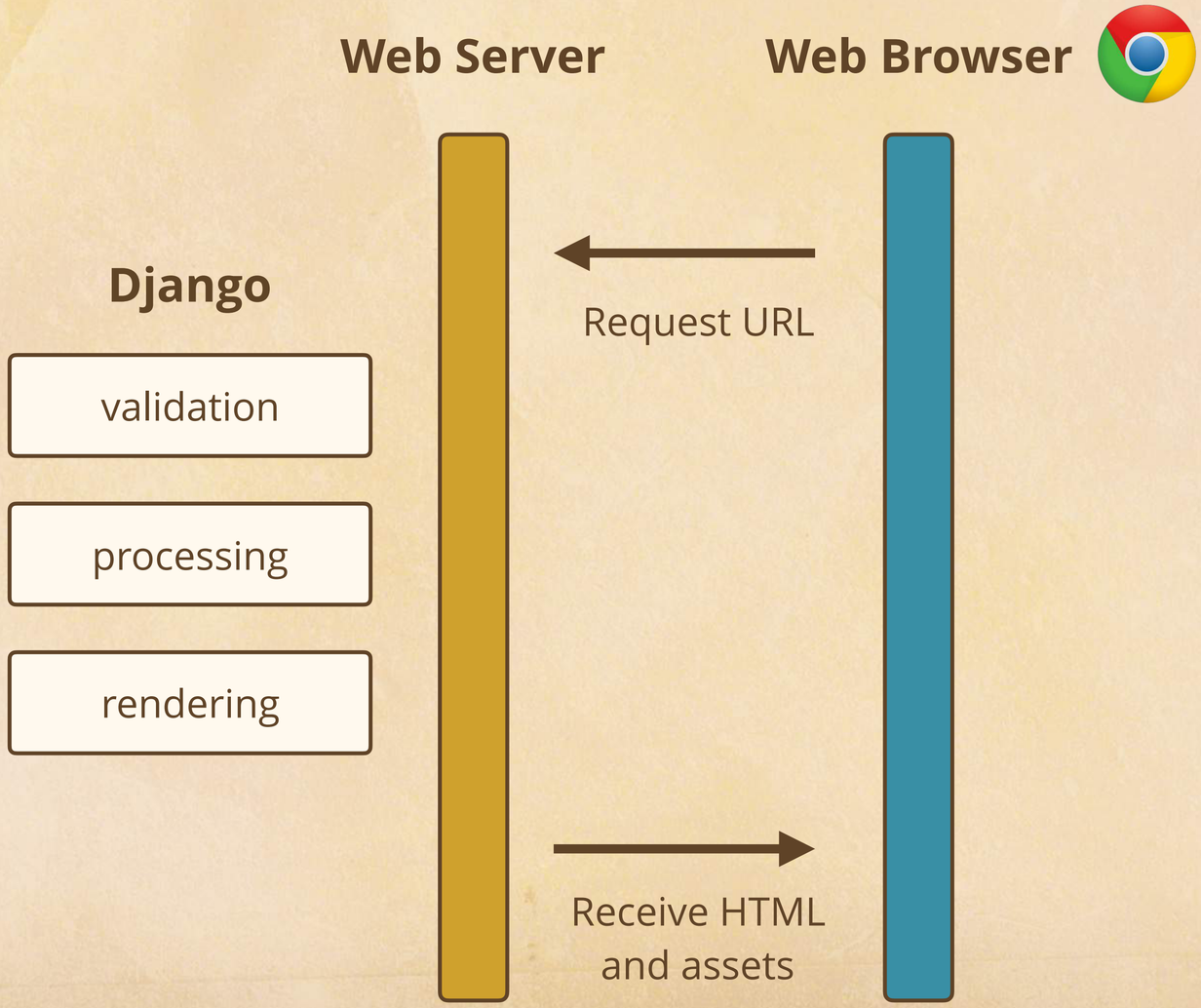
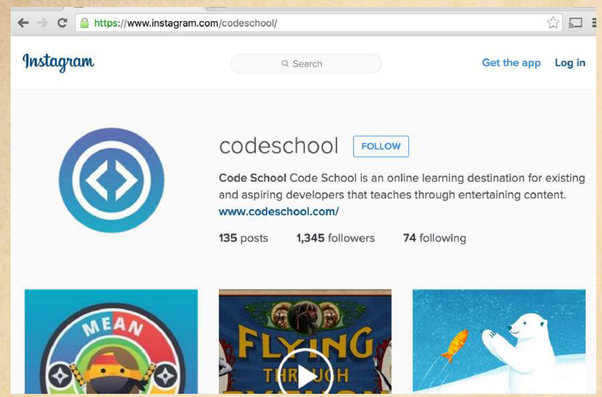
Django at a High Level

