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

October 14, 2015 By Abhisek Jana — 2 Comments (Edit)

How to Learn and Master Angular easily – Part6

This is the last part of our series **How to Learn and Master Angular easily – Part6**. We will complete our TaskTracker application and will also learn about Angular filter, $rootScope$,watch etc.

In Part 5 we went through Routing in detail using ui-route and implemented that in the app.

Let's try to finish our application with the knowledge we have gained so far. We will start with creating the New Label.

At first we need to create a partial (View/Template) for the New Label.

1. Create a new file named `newLabel.template.html` inside `\app\label\partial`.
2. Copy the element with `id='page-body'` from the `createLabel.html` to the `newLabel.template.html` file.
3. Now you can delete the createLabel.html file.

We will create a new controller for the header named `headerController`.

4. Create another js file named `header.js` inside `/app/header` and create a controller named `headerController`.
5. Create a new function named `createNewLabel()` and inside that change the route to `newLabel`.
6. Add the `header.js` in `index.html`.

```
angular.module('taskTracker')
    .controller('headerController',
    ['$scope', '$state',
        function ($scope, $state,
$stateParams,$rootScope) {
            $scope.createNewLabel=function(){

$state.transitionTo('newLabel',$stateParams,
{reload:true});
            };
}]);
```

7. Now define the `headerController` as `ng-controller` inside the element with id `page-content-wrapper` in the index.html.
8. Call the `createNewLabel()` function in `#btnCreateLabel` using `ng-click` in order to capture the click event.

```
...


...
```

```
...
```

In summary all we have done is to change the state to `newLabel` from the `headerController` when the `#btnCreateLabel` is clicked.However we don't have the state defined for showing the `newLabel.template.html`. Let's work on that.

Add new state named newLabel in the label.js.

```
module.config(['$stateProvider',
        function($stateProvider){

        $stateProvider.state('newLabel',{
                url:'/newLabel',

templateUrl:'app/label/partial/newLabel.template.html',
                controller:'newLabelController'
        });
}])
```

Add the new controller named `newLabelController` in the label.js.

Now if you open index.html in browser and click on "New Label", the New Label partial (view) should be displayed.

In this New Label we can enter a Label Name and select a color for the New Label then save it.Here are the changes we need to make in the `newLabel.template.html`.

1. Add the `ng-model="name"` in the text field to bind the field with the model named `name`.
2. Then replace `onclick()` function in the `Save` button with `ng-click="saveLabel()"`
3. In order to store the color value we will define a model named `color` in our `newLabelController` and based on the selected color we will display the tick icon. Let's call a method named `setColor()` from each of the buttons for setting the color and show/hide the tick icon using an expression in `ng-show` directive. Here is the changed code. We will pass the name of the css class in the `setColor()` function.I already have a css class defined for each color in the `style.css`
4. So once you click on the color button, thd `setColor()` will be called to set the `$scope.color` value, then the tick will be displayed based on the new `$scope.color` value. This will work automatically since Angular has Two Way Binding !

```
Color :
```

Here is the code for the `newLabelController` that we created.

```
module.controller('newLabelController',
    ['$scope','$state','$stateParams','dataService',
    function($scope,$state,$stateParams,dataService){

        $scope.name="";
        $scope.color=""

        $scope.setColor=function(value){
            $scope.color=value;
        };

        $scope.saveLabel=function(){

            var label={
                "name":$scope.name,
                "color":$scope.color
            };

            dataService.addNewLabel(label);

            dataService.setSelectedLabel($scope.name);
            $state.transitionTo('showTasks',$stateParams,
{reload:true});
```

```
        };

    }]);
```

In `newLabelController` we are creating label object using the `$scope.color` and `$scope.name` then calling a method in the `dataService` named `addNewLabel()` to add the New Label to the label array. Then we are setting the selectedLabel as the created label and changing the state to `showTasks`, which will display the list of tasks available for the new label (which will be empty).

This is the code to add the label object to the label array present in the `dataService` service.

```
this.addNewLabel=function(label){
        arrLabels.push(label);
};
```

Now open the index.html and click on New Label then add the name of the New Label and click on a color. Click on Save. You should have the label created and the taskList should be loaded for the label.

I will have you work on the “Add Task” button, which is very similar to the New Label Function. Here is the list of steps and code for your reference.

1. Create a new partial named `newTask.template.html` inside `/app/task/partial` and paste the dynamic content from the `createTask.html`.
2. Delete the `createTask.html` file.

