Pass a message from a view to an HTML page using Jinja templates

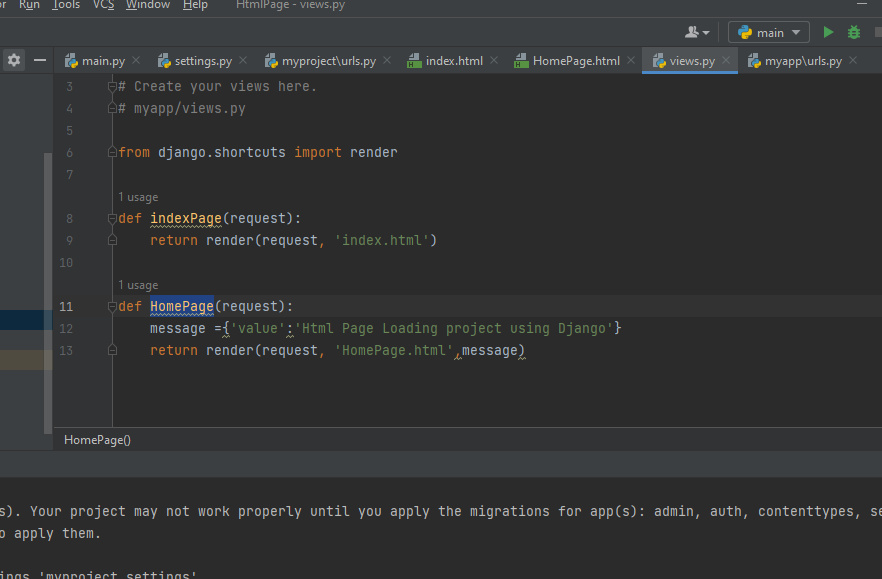
Jinja templates are a type of template engine used in web development frameworks, and they are commonly associated with the Flask framework in the Python ecosystem. Jinja2 is a modern and designer-friendly template engine for Python, modeled after Django's templates but with a few key differences.

* Syntax:
  + Jinja templates use double curly braces ({{ }}) for variable substitution, similar to Django's template syntax.
  + They also use {% %} for control statements such as loops, conditionals, and template inheritance.
* Template Inheritance:
  + Like Django, Jinja templates support template inheritance, allowing you to create base templates with common elements that can be extended or overridden by child templates.
* Filters and Functions:
  + Jinja provides a wide range of built-in filters and functions for manipulating data within templates, similar to Django's template filters.
* Custom Extensions:
  + Jinja templates support custom extensions, allowing you to create custom template tags, filters, and functions to extend the functionality of the template engine.
* Performance:
  + Jinja2 is known for its performance and efficiency, making it suitable for high-traffic web applications.
* Designer-Friendly Syntax:
  + Jinja templates are designed to be easy for designers and front-end developers to work with, with a focus on readability and simplicity.

Follow same methods

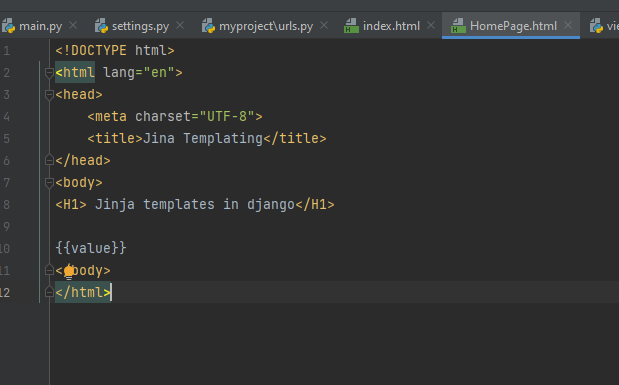
1)Define Your View:

In your view function, define the message you want to pass to the HTML page. Then, render the HTML template with the message passed as a context variable.