Today, most web applications including Django send data to the server to validate, process, and render HTML.



How Data Moves Through a Django App

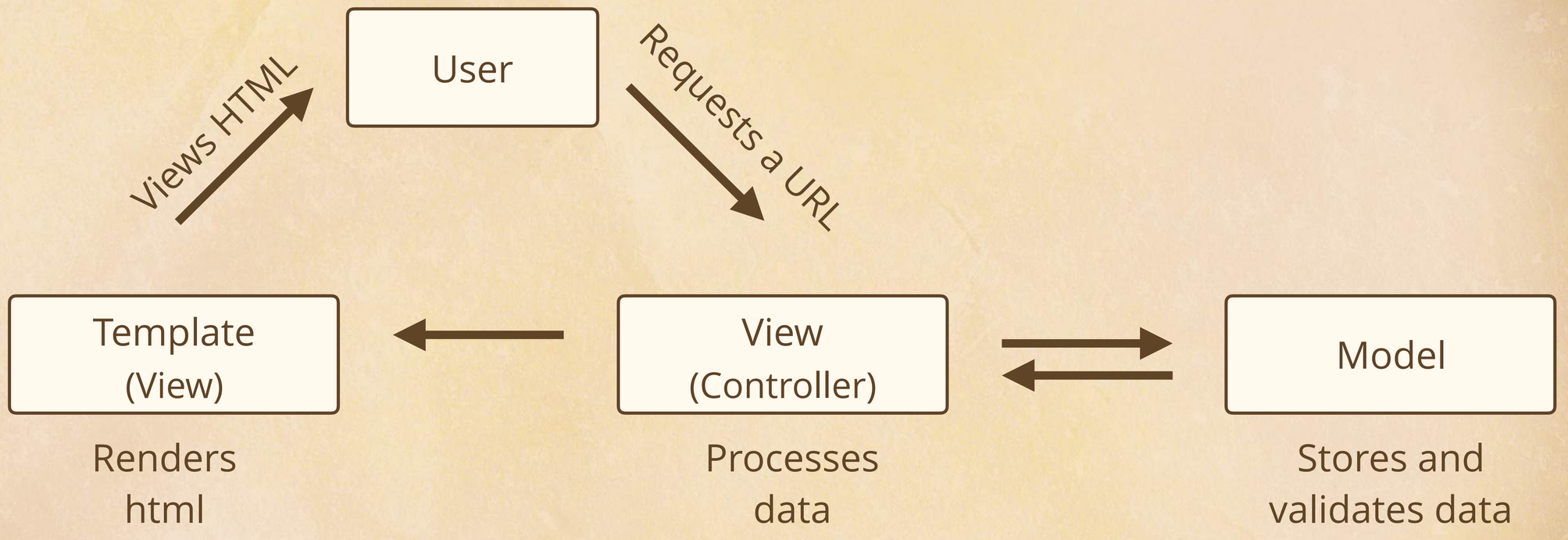


You could do these things in one script, but separating these components makes it easier to maintain and collaborate on the project!

