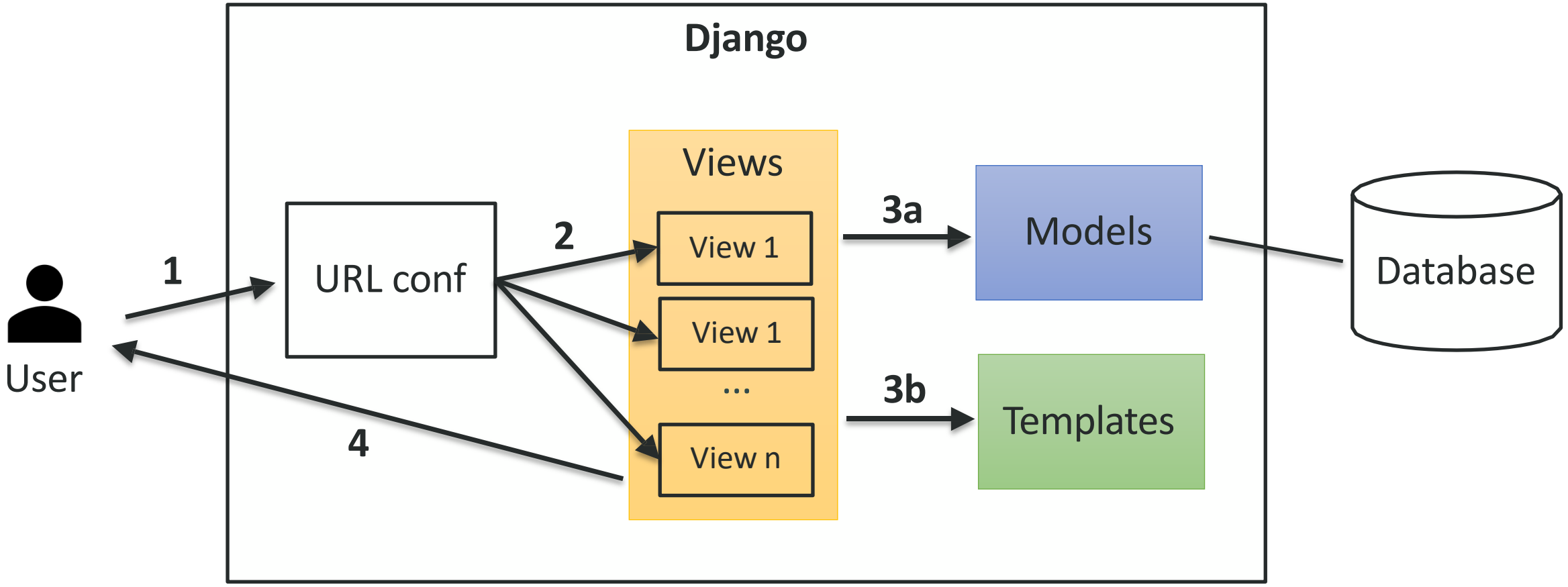
Web Programming

Django II

Prof. Josué Obregón
Department of Industrial Engineering- ITM
Seoul National University of Science and Technology

Django architecture overview



Objectives

- Describe more specific features of the Django Template Language that are essential to construct dynamic webpages.
- Outline the concept of a session to maintain the state information of the users of a web application.
- Define what is a Django form and how to use it.
- Apply the acquired knowledge of the Django framework to construct two dynamic web applications.