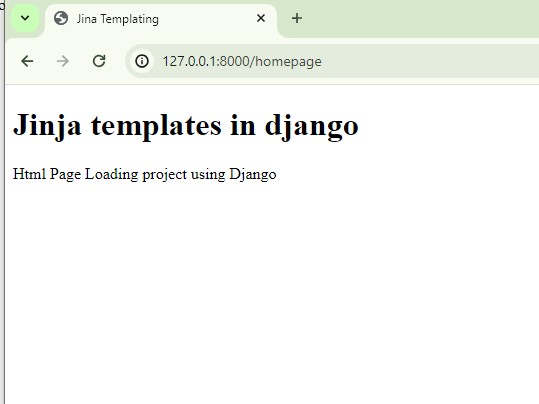
2)Accessing the Data in the Template:

In the Jinja2 template, you can access the data passed from the view using the same syntax as Django's default template engine:



3)Access the HTML Page:

After making these changes, start the development server using python manage.py runserver and access the HTML page as before. Django will now look for templates in the templates directory in the root folder of your project



# 

# 

# 

# 

# 

# 

# Conditional Statements In Django Using Jinja

Instead of dealing with if...else conditionals in the django directly embed them into the Jinja2 templates. With the default syntax, control structures appear inside

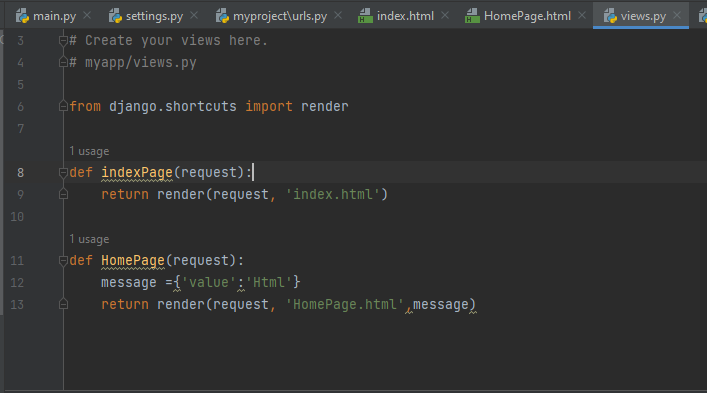
{% ... %} blocks.

* We use {% if %} to start the conditional statement.
* Inside the {% if %} block, we have the condition value == 'Html'. If this condition is true, it will render the <p>Yes</p> paragraph; otherwise, it will render the <p>No</p> paragraph.
* We use {% endif %} to end the conditional statement.

Similarly, you can use other control structures like loops ({% for %}), conditional loops ({% while %}), and template inheritance ({% extends %}, {% block %}, {% include %}) using the {% ... %} syntax in Jinja2 templates. This approach allows for more flexibility and expressive power when creating templates in Django projects.

1)provided HTML template and the associated Django view function step by step:

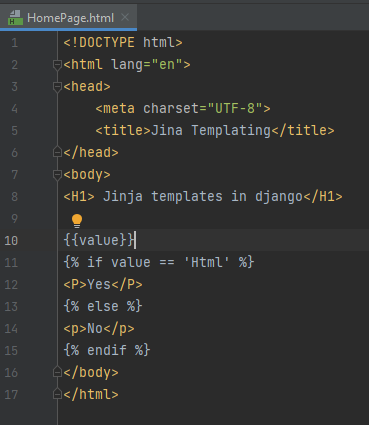
Django View Function (HomePage):



* This is a Django view function named HomePage.
* It takes a request parameter, which is an HttpRequest object representing the request from the client.
* It creates a dictionary named message with a key 'value' set to 'Html'.
* It renders the HomePage.html template, passing the message dictionary as context data to the template.

2)

* The template is a basic HTML document with a title and a heading.
* It includes a placeholder ({{ value }}) where the value passed from the Django view will be rendered.
* It uses a Jinja conditional statement ({% if value == 'Html' %}) to check if the value is 'Html'. Depending on the value, it will render either "Yes" or "No".



