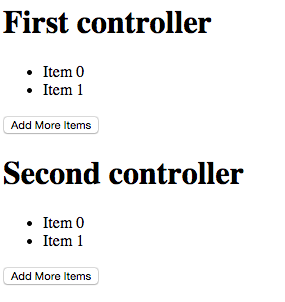
Worksheet

Services

In this worksheet you will be learning AngularJS services. However, before we go into services, lets first have a look at controllers.

# Controllers

One advantage (features) of controller is that we can initialize several different controllers using a single controller definition. For example, in the screen cast below I used two ‘instances’ of a single controller:



The html code should be easy to reverse-engineer. And below I show you the js code for this controller:



And the html looks like this:



Controllers make it convenient to do it this way. However, suppose we need to pass info between two different controllers, what should we do then?

# Value service

The simplest solution to pass data between controllers is to use value services. Make a copy of the file you created during the first section, and make the following changes.

First, define a module property called ‘.value’, as follows

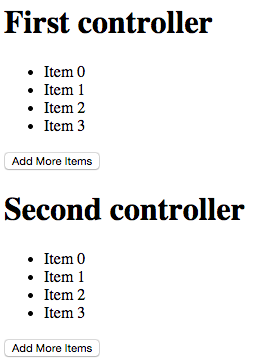


Basically this ‘myList’ value contains an array. Next make the following changes to the controller:



So what we did here is that, in a way similar to when we introduce $http service’, we define a dependency on ‘myList’ value, and assign this value to our controller property (i.e. list).

If you run this file, what you’ll see is the following:



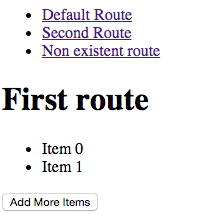
In other words, you manipulate the data provided by the service, and it became available to all controllers.

# Value service and routing

To make it more interesting, we then convert this into a routing app. First, we replace the actual contents with ‘ng-view’ directive. Next, we config $routeProvider as follows:



We you’ll see if you run the file is something similar to below:



At this point the whole discussion makes more sense – we typically want to pass info between controllers on different pages, instead of within a single page.

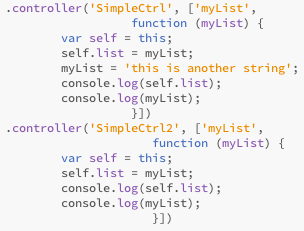
# Service and passing-by-reference

What we have seen so far is that we set-up an array in our ‘value’ services. But what will happen if we simply use a primitive type for example string or number. In such cases, values defined by ‘value’ services should be regarded as ‘constant’. In other words, we can only read, but cann’t write. Let’s see an example.

First, we define $routeProvider as follows:



Next, we define two controllers with exactly same contents:



If we run the code, we’ll see that the string defined by the value service didn’t change at all.

In the above example, we used value services. But the same principle applies to all AngluarJS services – Firstly, primitive types don’t get changed outside of service; Second, controllers will only run once, if your code needs to have an updated value returned by services, you need to assign it in a call-back function. In the section that follows, I’ll show you proper ways to do it.

# Service example

Following from out previous codes, first, make your $routeProvider and controller look like below:

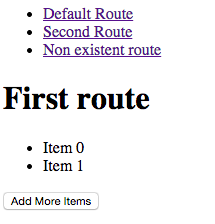


Next, we define our service using ‘factoroy’ keyword:



Basically what happened here is that we define a function called ‘ItemService’, and this function can be injected into other controllers, in a way similar to $http service.

If you follow along, in your browser you’ll see something similar to the one blow:



You can add more items on your first route, and the changes will be saved even if you move away from this current page.

Congratulations! Your now complete the lab sheets!