

Web Development

Technical Summer School 2019, IIT Bombay – Parth Patil

Part 5 – Django

Dynamic HTML

- Store only the template and data
- Create the “rendered” HTML only when asked for
- Instead of saving the HTML, send it to the client
- Allows changing data (very) frequently
- Can recognize user and generate specific content

Django

- Web framework,
- Written in Python,
- **Model-view-template** architectural pattern.

Template

- Format to display the data
- Like a sample
- HTML (or any other) with variables

Model

- Structure to access data
- Data access becomes easier
- Linked models
- OOP – Object Relation Mapping (ORM)

View

- Interacts with the user
- Performs operations like filling up the Model
- Generates the template and returns to the user

Database (RDBMS)

- Convenient way to store/access data
- Written by top coders
- Easy to use APIs
- Django – built-in ORM for many databases

SQLite

- One RDBMS (**R**elational Database Management System)
- Everything in one file
- Easier to manage for smaller applications
- Typically slow for larger real-world sites

Migrations

- Successively modify database structure
- Can go from one point to another easily
- Define database with code

Getting Started

```
$ python -V  
$ pip install django  
$ django-admin startproject mysite  
$ django-admin startapp product
```

Getting Started

- `INSTALLED_APPS`
- `python manage.py migrate`
- `python manage.py runserver`

Django Model

```
from django.db import models
```

```
class Product(models.Model):
```

```
    name = models.CharField(max_length=50)
```

```
    description = models.TextField(blank=True)
```

```
    image_url = models.URLField(blank=True, null=True)
```

```
    website_url = models.URLField(blank=True, null=True)
```

Making/ Applying Migrations

- `python manage.py makemigrations`
- `python manage.py migrate`

Django Admin

- `python manage.py createsuperuser`
- `localhost:8000/admin/`

```
#admin.py
```

```
from django.contrib import admin  
from product.models import Product
```

```
admin.site.register(Product)
```


__str__(self)

- Overriding default method
- Useful in admin

```
def __str__(self):  
    return self.name
```

Views

```
# views.py
from django.http import HttpResponse

def index(request):
    return HttpResponse("<h1>Welcome to Django!</h1>")
```

URLs

```
# urls.py
from django.contrib import admin
from django.urls import path
import product.views as pv

urlpatterns = [
    path('admin/', admin.site.urls),
    path('product/', pv.index),
]
```


Getting URL Information

```
path('product/<pk>', pv.index),
```

```
def index(request, pk):
```

```
    return HttpResponse("<h1>Welcome to Django! " + pk + "</h1>")
```

Querying Data

```
from product.models import Product
```

```
def index(request, pk):  
    p = Product.objects.get(name=pk)  
    return HttpResponse(p.description)
```

Rendering a Template

```
from product.models import Product
def index(request, pk):
    p = Product.objects.get(name=pk)
    context = {'product': p}
    return render(request, 'product.html', context)
```


Working Template

```
<h1> {{ product.name }} </h1>
```

```
<p> {{ product.description }}</p>
```

```
<a href="{{product.url}}">URL</a>
```

Adding more fields

- Add fields
- Make Migrations
- Migrate

```
is_discounted = models.BooleanField(default=False)
```

```
{% if product.is_discounted %}  
    <b>Currently under discount</b>  
{% endif %}
```

Extra – storing relational data

Thank You!
