A Developer Diary
{about:"code learn and share"}

October 9, 2015 By Abhisek Jana — 1 Comment (Edit)

How to Learn and Master Angular easily – Part2

Welcome to “How to Learn and Master Angular easily – Part2” !

In this chapter, we will start with understanding Angular Architecture, then move on to individual components such as `Service`, `Controller` & `View`. We will work on live demo and full working code base. We will also learn about few important concepts like `Dependency Injection` and `Two Way Binding`.

In Part 1 of the “The Best Way to Master Angular JS” we read about the background, objective etc and then started with the HTML template of the TaskTracker application.

Why Angular JS ?

Question: With the help of JQuery & vanilla JavaScript we can easily develop an application, so why do we even need Angular or any other such framework?

Here are few good points to be noted, I am not going to have the full list here since many of it may not make sense to you at this moment.

1. **Maintainability :** Angular helps to structure the code so that the maintainability becomes very easy. This is the most important point you

need to keep in mind when you start your development.
2. **MVC :** Angular supports the MVC framework at the front end. This makes our life much easier.
3. **Reusability :** Most of the Angular JS components are reusable. Anything you develop,even the views are fully reusable
4. **Rapid Development :** Angular itself is performing many heavy lifting and since everything is reusable in Angular, the development timeframe can be shorten significantly by experienced Angular developers.

There are many other functional advantages, we will discuss about them as you follow along.

We will talk about the architecture of Angular JS in a moment, but lets understand few important points on modern web development. You may know that with Angular we will always develop a Single Page Application (SPA) and we don't have the refresh the entire application. So what else has been changed since the edge of Struts,JSF,Spring MVC?

1. **Better Standard & Fast Browser :** The browser is getting more and more powerful everyday. Before Chrome & Firefox became popular we used to develop web applications in IE4, IE5. At that time we had to render the entire page at server end. However now the ECMA5 & ECMA6 standards are powerful enough to do the heavy lifting.
2. **MVC :** We all learnt the popular MVC framework at some point and most of us used that at the server end, since the server used to render all the html content. However the modern Java Script Libraries like Angular supports the MVC framework which made the server side development of the presentation layer light, easy and service oriented.
3. **JSON :** JSON (Java Script Object Notation) is another very powerful element. JavaScript now can read/parse the JSON very easily, the communication with the server has become the most simple task.
4. **Debugging Tool :** Earlier the browser didn't have any development tool, with Fire Bug and Chrome Development tools it has become very easy to

debug the Java Script.

Angular Architecture:

