DOWNLOAD PDF ANGULARJS UI-ROUTER TUTORIAL

Chapter 1 : AngularJS ui router tutorial

The UI-Router is a routing framework for AngularJS built by the AngularUI team. It provides a different approach than ngRoute in that it changes your application views based on state of the application and not just the route URL.

In this article we will go through the next useful feature of AngularJS called Routing. Also we will see how we can divide a single page application in multiple views. As we add more and more logic to an app, it grows and soon become difficult to manage. Dividing it in Views and using Routing to load different part of app helps in logically dividing the app and making it more manageable. Routing helps you in dividing your application in logical views and bind different views to Controllers. Each points to a specific view and is managed by a controller. Soon we will see some code and it all will be clear. An Angular service is a singleton object created by a service factory. These service factories are functions which, in turn, are created by a service provider. The service providers are constructor functions. Sometime the service provider needs certain info in order to instantiate service object. This service makes it easy to wire together controllers, view templates, and the current URL location in the browser. Syntax to add Routing Below is the syntax to add routing and views information to an angular application. When we open http: It then invokes AddOrderController where we can add logic for our view. It includes AngularJS library and define structure for our app. We have two links: Add New Order and Show Order. Each link loads template in below section. In our angular app, we need to define ng-app directive once. This becomes the placeholder for views. Each view referred by the route is loaded in this section of document. You can define ng-view in main html file in one of the below way. Below is the content of app. Notice how we used otherwise method to define a default route. In case routeProvider does not matche with any url, it redirects to default route. We still needs to define two html templates. These are partial templates of our app. For sake of simplicity we just show a message.

Chapter 2 : How to Write Modular Code with Angular UI-Router & Named Views â€” SitePoint

So what is angular-ui router, well it is a routing framework specifically designed for AngularJS. Instead of dealing with views directly, angular-ui router organizes them in to states. It allows us to create more powerful nested views.

Whereas ngRoute functions based on routes URL, ui-router is based on states in application. With ui-router, your user interface can be organized into states which can have all sorts of logic applied on them as with ngRoute. This post and demo is inspired by Official UI-router documentation. Live Demo Lets see a quick and simple example showing AngularJS routing using ui-router, before diving into details. Above trivial application contains two links: Business page contains Nested views, while Portfolio page contains multiple named views. We will learn each step in details creating required files along the way. Basic Setup Before we continue, we need to make sure to perform some initial setup. It can be included in head or body section. It is required before we can use anything from this routing module in our code. Notice that it is different than ngRoute where we include ngRoute module Below is the index. Notice that hyperlink for them uses ui-sref instead of regular href. Clicking the link will trigger a state transition with optional parameters. In above excerpt, we have defined which states will be used [business or portfolio] when those links will be clicked. Actual states itself will be defined in config section [which we will cover in app. Additionally, notice the ui-view directive in above HTML. The ui-view directive tells angularJS where to inject the templates your states refer to. Click on below links to see Nested states in action. This represents Nesting view. Moreover, these links uses ui-sref directive, but notice that the states are named with a dot prefix [. This represents Nested States. We use dot donation when linking nested views. When clicking on these links, templates of these nested views will be injected in ui-view of their parent page which in this case is Business page. Below is our app. The state function takes two arguments: First argument is the name of state that specifies when this particular state is applicable. We can say that products and services are child states and there parent state is business [look at dot notation]. When a state is activated [for example by clicking on products or services link in business page], its templates [products. Products parent is Business [look at the dot notation in business. Note that you can assign a controller to your template, as we did in this example. Below shown are products. In the next section we will cover Multiple named views. Below shown is business. This is a setup for multiple, named views. These ui-view will be filled by templates defined in state. Notice that in our app. Key is view-name [which can be Relative or Absolute] and value is object which in turn contains actual templates to be used, and any other property [like controllers,resolve,etc. That means the specified template can be inserted into mentioned view [view1 of portfolio state e. Again, a controller can be applied on the template as we did in above example. That is all about multiple-named views. A more detailed description of Multiple-Named view can be found on ui-router wiki.

Chapter 3 : AngularJS Tutorial

Building Hello World. Follow these steps to make your own copy of the Hello World app. Get UI-Router. Get the UI-Router for AngularJS code using your preferred mechanism.