3. Now create a `createNewTask()` function inside `taskListController` to change the state to `newTask`.
4. Add a new state named `newTask` and create a new controller named `newTaskController` inside `task.js`.


```
module.config(['$stateProvider',
'$urlRouterProvider',function($stateProvider){
        ...
// Add the new state named newTask
.state('newTask',{
            url:'/newTask',

templateUrl:'app/task/partial/newTask.template.html',
            controller:'newTaskController'
        });

}]);

// add the createNewTask() function to change the state.
//The other way of doing this is using ui-sref
module.controller('taskListController',
['$scope','dataService','$state','$stateParams'
    ,function($scope,dataService,$state,$stateParams){
 ...
$scope.createNewTask=function(){
        $state.transitionTo('newTask',$stateParams,
{reload:true});
```

```
    };

}]);

// Add the newTaskController to save the Task.
module.controller('newTaskController',
['$scope','$state','$stateParams','dataService',
        function($scope,$state,$stateParams,dataService)
{

        $scope.name="";
        $scope.due_date="";
        $scope.labelName=dataService.getSelectedLabel();


        $scope.saveTasks=function(){

            if($scope.name.trim()!="") {

                $scope.task = {
                    "id": (new Date()).getTime(),
                    "name": $scope.name,
                    "due_date": $scope.due_date,
                    "completed": false,
                    "labelName": $scope.labelName

                };
                dataService.addNewTask($scope.task);

                $state.transitionTo('showTasks',
$stateParams, {reload: true});
            }
        };
```

```
}]);


    this.addNewTask=function(tasks){
        arrAllTasks.push(tasks);
    };
```

5. In the `newTask.template.html` file, replace onclick of the Save button to `ng-click="saveTasks()"`.
6. Add `ng-model="name"` in the textbox for Task Name and `ng-model="due_date"` for the Due Date text box.

Note : I have not implemented the date or note field in the New Task view. This is something you may want to work on and comeplete.

Save all the files and test the code now. You can always find the full codebase in github, link given at the end of the article.

In the `taskList.template.html` we have a text box on top of the Task Lists to create task just by entering the task name and then pressing enter/return. We would need to create a method named `createQuickTask()` inside `taskListController`.

```
// Add this inside taskListController.

$scope.tasks=[];
$scope.txtTaskName="";

$scope.createQuickTask=function(){

        if($scope.txtTaskName.trim()!=""){
                $scope.task={
```

```
                    "id":(new Date()).getTime(),
                    "name":$scope.txtTaskName,
                    "due_date":null,
                    "completed":false,
                    "labelName":$scope.label
                };
                dataService.addNewTask($scope.task);

$scope.tasks=dataService.getTasksForLabel($scope.label);
                $scope.txtTaskName="";
        }
};
```

In the `taskList.template.html`, bind the textbox with `$scope.txtTaskName` and call the `createQuickTask()` method on pressing enter. Here is the code :

```
Create New Task
```

We are using a new directive here named `ng-keyup`. The `$event.keyCode` provides the ACSII code for the key pressed. In this instance we will validate whether it's 13, which is the ACSII code for enter key, then call the `createQuickTask()` method.You can see expression can be very useful.

Now test the code by entering a task name and press enter. The new task should be added automatically.

in case you haven't noticed, there is a filter textbox in the header of our Task Tracker application. The list of tasks displayed should be filtered by the text

that has been entered into the filter textbox.

In order to implement this, we need to first capture any text entered in the filter textbox then pass the details to the `taskListController`, then apply a “filter” to display only teh tasks name matches with the entered text in Filter.

In index.html lets bind a model named `txtFilter` with the filter textbox and call a method named `updateFilter()` on `ng-change`

```
        ...
```

```
        ...
```

Since the header already has a controller named `headerController` we are going to create our updateFilter() there.

```
$scope.txtFilter="";

$scope.updateFilter=function(){
        $rootScope.filter = $scope.txtFilter;
};
```

Here we are assigning the `$scope.txtFilter` to `$rootScope.filter`.

$rootScope :

We know that the `$scope` is used between controller and view to share the objects (model, function). `$rootScope` is very similar to a variable with global scope, means anything you put in $rootScope can be accessed from any controller or service.

Here we are using `$rootScope` to share the value of the filter text field with `taskListController`.

Note : You need to be very careful while using `$rootScope` since it may impact the performace of the application.

Next, in the `taskListController` we need something like a listener to get the values of the `$rootScope.filter` only when it has changed.