TRY DJANGO

The Django MTV Framework

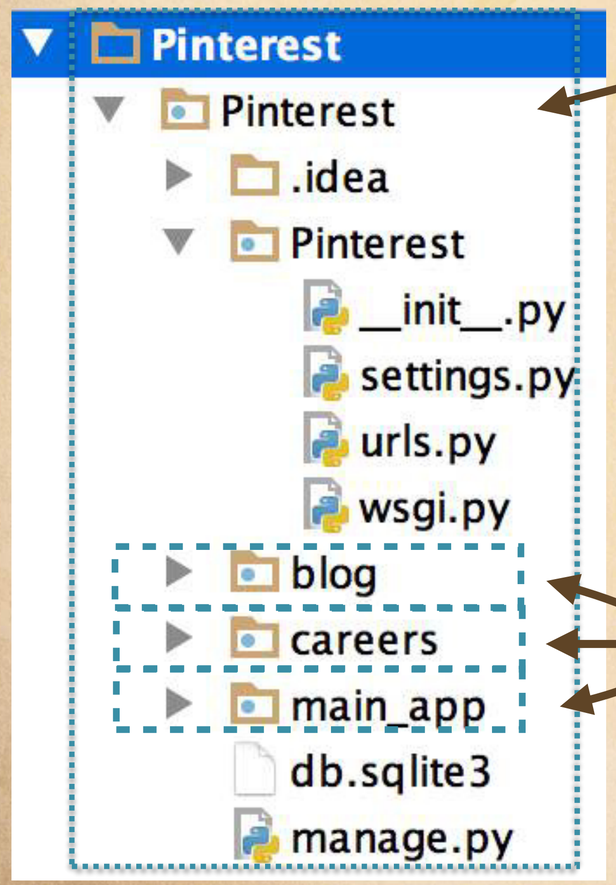
The validation, rendering, and processing is taken care of by separate components in Django: the model, template, and view (MTV).



If you're familiar with MVC, then you can think of the view as the controller and the template as the view, so MVC → MTV in Django.

Django Projects vs. Apps

Let's say we have a Django project, Pinterest, that has the .com, blog, and jobs pages as separate apps in Django.



The outer project has its related settings and files.

The apps inside have their own directory with related files.

