DJANGO

Web framework for deadline driven perfectionists!

By

Tanvi Shah

What is Django?

- Open Source Web application framework in Python.
- Designed to handle two challenges
  - hard deadlines
  - Rigid requirements
- To build web applications quickly and easily.

Tidbits : Django was named after a gypsy jazz guitarist Django Reinhardt

DJANGO AT AGLANCE

A quick view of Django

Django At A Glance

- DESIGN THE MODEL
  - Describe the database layout in Python code.
- INSTALL IT
  - Run the following utility to create the database tables: Manage.py syncdb
- API CREATION
  - API is created on fly. No code generation is necessary.

Django At A Glance

- ADMIN INTERFACE
  - Automatically provides an admin interface.
  - No need to create backend interfaces to manage content.
- DESIGN URLs
  - Create a Python module called **URLconf**.
  - The code maps URLs.
- WRITE VIEWS
  - A View is responsible for returning a **HttpResponse object** or **Http404** exception object.

Django At A Glance

- WRITE VIEWS
  - A view retrieves data according to some parameters, loads templates and renders template (with the data that is retrieved).

- DESIGN TEMPLATES
  - Django uses the "**template inheritance**" concept.

Design Philosophy

Django designs

Fundamental Philosophies

- Loose coupling and tight cohesion
  - Every piece of a Django system is independent of each other.
- Use as much **less code** as possible
- Quick development
- DRY – Don't Repeat Yourself!

Fundamental Philosophies

- Explicit better than implicit
  - This follows from the core Python principle.
  - Let there be no “magic” until very necessary.
- Consistency
  - Consistency for both Python code and higher –level code.
- Model-view-controller architecture pattern of web development.

WHAT IS DRY??

Duplication is Devil

- Change of one element in a system should not affect any other part of the system.

- **Data transformation** and **code generators** are the key to DRY code.

Model-View-Controller (MVC)

- Design pattern used for maintaining multiple views of same data.

- Object is divided into three parts:
  - **Model** – is used to maintain the data
  - **View** – is used to display the data
  - **Controller** – handles events affecting the model

- Multiple views and controllers can interact with the same model.

DJANGO FRAMEWORK FEATURES

Framework Features

Framework Features In Brief

- Object Relational Mapper
  - Data model defined in Python with a dynamic database-access API.
  - SQL queries can be written if necessary.
- Automatic Admin Interface
  - Interface for adding and updating readily provided.
- Elegant URL Design
  - Designing clean URLs with no ".php" or ".cgi" extension.

Framework Features

- Template System
  - Uses the Template Language for separating the design, content and Python code.
- Cache System
  - Uses Caching Frameworks for super performance.
- Internalization
  - Provides support for different languages.

OBJECT RELATIONAL MAPPER AND MODEL

FRAMEWORK FEATURES IN DETAIL

What Exactly Is A Model ?

- Data source that contains fields and behaviors and it maps to a single database table defines a model.

```
from django.db import models

Class Person(models.Model) :
first_name = models.CharField(max_length =30)
last_name = model.CharField(max_length = 30)
```

Every model is a Python class that subclasses "models"

first_name and last_name are fields of the model = database columns

```
CREATE TABLE myapp_person (

"id" serial NOT NULL PRIMARY KEY,
"first_name" varchar(30) NOT NULL,
"last_name" varchar(30) NOT NULL );
```

myapp_person database table automatically created. Id field also automatically added.

Fields

- This is the most important part of the model.
- Specified by class attributes as shown in the example in the previous slide.
- Field class type determines integer or varchar data type for the database column.

Fields

- There are different field options like default, null, max_length, primary key, choices and unique are provided.