This is especially relevant when working as part of a team on highly complex applications. The Angular framework was built to create high-level applications, which can become very complex very fast, which in turn makes writing modular code all the more vital. One of the tools that can aid you greatly in this quest is the Angular UI Router , which was built to help manage different states of your web app. In contrast to the native AngularJS routing implementation, it gives you complete control over each of its views. If you have used ui-router before, you may be aware of how the dot-notation is used to define child states inside of a parent state. You may not, however, be aware of how the ui-router library deals with named views nested inside of a parent state. And this is what I am going to explain today. I am going to show you how ui-router uses absolute names to give you complete control over where specific pieces of a web app are displayed. This allows you to easily add and remove different pieces of the interface in order to make a modular application, built up of different components. Specifically, I am going to show you how to map a navigation bar, some body content, and a footer, to a specific state. As ever, the code for this tutorial can be found on GitHub and you can also find a demo at the end of the article. You can see we have an index. In this file we also have two ui-view directives into which we will insert our content. Next we have an app. This directory will be used to house all of our different views. Although this folder is not necessary, it is definitely in your best interest to use some sort of structure when building applications with Angular. As you can see, I have included an assets folder inside of the templates folder. This is where I like to keep my reusable components: I figured you may find this convention helpful as it keeps your code extremely modular. Inside of our app. A state describes what the UI looks like and what it does at that place. It works in the same way that ngRoute uses routeProvider. Below is a how the app. In this instance, we are defining a state named home, and only the URL is configured. Inside this object is where we are going to define the names of our views, as well as the paths of their templates. Here you can also define things such as controllers; however, I have passed over that for the sake of brevity in this tutorial. Moving on, we must first create and define an unnamed view that will target the parent state â€” which in this case is home. The templateUrl of this unnamed view will essentially tie the two together. Your code should now replicate the app. In order for these components to appear under the home state, we must define them using absolute naming. Specifically, we must use the syntax to tell AngularJS that these components of our application should be mapped to a specific state. This follows the viewName stateName syntax and tells our application to utilize named views from an absolute, or specific state. You can read more about relative vs. You will see home used throughout our config object, to ensure that Angular knows our named views target our home state. If these absolute names are not present, the application will not know where to find these named views. That said, take a look below and see how the application should be routed. In this tutorial, I placed all of our views inside of a templates folder. However, you can take this a step further and create folders for the different views of your applications. This allows you to reuse templates throughout your application, as well as in future projects! The ui-router library makes it extremely easy to use different components of a web application, such as header and footers for specific views. This will make it easier to reuse code throughout different projects, and can definitely save you time. Conclusion There is much more complex, high-level nesting you can do with absolute names â€” this was only one example! Nonetheless, I hope you gained a deeper perspective of some of the things that ui-router makes possible. As always, let me know if you have any questions regarding this tutorial. I would be happy to answer them. Before web development, Thomas worked as a graphic designer, and he continues to utilize his background in design when building web applications. Have a question for Thomas? You can reach him on Twitter.

Chapter 4 : Angular Routing

angularjs routing tutorial for beginners angularjs routing tutorial video angular ui router tutorial angularjs ui routing tutorial In this video I will provide an introdutcion to ui-router.

Chapter 5 : AngularJS UI Router Full Tutorial - racedaydvl.com

UI-router is a routing framework for AngularJS. It's a flexible alternative to ngRoute as it supports Nested & Multiple Named views. Whereas ngRoute functions based on routes URL, ui-router is based on states in application.

Chapter 6 : Routing in AngularJS with ui-router - Thinkster

-What is squashing in UI Router -How to use squashing with default and optional parameters -How to use query parameters using UI Router (Query String Parameters).

Chapter 7 : javascript - AngularJS ui-router tutorial undefined binding es - Stack Overflow

To create a UI-Router bundle to test a bug fix against your app, run npm run package You can then run npm link, and then run npm link @uirouter/angularjs in your app's directory. Your app's npm dependency will use the local @uirouter/angularjs package that you just built.

Chapter 8 : Angularjs Routing Tutorial using ui-router - WebSystique

What is Routing in AngularJS? If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

Chapter 9 : 5 AngularJS UI-Router Examples | AngularJS 4U

This tutorial will teach you how to build a single page javascript application using racedaydvl.com from Google. Angular is an amazing framework for rapid development and building stable apps with.