Project

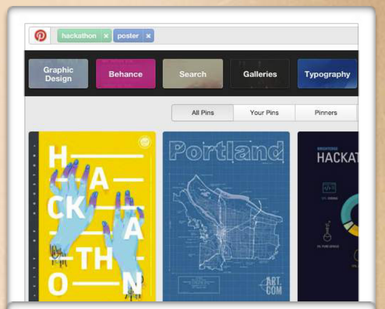
Pinterest

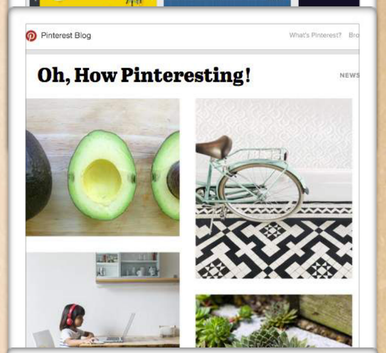
Apps

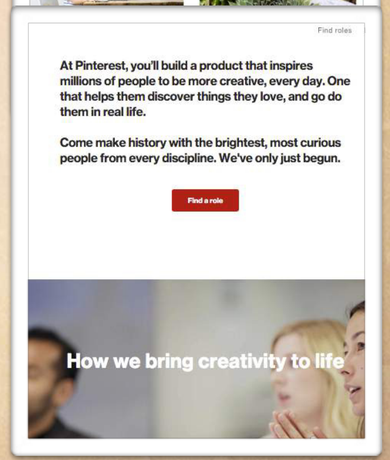
pinterest.com

blog.pinterest.com

careers.pinterest.com

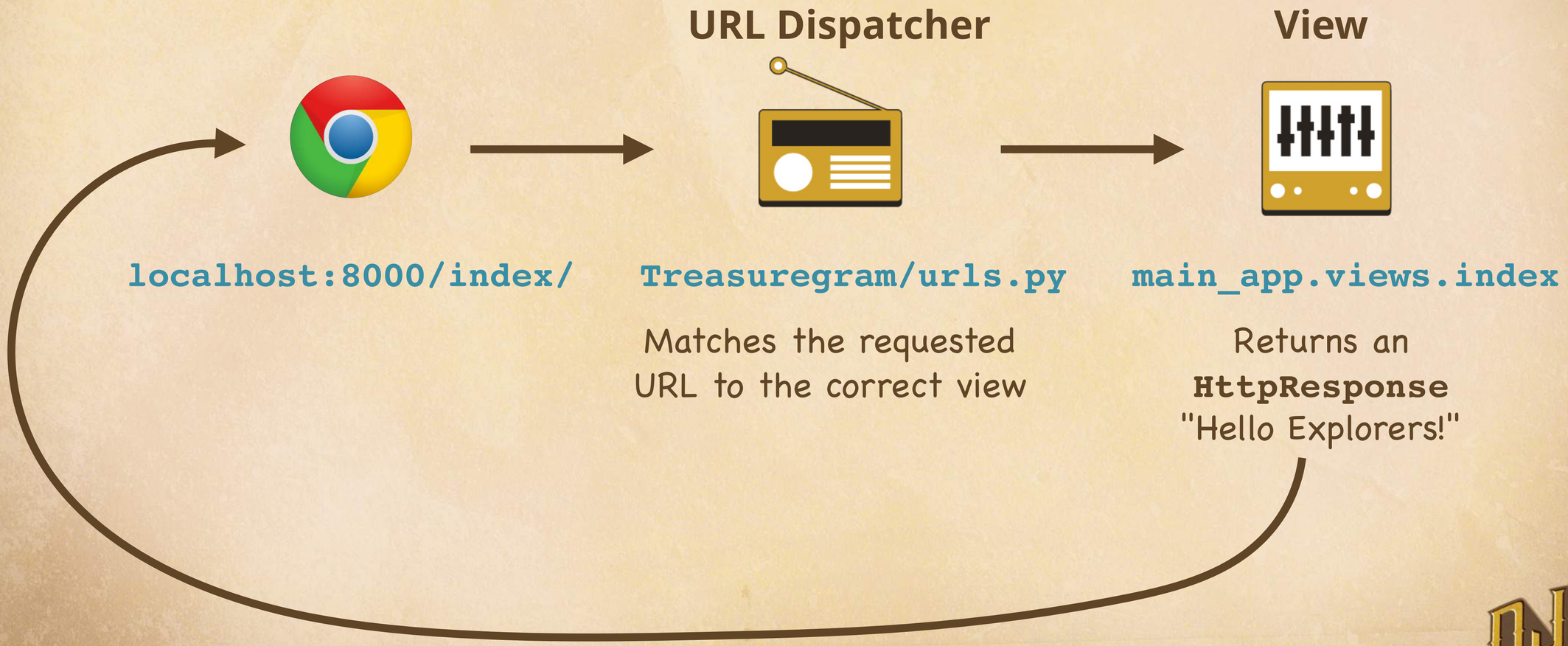






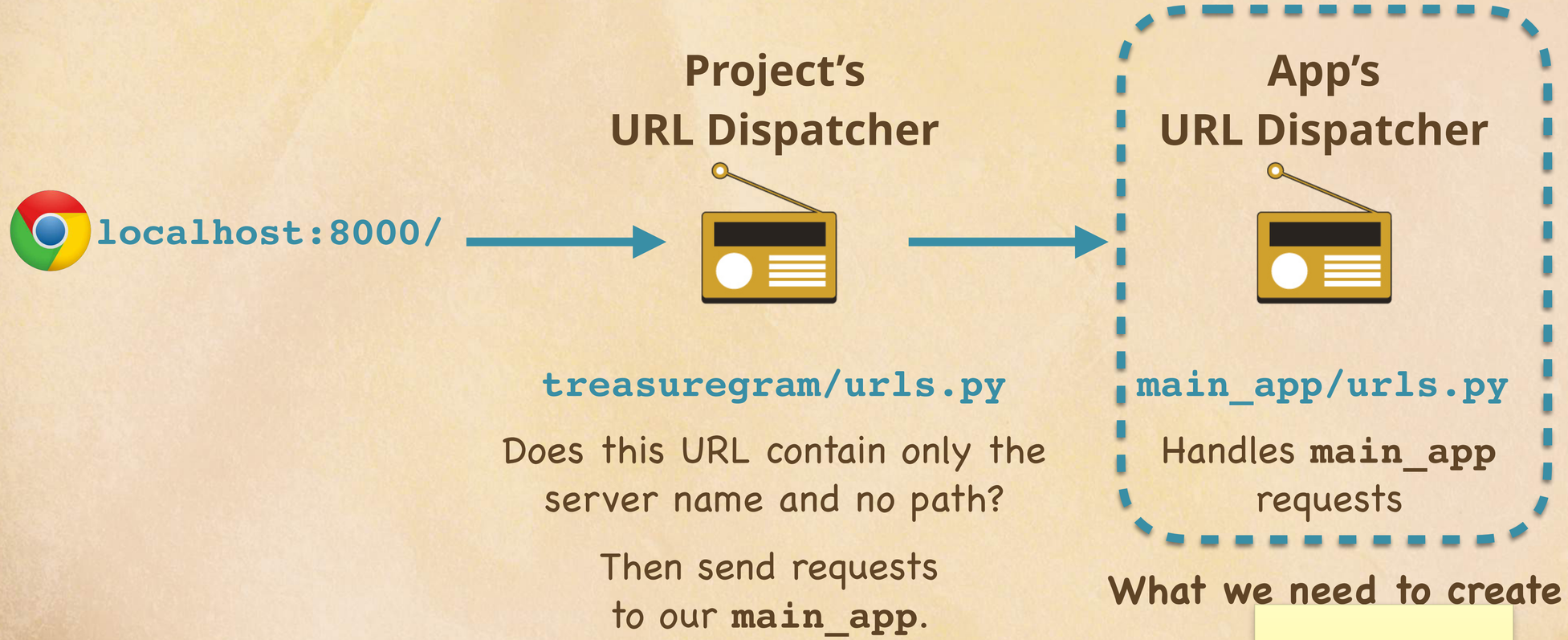