3)Access the HTML Page:

After making these changes, start the development server using python manage.py runserver and access the HTML page as before. Django will now look for templates in the templates directory in the root folder of your project



# **for loop – Django Template Tags**

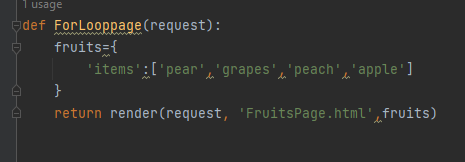
###### **Syntax**

{% for i in list %}

{% endfor %}

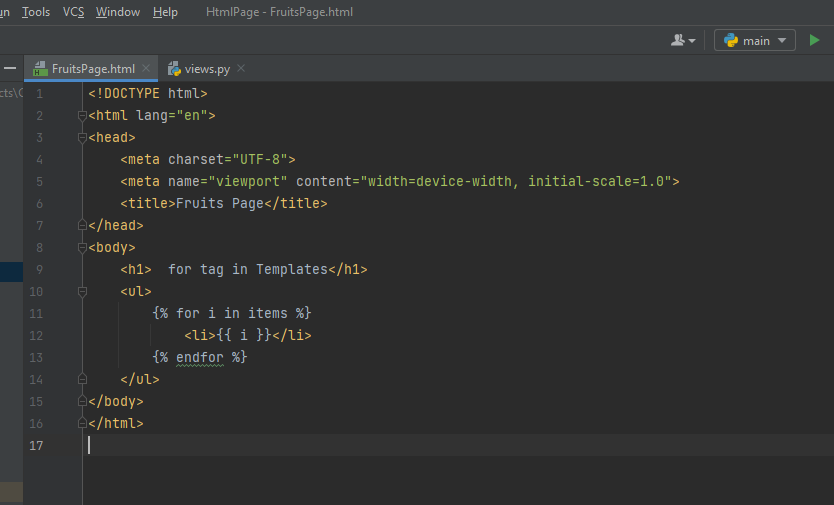
In Django templates, you can use the {% for %} template tag to loop over a sequence, such as a list or queryset, and render the template content multiple times, once for each item in the sequence. Let's break down how to use the {% for %} loop tag step by step:

1)



* This is a Python function defined within a Django views file.
* It's likely a part of your Django web application. When a user requests a particular URL associated with this view, this function will be called.
* Inside the function, a dictionary named fruits is defined with one key-value pair. The key is 'items', and the value is a list of fruits.

2)



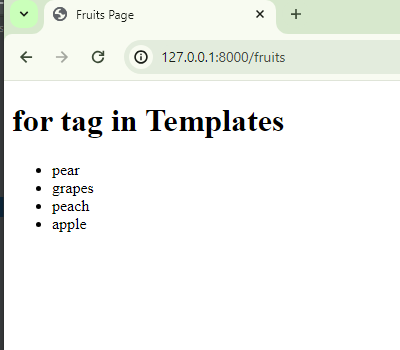
3)This is an HTML template file which will be rendered and returned as a response when the ForLooppage view function is called.

* + It starts with the usual HTML structure.
  + The <title> tag sets the title of the HTML page to "Fruits Page".
  + Inside the <body> tag:
    - <h1>for tag in Templates</h1> displays a heading.
    - {% for i in items %} initiates a loop over each item (i) in the items list passed from the view.
    - {{ i }} displays each item in the list.
    - {% endfor %} marks the end of the loop.
  + So, this template iterates over the list of fruits passed from the view and displays them as list items (<li>).

When a user accesses the corresponding URL, Django will render this HTML template, replacing template tags ({% ... %} and {{ ... }}) with the actual data, and return the resulting HTML page as the HTTP response.

4)Access the HTML Page:

After making these changes, start the development server using python manage.py runserver and access the HTML page as before. Django will now look for templates in the templates directory in the root folder of your project



{{ forloop.counter }}. This tag is used within a loop in Django templates to output the current iteration count of the loop.



OutPut:

