Monday: MVC Architecture with Angular

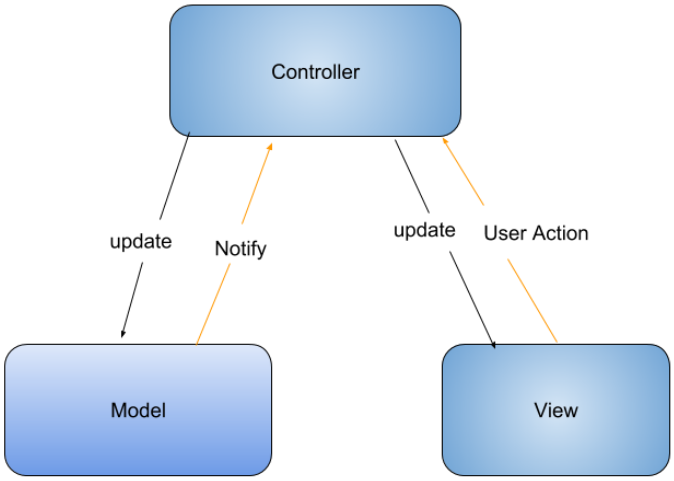
One of the popular ways of organising application is using the **MVC architecture**. Angular uses this architecture to organise its application. Let's look at what is MVC architecture in Angular.

Model-view-controller, or MVC as its popularly known, is a software design pattern for developing web applications. This design pattern is made up of three parts:

1. **Model**: Responsible for maintaining data
2. **View**:Responsible for displaying data to the user
3. **Controller**: Responsible for controlling the interaction between the Model and Views

The MVC design pattern separates the application's logic from the user interface.

The MVC concept can be represented graphically as follows:



Let's walk through what's happening in the diagram. Assume that you have an application that tracks your goals. Each time you delete or add a new goal, the controller updates the model by either deleting or adding a goal. In other words, the controller updates the model as per the user input.

The model then notifies the controller of the changes, which in turn updates the views to display the updated information. When you add your goals and the model is updated, the controller updates the view which displays the new goal on your site.

Let's look at each part separately and how they function:

Model

This is the application's data structure. It represents the actual data that an app deals with, and it responds to requests from views and instructions from controllers to update itself. Moreover, a model

does not depend on either view or controller. In Angular applications, models are represented by objects. In the example of the goal application from earlier, our model would be:

```
class Goal{
  constructor(){}
}
```

Don't worry about mastering the code right now - we'll be creating applications soon enough. For now, focus on what each part represents, and how they relate to one another.

View

Views are what is presented to the user and how they interact with it. In other words, it's what the user actually sees on their screen, and can be made with HTML, CSS or Javascript. It displays the model data. In Angular, we can create a view in HTML by using double curly braces.

```
<h4 id="{{i}}" appStrikethrough> {{goal.name}} due on {{goal.completeDate|date|uppercase}} </h4>
```

Controller

The controller is essentially the glue between the model and the view. It updates the view when the view changes and also adds event listeners to the view. Additionally, it updates the model when the user manipulates the view.

Since the controller links both the model and the views, it can be separated into two; view controllers and model controllers

View controllers are responsible for pulling together the model used by the view and handling the input from the user of the view. For example, if you wanted to delete a goal from your goal application, you would have an event listener in the HTML that listens for when the user clicks the delete button and then a function that actually deletes the clicked goal.

goal.component.html

```
<button class="btn btn-danger btn-sm" (click)='deleteGoal(i)'>Delete Goal</button>
```

In the example above the event listener is `(click)` and the function that deletes the goal is `deleteGoal(i)`. For now, do not worry about how to create the function, we'll explain that later as we look at angular in depth. The important thing to understand for now is how the event listeners work together with functions as view controllers.

On the other hand, model controllers contain the data to be displayed as well as the data to be collected as input in forms. It can also be functions that are invoked based on the user's activity such as clicking a button or making changes to the model data. For example, a function that adds a new goal to the model. Whenever you add a new goal through a form, there is a function that is responsible for adding that goal to your database. This function is a model controller because it interacts directly with your model.