



# Django Level Four

Let's learn something!



# Django Bootcamp

- We've learned a lot so far and yet there is still so much more that Django can offer you!
- In Django Level Four we will focus on learning a lot more about templates.



# Django

- So far we've only used templates as a way of injecting simple pieces into our HTML files.
- But templates are actually capable of much more!



- For example, so far we've been manually creating everything individually for each .html file.
- However we can actually use templates to have a “base” template and inherit that template in the .html files.



- This saves you a lot of time and will help create a unified look and feel across every page of your website!
- Templates are also used to solve issues with relative paths and working with variables.



- Templates can also help solve issues by avoiding hard-coded URL paths.
- Templates also come with built-in filter capabilities so you can adjust variables on the actual individual page.



- Let's get started by talking about using Templates for Relative URLs!



# Relative URLs with Templates

Django Level Four





- So far when we've had to use an anchor tag with an href we've passed in a hardcoded path to the file.
- This is poor practice if we want our Django project to work on any system.



- We will show how to use various methods to pass relative URLs with Templates.
- At the end we will show the preferred method that will be the main method used when future versions of Django are released.



- A quick side note on Django and future releases!
- Django generally has a really good roadmap for future releases, and every 2 years released a “LTS” version of Django, with support of at least 3 years.



- I encourage you to explore the Django Release Notes and documentation for further exploration!
- Usually new releases involve better features and easier methods, not huge paradigm shifts.



- Okay, back to the topic at hand!
- How can we replace a hardcoded URL path in an href with a URL Template?
- Let's see!

- We can easily fix this with the use of URLs in our templates. For example:
  - `<a href="basicapp/thankyou">Thanks</a>`
  - Can be changed to:
  - `<a href="{% url 'thanku' %}">Thanks</a>`
  - `name='thanku'` is in the `urls.py` file.



- Could also just directly reference the view.  
For example:

```
<a href="basicapp/thankyou">
```

```
    Thanks</a>
```

Can be changed to:

```
<a href="{% url 'basicapp.views.thankyou' %}">
```

```
    Thanks</a>
```



# Django

- However this method will eventually go away with Django 2.0 in the future.

```
<a href="basicapp/thankyou">
```

```
    Thanks</a>
```

Can be changed to:

```
<a href="{% url 'basicapp.views.thankyou%'">
```

```
    Thanks</a>
```





- The suggested (and most future-proof) method to do all of this involves the `urls.py` file.
- Inside the `urls.py` file you add in the variable **`app_name`**
- You then set this variable equal to a string that is the same as your app name



- This is the best way to use URL templates:

```
<a href="basicapp/thankyou">Thanks</a>
```

Can be changed to:

```
<a href="{% url 'basicapp:thankyou' %}">  
    Thanks</a>
```

This method requires that `app_name` variable to be created inside the `urls.py` file!



- So far we've only really worked with single application Django projects.
- Later on with the clone projects we will build out Multi-application Django Projects.



- Using templates for relative URLs will really help with multiple applications.
- Let's work through a basic example!



# URLs with Templates Code Examples

Django Level Four



# Django

- The project for this lecture (and the entire section) can be found under the Django\_Level\_Four folder called template\_project.
- Let's get started!



# Template Inheritance

Django Level Four



- Let's learn how we can use Django Template Inheritance to practice DRY coding principles.
- Template inheritance allows us to create a base template we can inherit from.





- This saves us a lot of repetitive work and makes it much easier to maintain the same base look and feel across our entire website!



- For example, if we wanted a navbar at the top of our page, it wouldn't make sense to continually have the same navbar HTML code in each individual .html file.
- Instead we set it to the base.html file and inherit it using template inheritance.



- This idea is sometimes also known as template extending, as in extending the base.html to other .html files.
- The inheritance doesn't need to just be limited to one base.html file, you can extend multiple templates



- Before you begin any Django Project, it is always a good idea to sketch out the main idea and organization by hand.
- This will help you realize what can be used for template inheritance and what applications you should create!



- Here are the main steps for inheritance:
  - Find the repetitive parts of your project
  - Create a base template of them
  - Set the tags in the base template
  - Extend and call those tags anywhere

PIERIAN  DATA



- base.html

```
<links to JS, CSS, Bootstrap>
<bunch of html like navbars>
  <body>
    {% block body_block %}
    {% endblock %}
  </body>
</More footer html>
```

## other.html

```
<!DOCTYPE html>
{% extends "basic_app/base.html" %}
{% block body_block %}
<HTML specific for other.html>
<HTML specific for other.html>
{% endblock %}
```



- Let's walk through a basic example of template inheritance.



# Template Inheritance Code Examples

Django Level Four





# Templates Features and Filters

Django Level Four



- Before we complete Django Level Four and our understanding of templates, let's quickly touch upon Django Template Filters!



- Imagine that you had some information from your model that you wished to use across various views/pages.
- But perhaps you wanted to make a slight edit to the information before injecting it, like string operations, or arithmetic.



- Luckily Django provides a ton of easy to implement template filters that allow you to effect the injection before displaying it to the user.
- This allows for flexibility from a single source!



- The general form for a template filter is:
  - `{{ value | filter:"parameter" }}`
- Not all filters take in parameters.
- Many of these filters are based off of common built-in Python functions so you will be already familiar with them!



- Let's show you the documentation on them so you know how to reference all of them!
- Later we will show you how to create your own filters!



**[docs.djangoproject.com/en/1.10/topics/templates](https://docs.djangoproject.com/en/1.10/topics/templates)**

Go to this link! Or just Google Search:  
Django+Templates



# Template Filters Coding Example

Django Level Four