:24: RemovedInDjango19Warning: The syncdb command will be
removed in Django 1.9
  warnings.warn("The syncdb command will be removed in Django 1.9", RemovedInDja
ngo19Warning)

Operations to perform:
  Apply all migrations: admin, contenttypes, auth, sessions
Running migrations:
  No migrations to apply.
  Your models have changes that are not yet reflected in a migration, and so won't
be applied.
  Run 'manage.py makemigrations' to make new migrations, and then re-run 'manage
.py migrate' to apply them.

You have installed Django's auth system, and don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'icl'): root
Email address: zhoushusen@aliyun.com
Password:
Password (again):
Superuser created successfully.

D:\Dropbox\Eclipse\DTest\src>_
半:
```

用命令行同步数据库



```
C:\Windows\system32\cmd.exe

D:\Dropbox\Eclipse\DTest\src>python manage.py makemigrations
Migrations for 'eblog':
  0001_initial.py:
    - Create model BlogPost

D:\Dropbox\Eclipse\DTest\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, eblog, contenttypes, auth, sessions
Running migrations:
  Applying eblog.0001_initial... OK

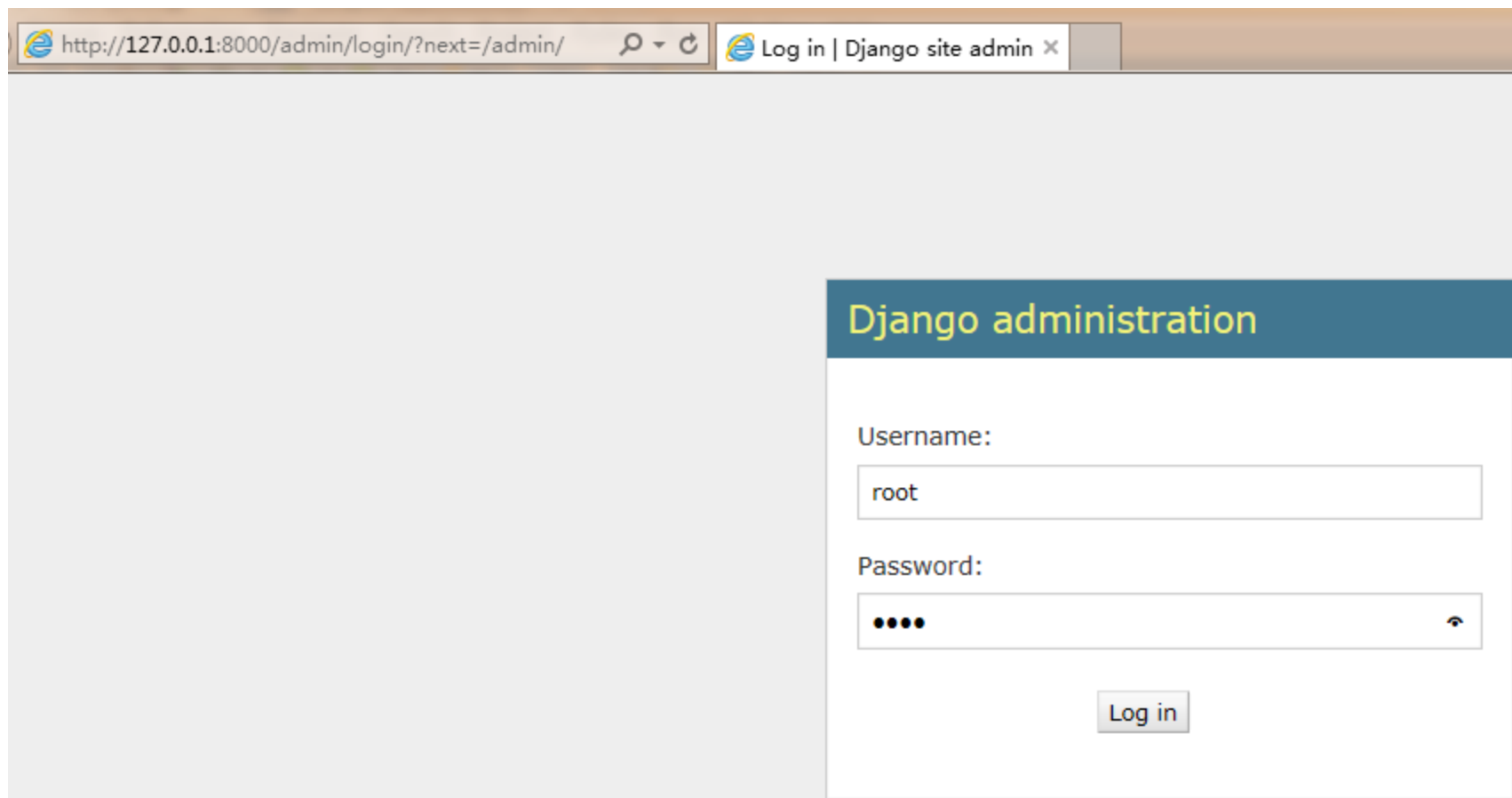
D:\Dropbox\Eclipse\DTest\src>python manage.py syncdb
C:\Users\ICL\Anaconda\lib\site-packages\django-1.8-py2.7.egg\django\core\managem
ent\commands\syncdb.py:24: RemovedInDjango19Warning: The syncdb command will be
removed in Django 1.9
  warnings.warn("The syncdb command will be removed in Django 1.9", RemovedInDja
ngo19Warning)

