Model View Controller (MVC)

1.

The Model-View-Controller Design Pattern

All of the best web frameworks are built around the MVC concept.

Web Frameworks for Python

- A Web framework is a collection of packages or modules which allow developers to write Web applications without having to handle such low-level details as protocols, sockets or process/thread management.

- Frameworks provide support for a number of activities such as
  - interpreting requests (getting form parameters, handling cookies and sessions),
  - producing responses (presenting data as HTML or in other formats),
  - storing data persistently, and so on.

Web Frameworks for Python

- Since a non-trivial Web application will require a number of different kinds of abstractions, often stacked upon each other, those frameworks which attempt to provide a complete solution for applications are often known as **full-stack** frameworks in that they attempt to supply components for each layer in the stack.

- Projects provide the base "application server" are called **non full-stack** frameworks.

- For the purposes of this course we will use non full stack frameworks (like Bottle or Flask)

*What if I were to tell you that building a web application is exactly like building with **Legos**?*

It all starts with a request...

- In the case of the Legos, it was your brother who asked you to build something.
- In the case of a web app, it's a user entering a URL, requesting to view a certain page.
- So your brother is the user.

The request reaches the controller...

- With the Legos, you are the **controller**.
- The controller is responsible for grabbing all of the necessary building blocks and organizing them as necessary.

Those building blocks are known as models...

- The different types of Legos are the **models**. You have all different sizes and shapes, and you grab the ones you need to build the spaceship.

- In a web app, models help the controller retrieve all of the information it needs from the database.

So the request comes in...

- The controller (you) receives the request.
- It goes to the models (Legos) to retrieve the necessary items.
- And now everything is in place to produce the final product.

The final product is known as the view...

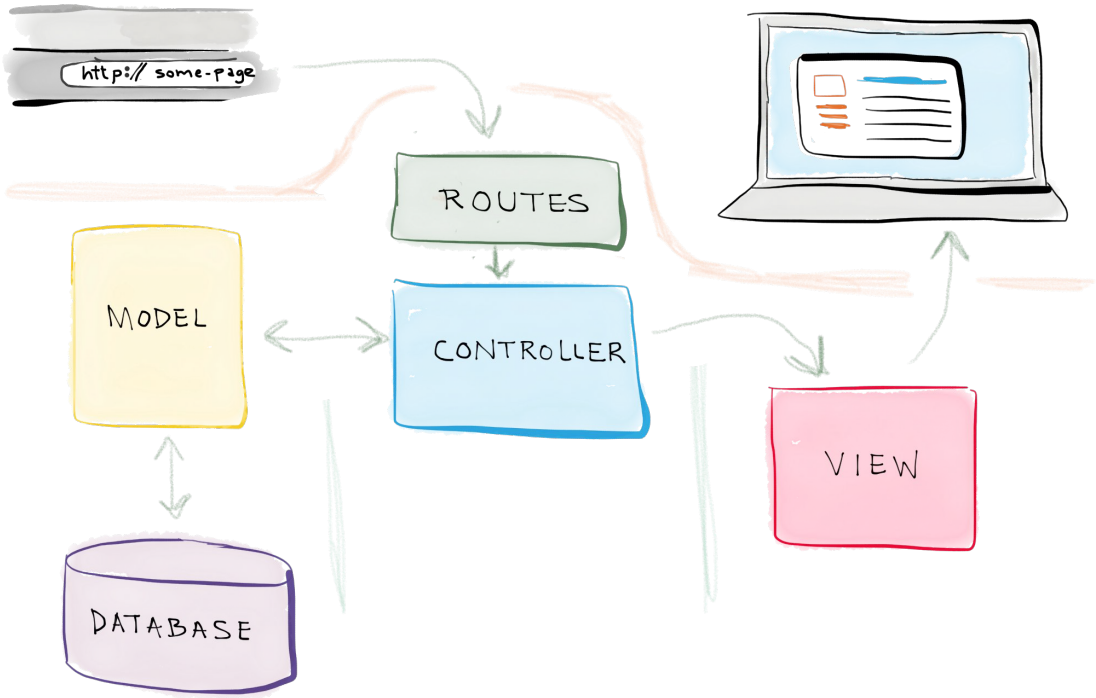
- The spaceship is the **view**. It's the final product that's ultimately shown to the person who made the request (your brother).
- In a web application, the view is the final page the user sees in their browser.

To summarize...

- Your brother makes a request that you build a spaceship.
- You receive the request.
- You retrieve and organize all the Legos you need to construct the spaceship.
- You use the Legos to build the spaceship and present the finished spaceship back to your brother.

And in a web app:

1. A user requests to view a page by entering a URL.
2. The Controller receives that request.
3. It uses the Models to retrieve all of the necessary data, organizes it, and sends it off to the…
4. View, which then uses that data to render the final webpage presented to the user in their browser.



From a more technical standpoint

- With the MVC functionality summarized, let's dive a bit deeper and see how everything functions on a more technical level.

- When you type in a URL in your browser to access a web application, you're making a request to view a certain page within the application. But how does the application know which page to display/render?

- When building a web app, you define what are known as **routes**. Routes are, essentially, URL patterns associated with different pages. So when someone enters a URL, behind the scenes, the application tries to match that URL to one of these predefined routes.

-

From a more technical standpoint

- The **model(M)** is a model or representation of your data.
- It's not the actual data, but an *interface* to the data.

- The model allows you to pull data from your database without knowing the intricacies of the underlying database.

- The model usually also provides an *abstraction* layer with your database, so that you can use the same model with multiple databases.

From a more technical standpoint

- The **view(V)** is what you see. It's the presentation layer for your model.
- On your computer, the view is what you see in the browser for a Web app, or the UI for a desktop app.
- The view also provides an interface to collect user input.

From a more technical standpoint

- The **controller(C)** controls the flow of information between the model and the view.
- It uses programmed logic to decide what information is pulled from the database via the model and what information is passed to the view.
- It also gets information from the user via the view and implements business logic: either by changing the view, or modifying data through the model, or both.

Big concept

So, in fact, there are really four major components in play: **routes**, **models**, **views**, and **controllers**.

Thanks!

Any questions?

You can find me at:
akolovou@di.uoa.gr

Credits

- https://realpython.com
- https://djangobook.com
- https://wiki.python.org