Agenda

- Django Template Language
  - Tags
    - if and else
    - for
    - block and extends
  - Filters
    - lower, capfirst, default, length, pluralize
- Django forms
- Django sessions

Tags: `if`, `elif`, and `else`

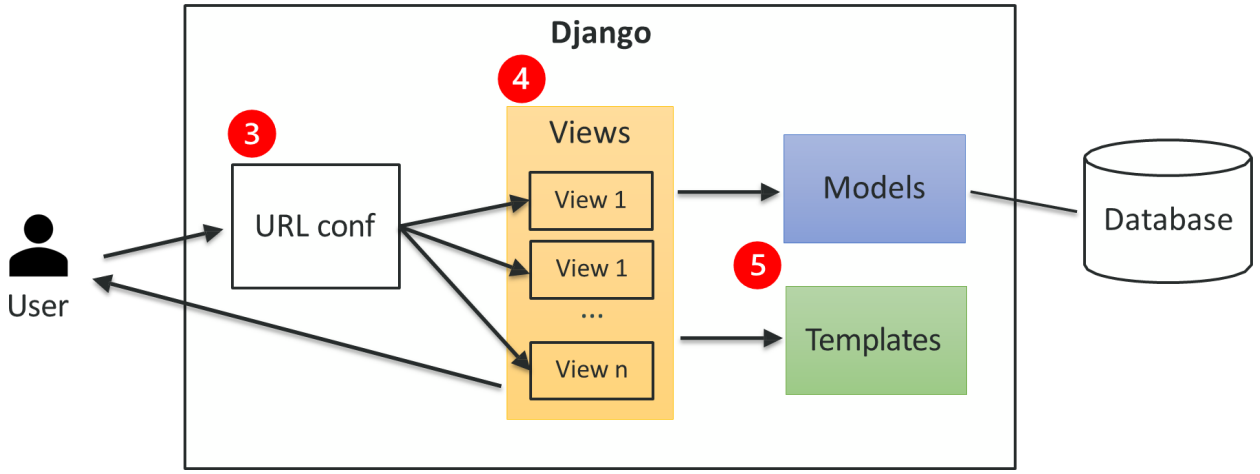
- Evaluates a variable, and if that variable is "true" the contents of the block are displayed

```
{% if athlete_list %}
    Number of athletes: {{ athlete_list}}
{% elif athlete_in_locker_room_list %}
    Athletes should be out of the locker room soon!
{% else %}
    No athletes.
{% endif %}
```

https://docs.djangoproject.com/en/5.0/ref/templates/builtins/#ref-templates-builtins-tags

Create a Django app - Homeplus

1. Create a new app using **startapp** (inside pr_league)
2. Add it to the **settings**
3. Create a **URL**
4. Link it to a **view**
5. Return a **template**
6. ~~Make it dynamic with **context**~~
7. ~~Add some **static** files~~



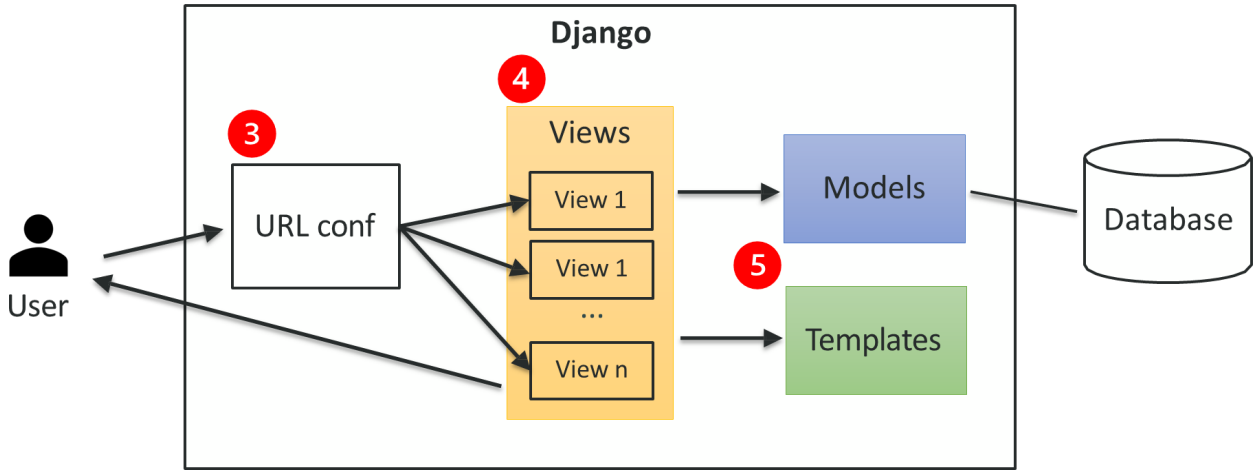
Tags: for

- Loop over each item in an array.
  - For example, to display a list of athletes provided in `athlete_list`:

