egghead.io presents

UI Router Cheatsheet

CONFIGURATION

$stateProvider

Used in the config block, the `$stateProvider` is used to configure states for an application or module.

state

`$stateProvider.state(stateName, stateConfig)`

stateName

This is a unique name for the state. Parent/child relationships within a state are defined by separating the name with a `.` (dot).

stateConfig

The `stateConfig` is an **object** that holds configuration properties for the state.

- `[template|templateUrl|templateProvider]` - An html string, a URL, or a function that returns HTML for the state's template.
- `[controller|controllerProvider]` - The controller **function** OR **string** name. Alternatively the controllerProvider is an injectable function that returns the controller function or string name.
- `resolve` - an **object** whose keys are the name of a dependency to inject into the state's controller. The values are factories that can be a string alias for an existing service **or** a promise. If it's a promise, it must resolve before the state will load.
- `url` - (parameterized) string url for the state.
- `params` - **array** of parameter names or regular expressions. **only used when no url is present.**
- `views` - `object` to configure multiple views for a state. The keys are the name of the view to target and the `values` are `objects` to set the controller and template for the given view .
- `abstract` - **boolean** when set to true won't allow the state to be directly activated, but can be used for inheritance to child states.
- `onEnter` and `onExit` - functions that will be executed when a state is entered or exited. Used to trigger an action (open a dialog?)
- `reloadOnSearch` - **boolean** if false will not retrigger the same state when a query (search) parameter has changed.
- `data` - **object** that contains arbitrary keys/values for configuration purposes.

$urlRouterProvider

Used in the `config` block, the `$urlRouterProvider` allows configuration of rules for the URL routing feature of ui-router.

`$urlRouterProvider.otherwise(path)` - If a URL doesn't resolve, this `path` will be used.

`$urlRouterProvider.when(whenPath, toPath)` - Takes a String or REGEX as the `whenPath` argument, and when the url matches, it will redirect to the `toPath`

`$urlRouterProvider.rule(handler)` - Custom URL handling. handler is any function that takes `$location` as its only argument and returns a valid path as a string.

```
Example Code!

.config(function ($urlRouterProvider) {
  $urlRouterProvider.otherwise('/');
})
```

Notes

DIRECTIVES

ui-view

Tells `$state` where to *transclude* (or place) your templates. Generally used as an `attribute`, but can be used as an `element` as well. Its optional value is a name.

Only **one** unnamed view can exist in a template.

```
Example Code!

<div ui-view></div>
<div ui-view=”viewName”></div>
```

Notes

Autoscroll

Allows you to set the scroll behavior when a `ui-view` is populated.

```
Example Code!

<!-- If autoscroll unspecified, then scroll ui-view into view
    (Note: this default behavior is under review and may be reversed) -->
<ui-view/>

<!-- If autoscroll present with no expression,
     then scroll ui-view into view -->
<ui-view autoscroll/>

<!-- If autoscroll present with valid expression,
     then scroll ui-view into view if expression evaluates to true -->
<ui-view autoscroll=’true’/>
<ui-view autoscroll=’false’/>
<ui-view autoscroll=’scopeVariable’/>
```

ui-sref

Creates a clickable link to a state.

`<a ui-sref=’stateName’>Click</a>` - creates alink to `stateName` with no parameters.

`<a ui-sref=’stateName({param: value})’>Click</a>` - create a link to a state with a parameter named param that contains value.

ui-sref-active

Directive used to denote active elements. Used with ui-sref, typically for navigation elements.

`ui-sref-active=’class1 class2 class3’` - will apply all of the given css class when the contained `ui-sref` is active.

When in state `app.user`, and the `user` parameter’s value equals ‘bilbobaggins’, the resulting HTML will be:

`<li ui-sref-active=”active” class=”item active”>`

```
Example Code!

<ul>
  <li ui-sref-active=”active”
class=”item”>
    <a href ui-sref=”app.user({user:
‘bilbobaggins’})”>@bilbobaggins</a>
  </li>
  <!-- ... -->
</ul>
```

SERVICES

$state

$state.go

`$state.go(to [, toParams] [, options])` - returns a **promise** representing the state of the transition.

to

string absolute state name **or** relative state path.

The name of the state that will be transitioned to **or** a relative state path. If the path starts with `^` or `.` then it is relative, otherwise it is absolute.

```
$state.go('contact.detail')
// will go to the 'contact.detail' state
$state.go('^')
// will go to a parent state.
$state.go('^.sibling')
// will go to a sibling state.
$state.go('.child.grandchild')
// will go to a grandchild state.
```

toParams

object map of parameters that will be sent to the target state as `$stateParams`

Any parameters not defined will be inherited from the current state's parameters.

options

object that contains options for the target state.

- `location` **boolean** or "replace" (default true), If true will update the url in the location bar, if false will not. If string "replace", will update url and also replace last history record.
- `inherit` **boolean** (default true), If true will inherit url parameters from current url.
- `relative` **stateObject** (default $state.$current), When transitioning with relative path (e.g `^`), defines which state to be relative from.
- `notify` **boolean** (default true), If true will broadcast $stateChangeStart and $stateChangeSuccess events.
- `reload` **boolean** (default false), If true will force transition even if the state or params have not changed, aka a reload of the same state. It differs from reloadOnSearch because you'd use this when you want to force a reload when everything is the same, including search params.

`$state.reload()` - returns **null** forces a reload of the current state

`$state.includes(stateName [, params])` - returns **boolean** to determine if the active state is equal to, or is a child of, the `stateName` with the optionally supplied parameters.

`$state.is(stateOrName [, params])` - similar to includes, but only checks if the state equals the state indicated by the given arguments.

`$state.href(stateOrName [, params] [, options])` - returns a **string** compiled URL for the given state and optional parmeters.

`$state.get([stateName])` - returns the **stateObject** when given the `stateName`, or an **array** of all states when given no arguments

---

`$state.current` - returns the current **stateObject**

$stateParams

A service that is populated by the current state's parameters. Useful for injecting into your own controllers or services to access the parameters. It will have one key per url parameter.

Notes

FILTERS

isState

`“stateName” | isState` - Translates to `$state.is(“stateName”)`

---

includedByState

`“stateName” | includedByState` - Translates to `$state.includes(“stateName”)`

EVENTS

state change events

All these events are broadcast from the `$rootScope`.

`$stateChangeSuccess` - fired once the state transition is complete.
`$stateChangeStart` - fired when the transition begins.
`$stateNotFound` - fired when a state cannot be found by its name.
`$stateChangeError` - fired when an error occurs during transition.

view load events

`$viewContentLoading` - fired once per view when the view begins loading (before DOM is rendered). Broadcast from $rootScope.

`$viewContentLoaded` - fired once per view when the view is loaded (after DOM is rendered). Emitted from view’s $scope.

Notes