The URLs Dispatcher

We want the URL server/index/ to go to our index view that returns, “Hello Explorers!”



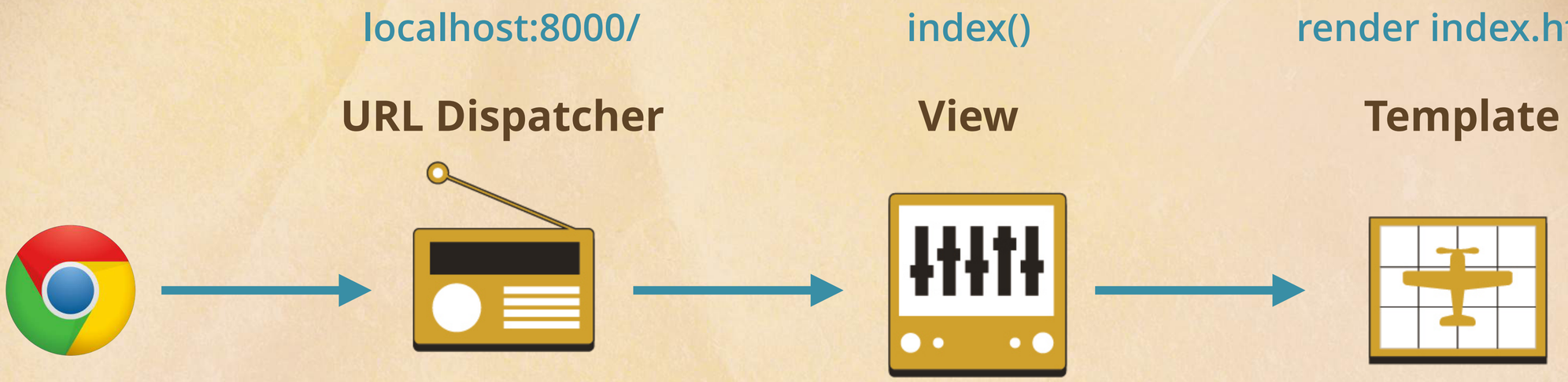
Best Practice: The App URLs Dispatcher

It's a best practice to have a project URL dispatcher ***and*** an app URL dispatcher.