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

Create a Django app - Books

1. Create a new app using **startapp** (inside pr_league)
2. Add it to the **settings**
3. Create a **URL**
4. Link it to a **view**
5. Return a **template**
6. ~~Make it dynamic with **context**~~
7. ~~Add some **static** files~~



Tags: `block` and `extends`

- The most powerful – and thus the most complex – part of Django's template engine is **template inheritance**.

- Template inheritance allows you to build a base "skeleton" template that contains all the common elements of your site and defines blocks that child templates can override.

The skeleton: layout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="style.css">
    <title>{% block title %}My amazing site{% endblock %}</title>
</head>

<body>
    <div id="sidebar">
        {% block sidebar %}
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/blog/">Blog</a></li>
        </ul>
        {% endblock %}
    </div>

    <div id="content">
        {% block content %}{% endblock %}
    </div>
</body>
```

The child template: child.html

```
{% extends "layout.html" %}

{% block title %}My amazing blog{% endblock %}

{% block content %}
{% for entry in blog_entries %}
    <h2>{{ entry.title }}</h2>
    <p>{{ entry.body }}</p>
{% endfor %}
{% endblock %}
```

Forms

- Django handles three distinct parts of the work involved in forms:
  - preparing and restructuring data to make it ready for rendering
  - creating HTML forms for the data
  - receiving and processing submitted forms and data from the client

- First, define your form in your in views.py

```
from django import forms


class ContactForm(forms.Form):
    subject = forms.CharField(max_length=100)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
```

https://docs.djangoproject.com/en/4.2/topics/forms/

Forms

- Then you send it as a context in the `render` method