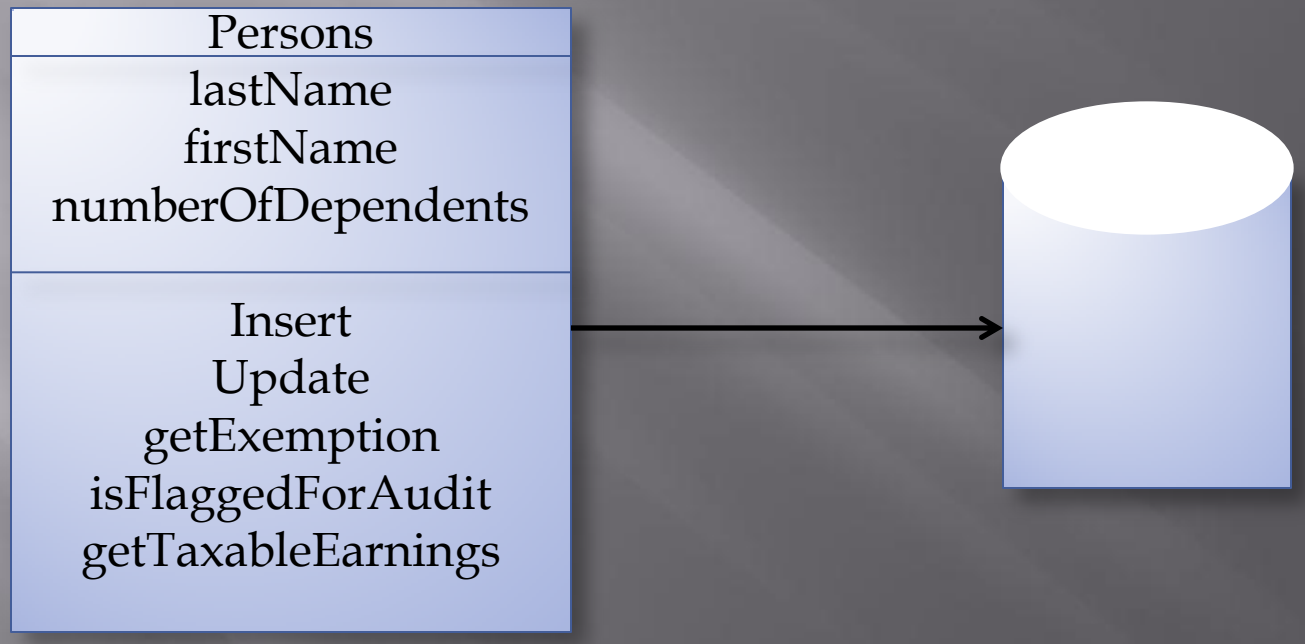
- Database relationships available in Django:
  - Many-to-one relationship using foreign keys.
  - Many-to-many relationships.
  - One-to-one relationships

Model's Design Philosophy

- Explicit is better than implicit.
  - Keyword arguments or type of the field determine the behavior.
  - No system knowledge is require and less faulty.
- Includes all relevant domain logic
  - Encapsulating data and information which is stored in the model.
  - Follows Active Record design pattern.

What is an active record??

- It connects business objects and database tables which creates a domain model.



- Active Record is used to put data access logic in the domain object which enables reading and writing data to and from the database.

Database API

- A database API is automatically created after creating the data-model. It can create, retrieve, update and delete objects.

- The main objectives of this API are:
  - SQL Efficiency
  - Terse, powerful syntax
  - Option to drop into raw SQL easily

Custom SQL

- Django provides two methods of executing raw SQL queries:

1. Manager.raw()

```
for p in Person.objects.raw
('SELECT * FROM myapp_person'):
print p
```

```
John Smith
Jane Jones
```

This query gives the following result

2. Execute Custom SQL directly

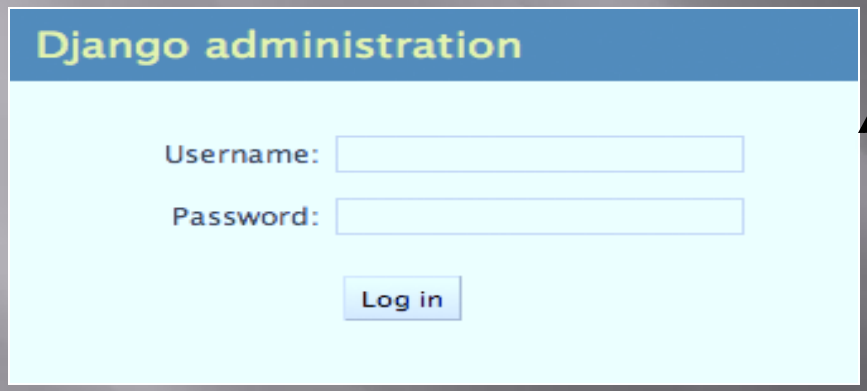
```
def my_custom_sql():
        from django.db import connection, transaction   cursor
= connection.cursor()

        # Data modifying operation - commit required
        cursor.execute("UPDATE bar SET foo = 1 WHERE
        baz = %s", [self.baz])
        transaction.commit_unless_managed()
```