Operations to perform:
  Apply all migrations: admin, eblog, contenttypes, auth, sessions
Running migrations:
  No migrations to apply.

D:\Dropbox\Eclipse\DTest\src>_
半:
```

运行测试

- 账号和密码是初始化数据库的时候设定的



The screenshot shows a web browser window with the address bar displaying `http://127.0.0.1:8000/admin/login/?next=/admin/`. The page title is "Log in | Django site admin". The main content area features a "Django administration" header in a blue box. Below the header, there are two input fields: "Username:" with the value "root" and "Password:" with masked characters (dots). A "Log in" button is positioned below the password field.

http://127.0.0.1:8000/admin/login/?next=/admin/ Log in | Django site admin ×

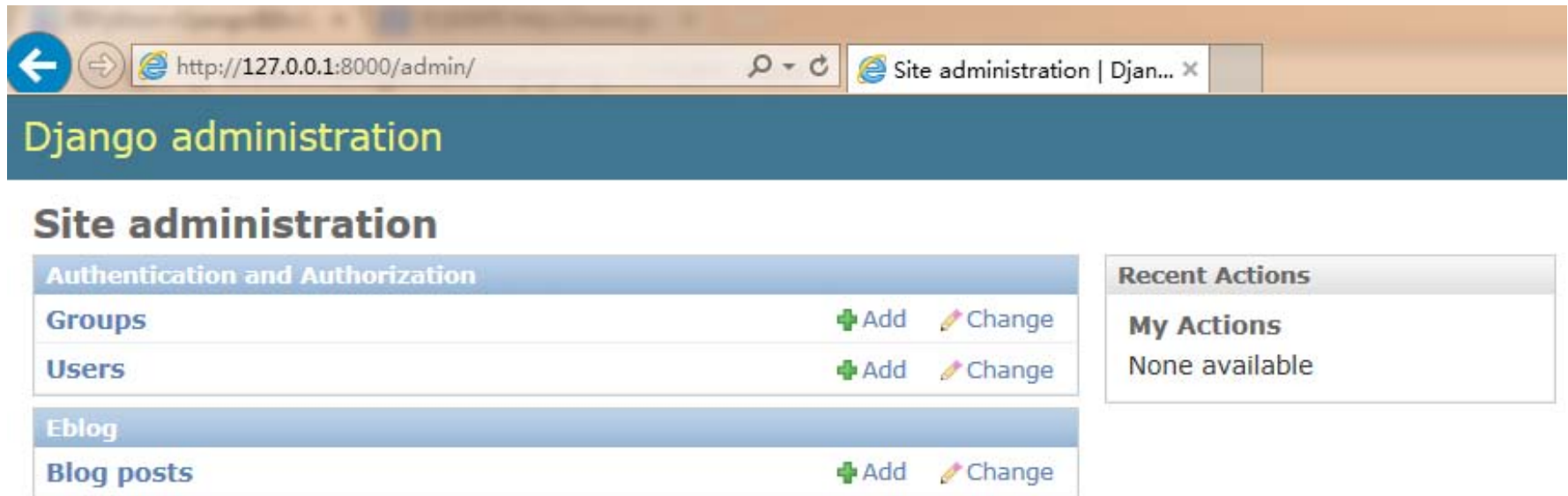
Django administration

Username:
root

Password:
.....

Log in

运行测试



运行测试

← → http://127.0.0.1:8000/admin/eblog/blogpost/add/ Add blog post | Django ... x