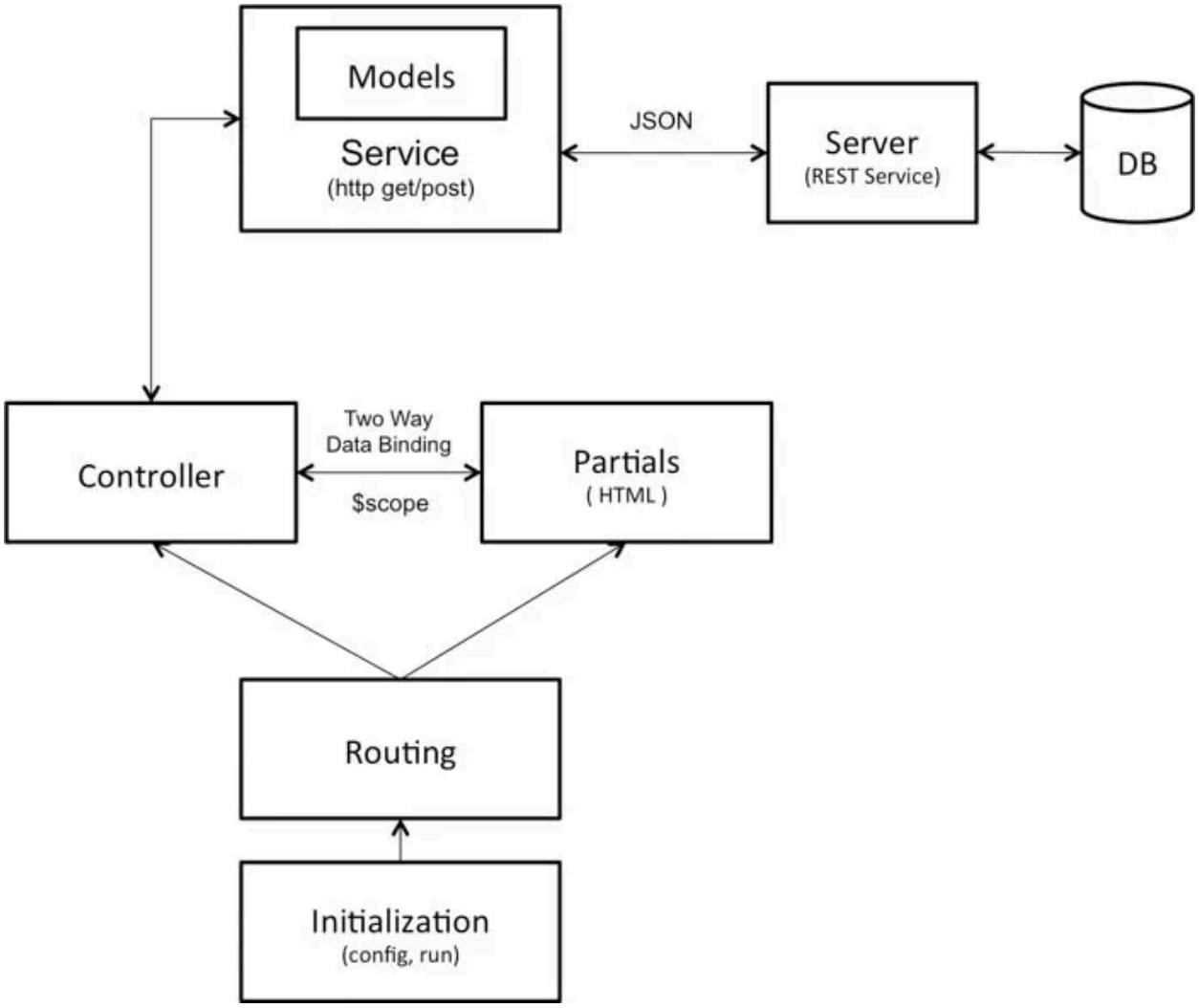
One thing I want to make sure is not to explain everything at once, the below picture has components which you need to understand now and not everything.

Angular has few integral parts which we need to understand to get started.

1. Service
2. Controller
3. View (Partials)
4. Routing

Again, the above list does not have everything, however sufficient for our purpose for time being.

Here is the high level architecture diagram of Angular JS 1.x, based on above components.



We can see how individual components are interacting with each other. Lets learn more about them. We will take the top down approach.

Service:

We need to get/push the DATA loaded from/to the server. The Angular Service is meant for that. The only important characteristics to be mentioned about `Service` is its persistent behavior. The model objects inside the service are static in nature so once the data is loaded into the model you can access it from anywhere. In nutshell, service acts like a data store and its encapsulates the server side interaction.

What to have in service?

1. HTTP POST/GET methods
2. Model Objects

Now, Lets see an example on how to create a service in angular.

```
var module = angular.module('TaskTracker',[]);
module.service('TaskService',function($http){
    var tasks=null;

    this.getTasks=function(){
      if(tasks!=null){
        return tasks;
      }else{
        $http.get('data/data.json')
          .success(function(response){
              tasks=response;
              return tasks;
          })
          .error(function(data){
              return "Error";
          });
      }
    };
});
```

At the first line we have created a module then added the service to the module. We will learn about module in the next chapter.

We can create a Service just by calling the `service()` function of any module. The first parameter is the name of the Service and 2nd parameter is a function.

The name of the Service is `TaskService`.In order to expose any method as part of the service we can use `this` keyword and add the function to this.Then that function will be available to be executed from outside of the service.

The `getTasks()` method is exposed to fetch the list of tasks from a json file. If the tasks variable is null then service will fetch the data from backend.

