



# WEB DEVELOPMENT

---

## Lesson 11

# Regular Expressions in URL

Django 1.*	Django 2.*
url	path
url	re_path

# Regular Expressions in URL

- — (dot) Any character
- \d** — Any digit
- [A-Z]** — Any character A-Z (uppercase)
- [a-z]** — Any character, a-z (lowercase)
- [A-Za-z]** — Any character, a-z (case insensitive)
- +** — One or more of the previous expression
- [^/]+** — All characters except forward slash
- ?** — Zero or more of the previous expression
- {1,3}** — Between one and three (inclusive) of the previous expression

<https://docs.djangoproject.com/en/2.1/topics/http/urls/>

# How Django Processes a Request

- `python manage.py runserver`
  - `settings.py`
  - `ROOT_URLCONF`
- `/time/`
  - look through all urls patterns and compare
  - `HttpRequest` object as first parameter
  - view function responsible to return `HttpResponse`

# 404 Errors

# Word About Pretty URLs

```
(r'^time/plus/\d+/$', hours_ahead),
```

- /time/plus/1/
- /time/plus/2/
- /time/plus/3/
- /time/plus/1000000/

# Word About Pretty URLs

```
(r'^time/plus/\d{1,2}/$', hours_ahead),
```

- /time/plus/**1**/
- /time/plus/**2**/
- /time/plus/**3**/
- /time/plus/1000000/

# Passing that data to the view function

parentheses around the data

```
(r'^time/plus/(\d{1,2})/$', hours_ahead),
```



# Coding Order

1. views —> urls

2. urls —> views

# Django new App

# The Django Template System

```
(r'^time/plus/(\d{1,2})/$', hours_ahead),
```

# The Django Template System

html template files

# The Django Template System

render function

# The Django Template System

include html template — block

# The Django Template System

context for render function

# The Django Template System

Jinja2



# The Django Template System

passing params to template as context

# Basic Template Tags and Filters

## `if/else`

```
{% if today_is_weekend %}  
    <p>Welcome to the weekend!</p>  
{% endif %}
```

# Basic Template Tags and Filters

## for

```
<ul>
  {% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
  {% endfor %}
</ul>
```

# Basic Template Tags and Filters

## Comments

```
{# This is a comment #}
```

# Basic Template Tags and Filters

Template inheritance

# MTV

- **M (Model)** — data access layer
- **T (Template)** — presentation layer
- **V (View)** — business logic layer

# Configuring the Database

# Configuring the Database

## `settings.py`

- `DATABASE_ENGINE = ''`
- `DATABASE_NAME = ''`
- `DATABASE_USER = ''`
- `DATABASE_PASSWORD = ''`
- `DATABASE_HOST = ''`
- `DATABASE_PORT = ''`



# Configuring the Database


## `settings.py`

- `DATABASE_ENGINE = ''`

Settings	Database	Required Adapter
<code>postgresql</code>	PostgreSQL	<code>psycopg</code> version 1.x
<code>postgresql_psycopg2</code>	PostgreSQL	<code>psycopg</code> version 2.x,
<code>mysql</code>	MySQL	<code>MySQLdb</code>
<code>sqlite3</code>	SQLite	<code>pysqlite</code>
<code>ado_mssql</code>	Microsoft SQL Server	<code>adodbapi</code> version 2.0.1+
<code>oracle</code>	Oracle	<code>cx_Oracle</code>

# Shop — new Django app

# Defining Models in Python

Python model  SQL **CREATE TABLE**

# Create new migration file if necessary

```
>>> python manage.py makemigrations
```

# Execute created migration file

(change database schemas)

```
>>> python manage.py migrate
```

# Inserting and Updating Data

- Create instance of model
  - `save()` // INSERT INTO "table\_name" ...
- Find object and update class fields
  - `save()` // UPDATE "table\_name" SET ...

# Selecting Objects

```
>>> products = Product.objects.all()
```

```
SELECT * FROM products;
```



# Filtering Data

```
>>> p = Product.objects.filter(name="Test name")
```

```
SELECT * FROM products  
WHERE name='Test name';
```

# Filtering Data

- `contains / icontains`
- `startswith / endswith`
- `istartswith / iendswith`
- `exact / iexact`
- `in`
- `isnull`
- `gt / gte`
- `lt / lte`

```
>>> p = Product.objects.filter(name__contains="Test")
```

# Retrieving Single Objects

```
>>> p = Product.objects.get(id=3)
```

```
SELECT * FROM product WHERE id=3;
```

# Ordering Data

```
>>> p = Product.objects.order_by("name")
```

```
SELECT * FROM products ORDER BY name;
```

# Ordering Data

```
>>> p = Product.objects.order_by("name", "address")
```

```
>>> p = Product.objects.order_by("-name")
```

# Meta class

```
class Product(models.Model):  
    ...  
    class Meta:  
        ordering = ["name"]
```

# Chaining Lookups

```
>>> p = Product.objects.filter(name="Laptop").order_by("-price")
```

# Deleting Objects

```
>>> p = Product.objects.get(id=3)
```

```
>>> p.delete()
```

```
>>> products = Product.objects.all()
```

```
>>> products.delete()
```



# Questions?