Django administration

[Home](#) > [Eblog](#) > [Blog posts](#) > [Add blog post](#)

Add blog post

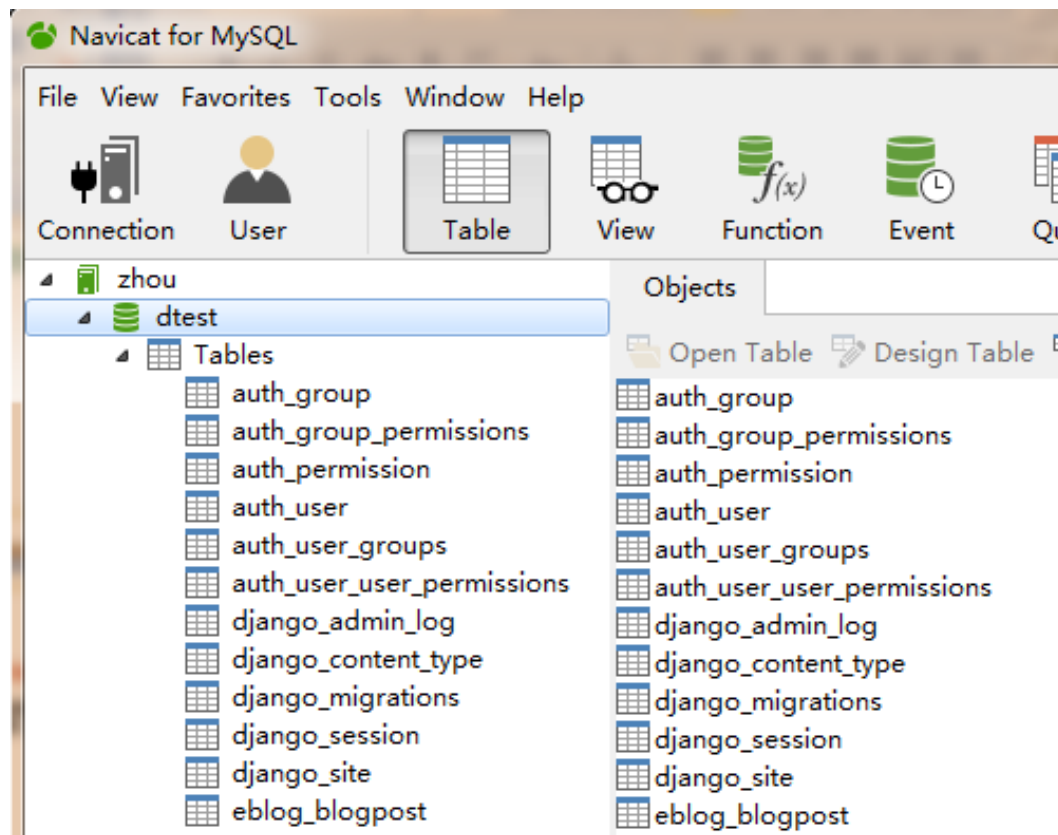
Title:	<input type="text" value="测试成功"/>
Content:	<div>Django 创建Blog , 终于测试成功了</div>
Timestamp:	<div>Date: <input type="text" value="2014-10-06"/> Today </div> <div>Time: <input type="text" value="17:44:27"/> Now </div>

运行测试



Navicat for MySQL查看数据库

- 使用Navicat for MySQL查看经过测试操作后的数据库。



Navicat for MySQL查看数据库

auth_user @dtest (zhou) - Table - Navicat for MySQL

File View Favorites Tools Window Help

Connection User Table View Function Event Query Report Backup Schedule Model

zhou

dtest

Tables

- auth_group
- auth_group_permissions
- auth_permission
- auth_user

Objects

Begin Transaction Memo Filter Sort Import Export

id	password	last_login	is_superuser	username	first_name	last_name
1	pbkdf2_sha256\$20000\$11	2014-10-06 09	1	root		
2	pbkdf2_sha256\$20000\$3e	(Null)	0	test		