`$http` object was used to make the http get service to call backend.The $http service is a core Angular service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP. If the remote call is successful then we will store that in our `tasks` variable otherwise it will just return "Error".

Remember, anything starts with a `$` is a native angular object/service.

In order to invoke the `getTasks()` method from the `TaskService` service you need to execute following line.

```
var tasks=TaskService.getTasks();
```

It is that simple! The variable tasks is the model object which is going to hold the data.

Just that you know, there are 4 ways to create a Service in Angular. Factory is another popular one. Essentially all of them kind of do the same thing.

In MVC the M=Model is the service here.

Dependency Injection

Notice, while we are calling the `TaskService.getTasks()` we are not passing the object named `$http` into the method `getTasks()`.

Question: So how the service can get the instance of $http service?

Angular can automatically detect a dependent Angular Component (Service, Module, Controller etc) and inject it automatically to the specific Component. This is called `Dependency Injection`.

If you need to pass any specific data like id or name you still need to pass that as part of the getTasks() method. If you are confused don't worry, we have many more examples to come in order to understand this.

Controller:

The Controller is responsible to interact with the Service (Model) and the View (Partials/HTML). In simple words, Controller makes service call to get/push data and passes that data to the HTML in order to render the view. You can refer the diagram above to get a clear view.
So Controller in Angular JS is mainly responsible for:

1. Make Service Calls
2. Interact with the View
3. Initiate Routing Change

You have already seen how we can make a service call from Controller in the Service section. Lets get that code here and see how to create an Angular Controller.

```
var module = angular.module('TaskTracker',[]);

module.controller('TaskController',function(TaskService){

  var tasks=getTaskList.getTasks();

});
```

I have given `TaskService` as the parameter in the `TaskController`, since the TaskService is an Angular Component, it will be automatically injected in the Controller (Dependency Injection).

So now you know how to invoke a service from a controller. Next we will learn how to interact with the View (HTML) from the Controller.

In order to share the variables between Controller & Partials (HTML) Angular JS has a default variable named `$scope`.

`$scope` is like a global Object which is accessible to the View (Partials). Anything you put in `$scope` will be accessible to the View. If you are coming to JEE background, `$scope` is exactly same as the request object we use in our JSP. So from Servlet/ActionClass we had to load everything into the request object so that the data can be accessible from the JSP.

In our previous example, we have stored the task list in a local variable called tasks in the Controller, however var tasks wont be available to the View. So we need to store it in the `$scope` variable. Lets jump into the code.

```
module.controller('TaskController',function($scope,TaskService)
{

  $scope.tasks=getTaskList.getTasks();

});
```

We had to add `$scope` as the function argument so that Angular can inject it in our `TaskController`.
Now the tasks variable will be automatically available to the View. Similar way any function call from view (HTML) needs to be implemented in the Controller as well. We will see an example of that in the View section.

View :

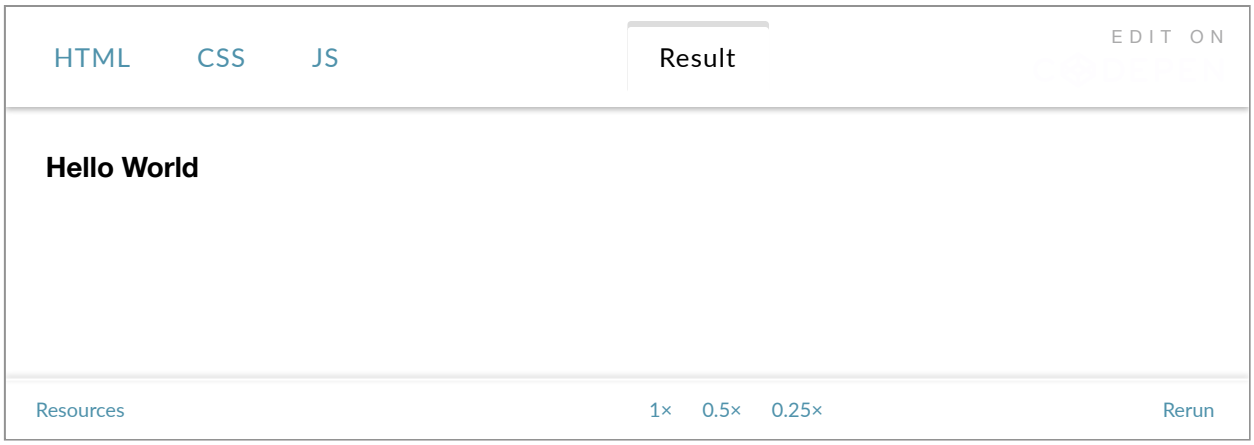
You might have noticed, I am using the name View, Partials, HTML interchangeably since all of them carries almost same meaning. If you have worked with handlebars or mustache then you can correlate Partials with the HTML template. In case you are from the Java world, the View (Partial) is nothing but the JSP where we manipulate the HTML with the help of scriptlet.