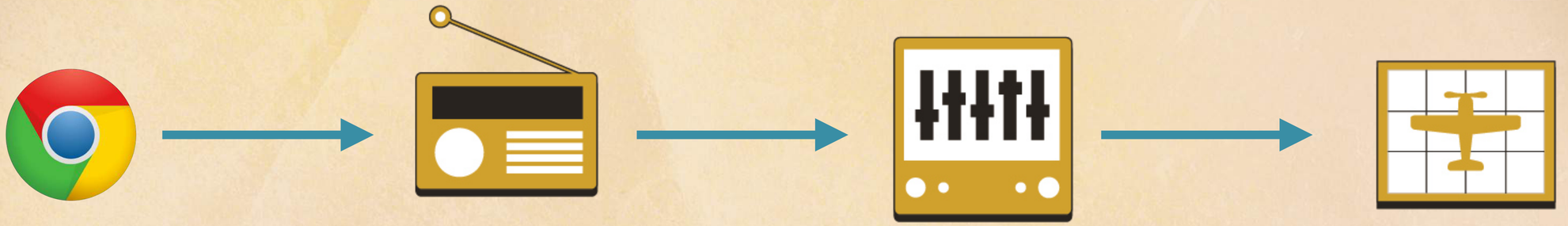


a little weird how the audio cuts out here, possibly need to say something about creating that now?

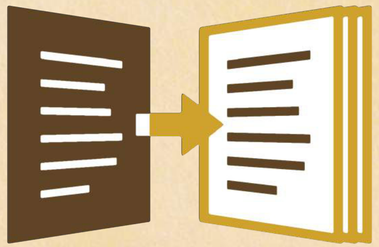
The URL-View-Template Process



Adding the Model to Our Django Flow



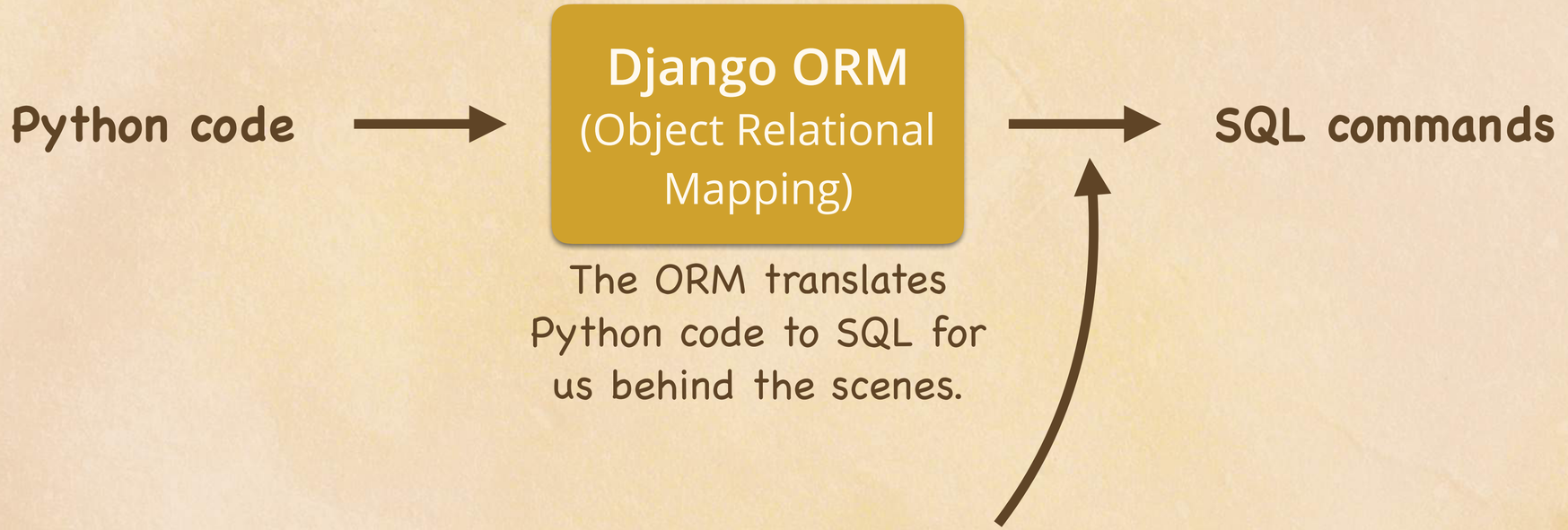
Model



Defines the data's structure and communicates with the database

A Model Is a Mapping to a Database Table

When we create a model and model objects in Django, we're actually creating a database table and database entries... but we don't have to write any SQL!



There's an extra step called **migrations** that needs to happen only right after you create a new model or update an existing model.