MORE FRAMEWORK FEATURES

ADMIN INTERFACE, URL DESIGN, TEMPLATE SYSTEM, CACHING SYSTEM, VIEWS

Admin Interface

- The automatic admin interface provided by Django looks like this:



Admin's login screen

Admin's Index page

URL Design

- URL design in Django follows these principles:
  - Do not tie URLs to Python function names.
  - Infinite flexibility
  - Encourages best practices
  - Definitive URLs

URL Dispatcher

- Designing URLs requires **URLconf** Python module.
- URL patterns simply map to Python callback functions(views).

Makes patterns() function available

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
        (r'^articles/2003/$',news.views.special_case_2003'),
        (r'^articles/(\d{4})/$', 'news.views.year_archive'),
        (r'^articles/(\d{4})/(\d{2})/$', 'news.views.month_archive'),
        (r'^articles/(\d{4})/(\d{2})/(\d+)/$','news.views.article_detail'
),
)
```

/articles/2005/03 matches the third entry in the list

Django calls the function News.views.month_archive(request, '2005', '03')

Template System Design Philosophy

- Separate logic from presentation.
- Discourage redundancy.
- Be decoupled from HTML.
- XML should not be used for template languages.
- Assume designer competence.
- Treat whitespace obviously.
- Don’t invent a programming language.
- Safety and Security.
- Extensibility.

Template Language

- Text file containing variables that are replaced with values and tags which control the logic of the template defines a template language.