Lets see an example of the View.

Note: All the examples here in this chapter are on view, since partials generally stays in another file and in runtime they get added dynamically. (Something like a template we use in JQuery) Partials are not really different than a simple view, so lets understand View here and then in later chapters will learn more on partials.

```
{{message}}
```

In the example above, I have added message to the `$scope` variable which will be available to the view. `{{}}` is the syntax for the template you can use to access the data from the `$scope` variable. So `{{message}}` will be replaced by Hello World.See the demo below:



`ng-controller` is an angular element where we need to specify the name of the Controller for this view.Since there will be many Controllers, the View needs the Controller name to bind the `$scope` object.I have assigned `MyController` as `ng-controller`, so any code inside the `div` will reference the `$scope` variable of the `MyController` Controller.

NOTE: Any Angular specific element like `ng-controller` is called as `Directive`.We can create our own custom directive.

Two Way Binding :

The `$scope` will be automatically updated from both Controller & View, means if you change any value of `$scope` in the Controller it will be automatically be updated in the View and vice versa. This is called `Two Way Binding`, another very important function of Angular. Lets see how it actually works.

```
{{message}}
```

The `ng-model` is another directive through which you can bind an HTML element to its $scope variable.

Here in the above example we have assigned the `message` to the `ng-model`. So whenever we change the text it will automatically update the `$scope.message`.

Since our `label` is displaying the `$scope.message` you can see the updated value as it changes.

Initially the Text and the Label both will display Hello World, once you change the text the label will automatically be updated. Check out the demo below.

Two Way Binding is a very important functionality, see how we can update data without any onChange() function or onClick() function. It has its own drawback, however most of the cases its very useful.

Going back to View, now lets see how to invoke a Controller function from View.

```
Show Message
{{message}}
```

`ng-click` is a directive which calls a function when the attached element has been clicked. Here we have attached a function name `showMessage()` with the $scope of the `MyController`, then called it using `ng-click()` from our view.

We are updating the `$scope.message` from showMessage() which will automatically be displayed in the `label` (Two Way Binding)

`ng-change`, `ng-dblclick`, `ng-keyup` are also similar directives like `ng-click`. They also take an expression or function with arguments.

Here is the live demo:

I will discuss about routing later, once we get to know more about the service, controller and partials.

In next chapter we will learn about Angular module, Project Folder Structure etc.

Related

How to Learn and Master Angular easily – Part1
In "Angular 1.x"

How to Learn and Master Angular easily – Part3
In "Angular 1.x"

How to use Filter in Angular JS
In "Angular 1.x"

Filed Under: Angular 1.x, JavaScript Tagged With: Angular, Angular Architecture, Angular JS, Code, example, JavaScript, JSON, Learn, master, Programming, tutorial

Subscribe to stay in loop

* indicates required

Email Address *

Subscribe

Comments

Alex says
February 13, 2016 at 5:04 pm

(Edit)

Just before the View title, you have a code snipplet:

$scope.tasks=getTaskList.getTasks();

where da heck did getTastList come from? Shouldn't it be TaskService.getTasks() instead?

Reply

Leave a Reply

Logged in as Abhisek Jana. Edit your profile. Log out? Required fields are marked *

Comment *

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.