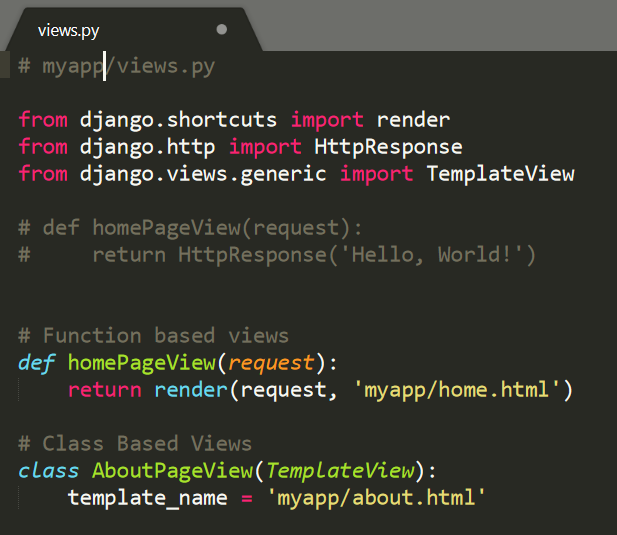
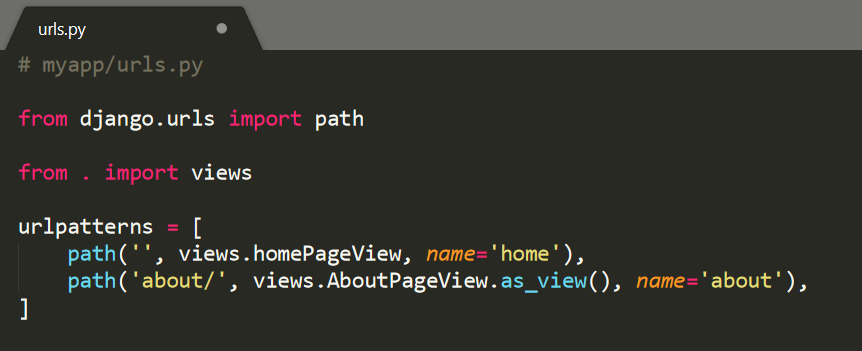
**Django - Views & Templates**

**Class-Based Views**

Early versions of Django only shipped with function-based views, but developers soon found themselves repeating the same patterns over and over again. Write a view that lists all objects in a model. Write a view that displays only one detailed item from a model. And so on. Function-based generic views were introduced to abstract these patterns and streamline development of common patterns. However, there was no easy way to extend or customize these views. As a result, Django introduced class-based generic views that make it easy to use and also extend views covering common use cases.





**Templates**

Every web framework needs a convenient way to generate HTML files. In Django, the approach is to use templates so that individual HTML files can be served by a view to a web page specified by the URL. It’s worth repeating this pattern since you’ll see it over and over again in Django development: Templates, Views, and URLs. The URLs control the initial route, the entry point into a page, the views contain the logic or the “what”, and the template has the HTML. For web pages that rely on a database model, it is the view that does much of the work to decide what data is available to the template. So: Templates, Views, URLs. This pattern will hold true for **every Django web page you make**.

By default, Django looks within each app for templates. In our myapp it will expect a home.html template to be located in the following location:

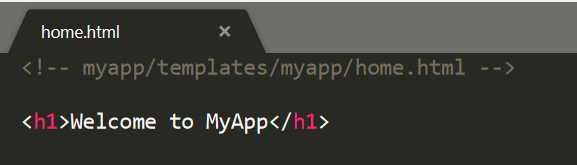
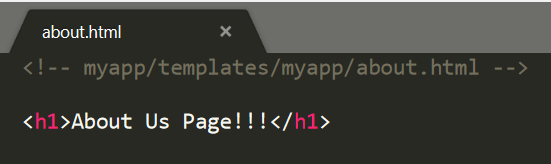
└── myapp

── templates

── myapp

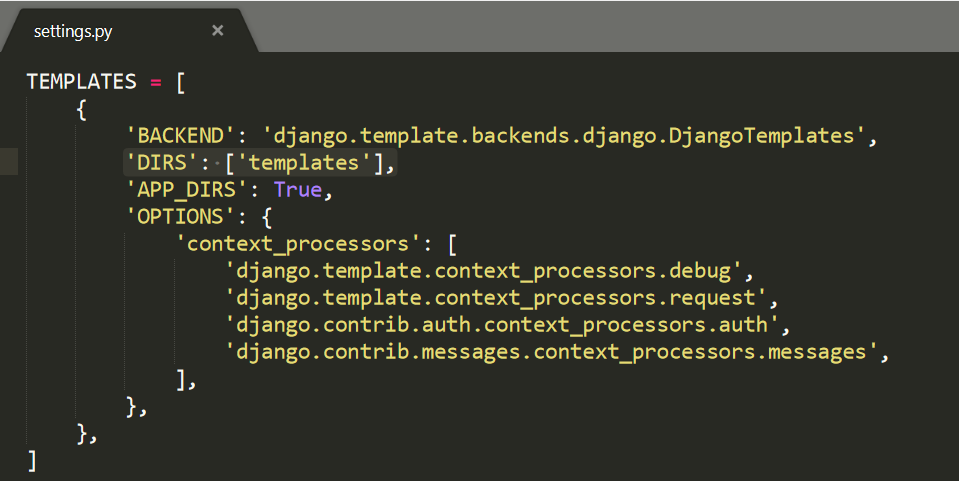
── home.html

This means we would need to create a new templates directory, a new directory with the name of the app, myapp, and finally our template itself which is home.html.

**Project-level Templates**

Another often-used approach to structuring the templates in a Django project is to instead create a single, project-level templates directory that is available to all apps. By making a small tweak to our settings.py file we can tell Django to also look in this project-level folder for templates.



The directory structure looks as below:

└── mysite

── templates

── myapp

── home.html

**Note**: It is completely up to the developer to opt for app level template hierarchy or project level template hierarchy.

Also, developer has freedom to choose between class-based views or function-based views.

**Extending Templates**

The real power of templates is their ability to be extended. If you think about most web sites, there is content that is repeated on every page (header, footer, etc). Let’s create a base.html file containing a header with links to our two pages.