$watch :

`$watch` is a function in angular to listen to any change in either `$scope` or `$rootScope`. If you attach `$watch` to any model then angular will have periodically look for any change in the model.You need to be careful while using $watch since using too many of them will impact the performace very badly. There is something called `Digest Cycle` in Angular which I am not going to talk about in this series, however look for another artical on this topic in future.

```
module.controller('taskListController',
['$scope','$state','$stateParams','dataService','$rootScope',

function($scope,$state,$stateParams,dataService,$rootScope)
{

        $scope.filterText={
                name:''
        };

        $rootScope.$watch('filter',function(data){
                if((data!=undefined || data!=null)){
                        $scope.filterText.name=data;
                }
```

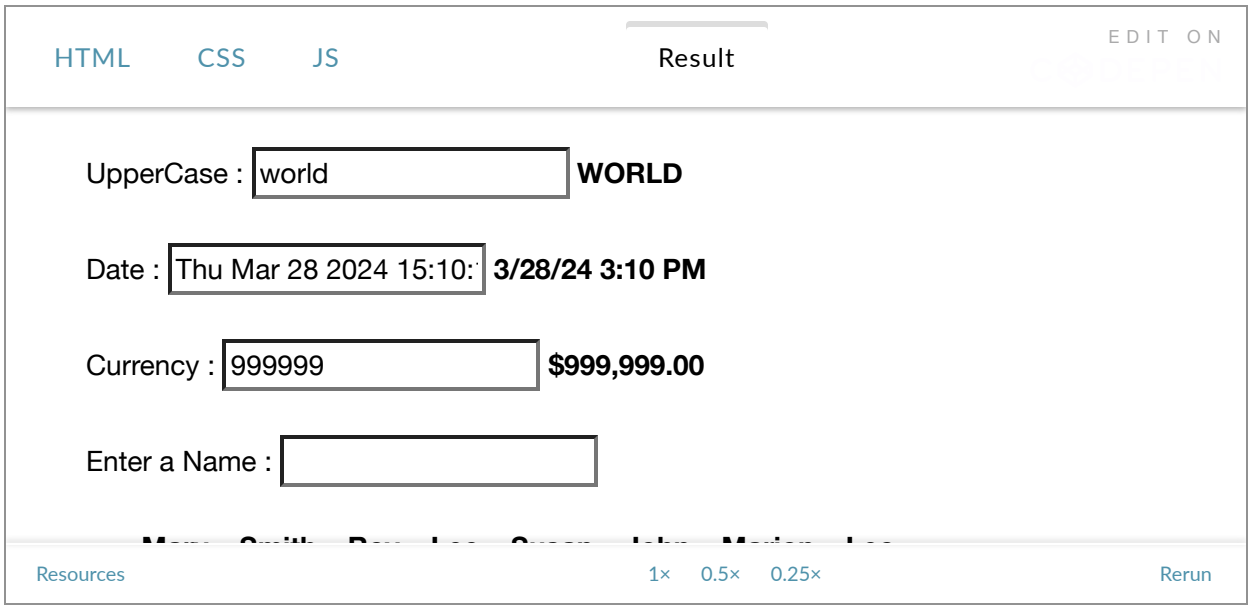
```
        });

...

}])
```

The first argument of the `$watch()` is the name of the model, `filter` in this case. The 2nd argument is a function, which will be called whenever there is a change in the model.
We are setting the updated value of the filter text to `$scope.filterText.name` model in the `taskListController`.

Angular Filter:

Angular provides many native functions to filter the data and reformat them. We can also create our own custom filters. We will be using one of the predefined Angular filter in our Task Tracker application to filter the list of tasks. You can access the filter in the View or from the Controller and Service. In order to access a filter from View (HTML) you need to use the a pipe (|), e.g. – `{{name | uppercase}}`.

Here is a demo of some of the available filters.



We can also create custom filters, however that will be a seperate discussion. In real application you don't have to write too many filters though.

Check our the last example, where you can filter the names so easily. We will implement the similar solution in our application.

```
...

...
```

Save and test the app, it should now work fine.

I am using the `strict` option here to filter only the name field in the task object. Since the task object has many fields, if we simply use a filter like in the demo above, it will apply filter on each field. However the `filterText` only has one field named `name` which is the same as the `name` present in a task object.In order to match `filtertext.name` to `task.name` we are setting the filter text to `filterText.name` and not directly to the `filterText`.

You can find the full source code for Part 6 here.

So we have successfully completed learning the basiscs of Angular !! I hope this **The Best Way to Master Angular JS** series will help you to have a strong foundation in Angular JS.

As a next step, you can complete the TaskTracker application. There are many action icons for each tasks, you can enable them. Also to help you, I have already implemented the delete task functionality. You can find the code here. I also have few more minor changes.

P.S. I will publish more articles on Routing, Directive, Filter 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 Learn and Master Angular easily – Part5
In "Angular 1.x"

Filed Under: Angular 1.x, JavaScript Tagged With: $rootScope, $watch, Angular, Angular JS, Code, example, filter, JavaScript, Learn, master, Programming, step by step, tutorial

Subscribe to stay in loop

* indicates required

Email Address *

Subscribe

Comments

gaurav soni says
June 10, 2016 at 10:47 am

(Edit)

This is best tutorial, thanks for this

Reply

A Developer Diary says
June 10, 2016 at 4:55 pm

(Edit)

Hi Gaurav,
Appreciate your feedback ! Glad that the tutorial helped you.

Thanks,
A Developer Diary

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.