```
context = {'form': ContactForm()}
```

- Next step is to render them in your template using different formats
  - {{ form }}
  - {{ form.as_div }}
  - {{ form.as_p }}
  - {{ form.as_ul }}
  - {{ form.as_table }}

Forms

- Form data sent back to a Django website is processed by a view, generally the same view which published the form. This allows us to reuse some of the same logic.
- A `Form` instance has an `is_valid()` method, which runs validation routines for all its fields. When this method is called, if all fields contain valid data, it will:
  - return `True`
  - place the form's data in its `cleaned_data` attribute.

Sessions

- Django provides full support for anonymous sessions.

- The session framework lets you store and retrieve arbitrary data on a per-site-visitor basis.
  - User id
  - Information about the session

- Sessions are implemented via a piece of middleware.
  - Sessions are enabled by default (Check INSTALLED_APPS in your settings.py)

https://docs.djangoproject.com/en/4.2/topics/http/sessions/

Sessions

- When SessionMiddleware is activated, each HttpRequest object – the first argument to any Django view function – will have a **session attribute, which is a dictionary-like object**.

- You can read it and write to `request.session` at any point in your view.

- You can edit it multiple times.

Sessions

- By default, Django stores sessions in your database (using the model django.contrib.sessions.models.Session).

- If you want to use a database-backed session, run `manage.py migrate` to install the single database table that stores session data.
  - More on databases next week

- Other forms are cache-based sessions and cookie-based sessions
  - Check more information here https://docs.djangoproject.com/en/4.2/topics/http/sessions/

Filters

- You can modify variables for display by using filters.

- Filters look like this: **{{ name|lower }}.**
  - This displays the value of the {{ name }} variable after being filtered through the lower filter, which converts text to lowercase.

- Use a pipe (|) to apply a filter.

https://docs.djangoproject.com/en/4.2/ref/templates/builtins/#ref-templates-builtins-filters

Filters

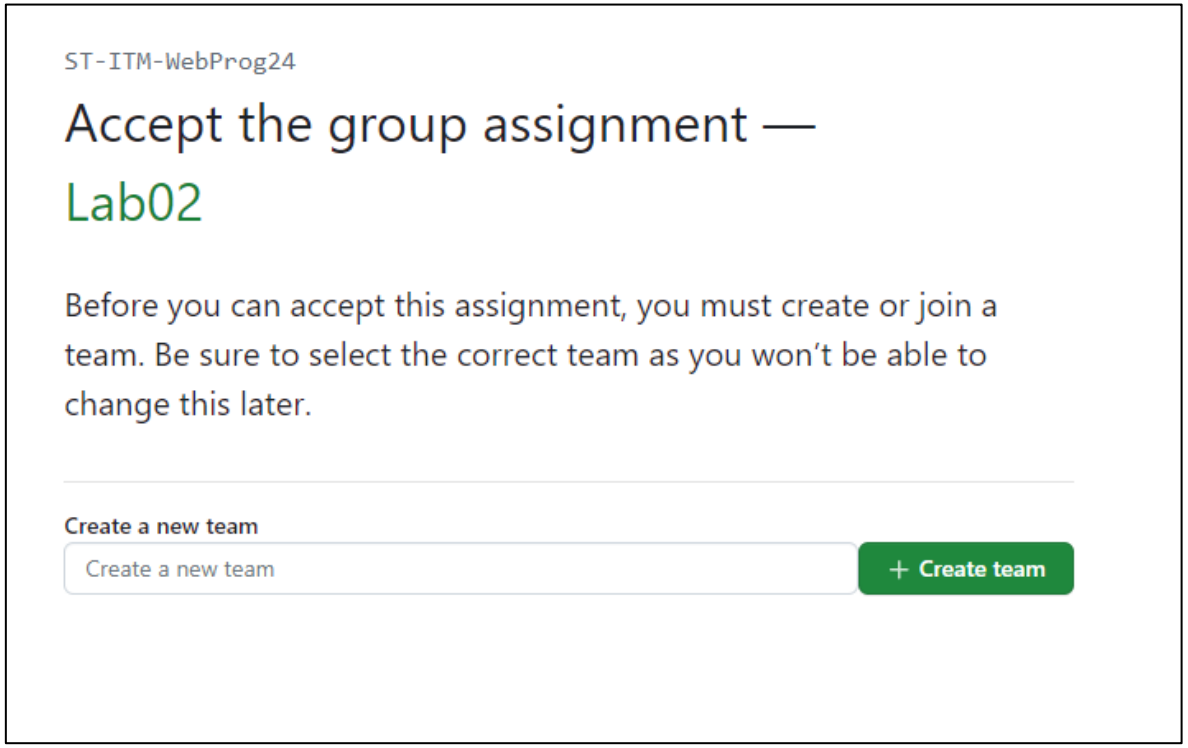
- {{value|default:"nothing"}}
  - If a variable is false or empty, use given default. Otherwise, use the value of the variable
- {{value|length}}
  - Returns the length of the value. This works for both strings and lists
- {{value|capfirst}}
  - Capitalizes the first character of the value. If the first character is not a letter, this filter has no effect.
- {{num_messages|pluralize}}
  - Returns a plural suffix if the value is not 1, '1', or an object of length 1. By default, this suffix is 's'.
  - Ex:

```
You have {{ num_messages }} message{{ num_messages|pluralize }}.
```

Lab Assignment # 2 - Django

GitHub Classroom link

https://classroom.github.com/a/AD-5Ebmg



Web Programming

The contents of this slide are based or adapted from the content of the video course "Introduction to Django" from Arianne Dee, Pearson 2023.