```
{% extends "base_generic.html" %}
{% block title %}{{ section.title }}{% endblock %}
{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
        <a href="{{ story.get_absolute_url }}">
         {{story.headline|upper }}
 </a>
</h2>
 <p>{{ story.tease|truncatewords:"100" }}</p>
 {% endfor %} {% endblock %}
```

This template "extends" base.html

{{section.title}} variable replaced by *title* of the *section* object

Dot (.) access attributes of a variable

Designed to strike balance between power and ease.

Other Useful Features

- Caching Framework
  - Dynamic pages can be cached to reduce overhead.
  - Django offers different levels of caching
    - Caching output of specific views.
    - Caching only difficult pieces.
    - Caching entire site.
- Syndication Framework
  - Helps creating RSS and Atom feeds easily.

Objectives Of Views

- Writing views should be very SIMPLE.
- Use REQUEST objects
  - A request object that stores metadata about the current request should be passed directly to a view function.
- A view should not care about the template system being used.
- Easily differentiate between GET and POST.

DJANGO IN YOUR LIFE

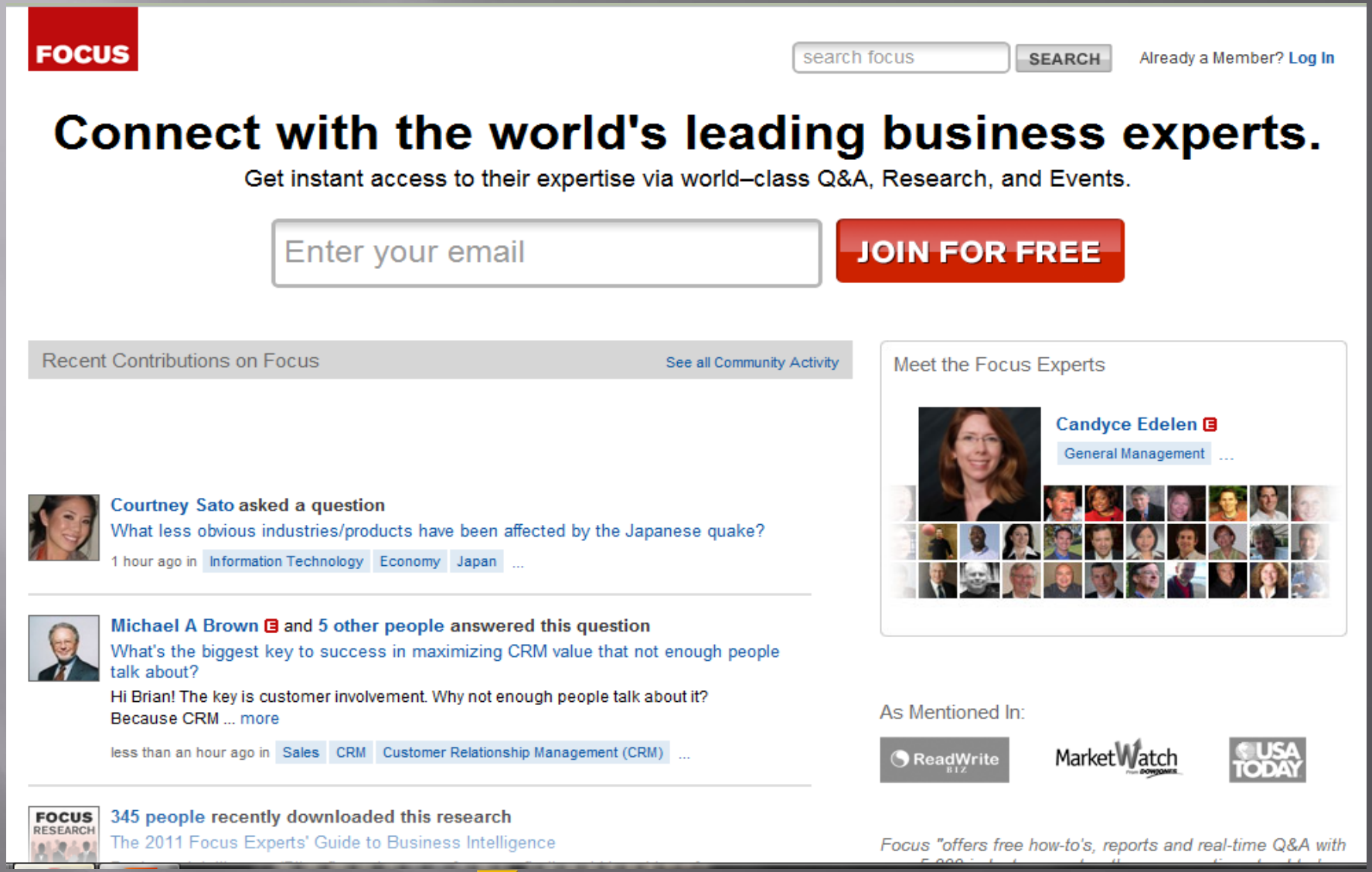
A few examples!

Django in action

Magazine Publishers	Zimbio.com
For networking	Focus.com
Online Advertisers	Discountshuffle.com
Chinese learners	TargetChinese.com

Tidbits : Django is pronounced JANG-OH where “D” is silent

TargetChinese.com



Focus.com
Business Networking

Getting Started with Django

Installation details

Installation

- Install Python first.
- Set up a database like PostgreSQL, MySQL or Oracle.
- If you have any older version of Django, remove them.
- Install Django.
- Verify if Django has been installed.

Getting Started With Django

- Let's build a website that has a public site and an admin interface to do special things.

- Create a Project
  - ***cd*** into a directory
  - Run django-admin.py startproject pollsite
  - Creates a pollsite directory in the current location.
  - Startproject created :
    - Pollsite/
    - _init_.py
    - Manage.py
    - Settings.py
    - Urls.py

Getting Started With Django

- Check the development server by changing into ***pollsite*** directory and run ***python manage.py runserver***.
- Setup the database by editing ***settings.py***.
- Change keys ENGINE, NAME, USER, PASSWORD and HOST to something more understandable to us.
  - Create the database table by running

    python manage.py syncdb
  - The environment is all set.

Getting Started With Django

- Creating Models
  - Go into the pollsite directory and type
  - ***python manage.py startapp polls***
  - Edit polls/model.py file to suit the needs of the user.

- Activating Models
  - The model code creates a database schema and a Python database-access API.
  - Edit the settings.py file.

Getting Started With Django

- Have fun with the API
  - Invoke the Python shell using *python manage.py shell*
  - This sets up the environment for Django.
  - Get ready to churn out websites in record time!

Thank You

References

- DjangaoProject.com
- Wikipedia
- www.c2.com/cgi/wiki?DontRepeatYourself
- www.enode.com/x/markup/tutorial/mvc.html
- www.martinfowler.com/eaaCatalog/activeRecord.html
- www.ar.rubyonrails.org