django_admin_log @dtest (zhou) - Table - Navicat for MySQL

File View Favorites Tools Window Help

Connection User Table View Function Event Query Report Backup Schedule Model

zhou

dtest

Tables

- auth_group
- auth_group_permissions
- auth_permission
- auth_user
- auth_user_groups
- auth_user_user_permissions
- django_admin_log

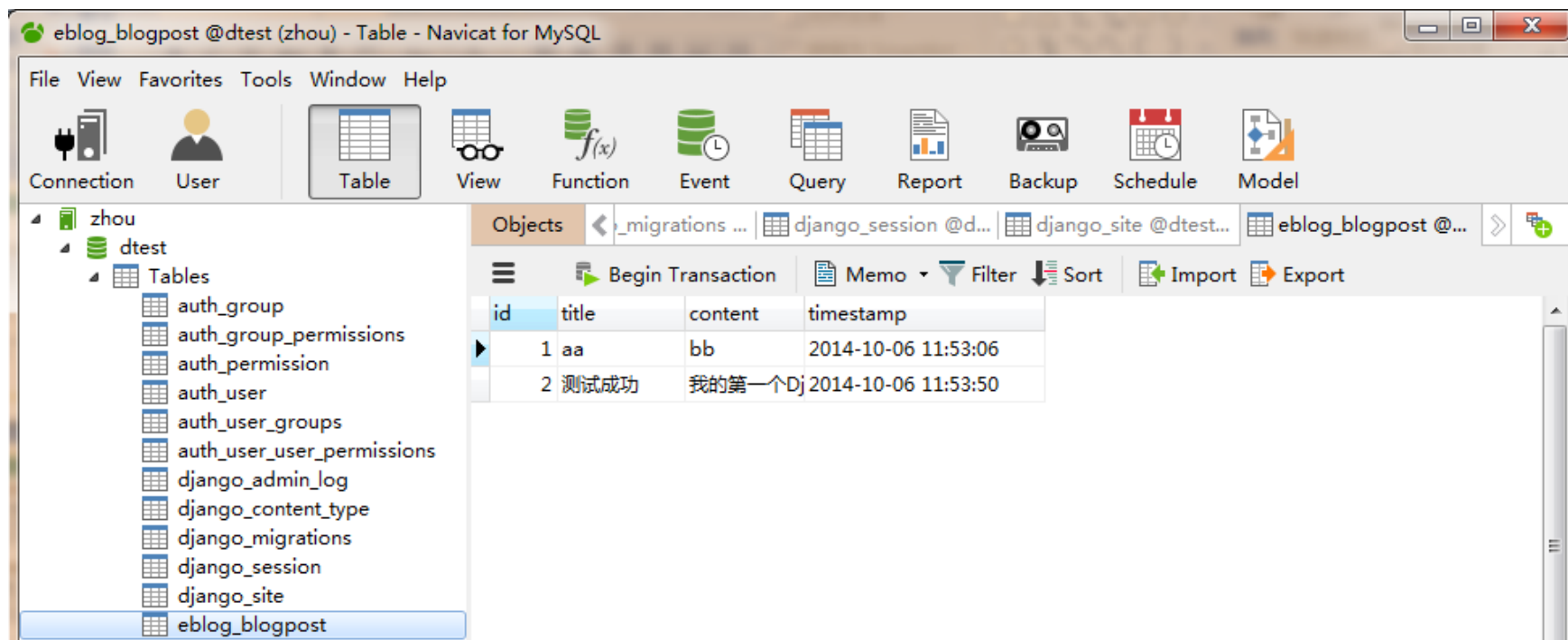
Objects

ser @dtest (... auth_user_groups ... auth_user_user_per... django_admin_log ...

Begin Transaction Memo Filter Sort Import Export

id	action_time	object_id	object_repr	action_flag	change_message	content_type_id
1	2014-10-06 11:3	1	ICL	1		
2	2014-10-06 11:3	2	test	1		
3	2014-10-06 11:4	2	aa	1		
4	2014-10-06 11:5	1	BlogPost object	1		
5	2014-10-06 11:5	2	BloqPost object	1		

Navicat for MySQL查看数据库



小结

■ Django

- Django的主要目的是**简便、快速**地开发**数据库驱动**的网站——**动态**网站。
- Django强调**代码复用**，多个组件可以方便地以“插件”形式服务于整个框架。

谢谢大家