Meet The AngularJS Injector

Monday, January 13, 2014

Most of us never see the $injector service in Angular until there is a problem with dependencies, at which point we'll see an error message like the following.

Uncaught Error: [$injector:unpr] Unknown provider: someServiceProvider <- someService

It is the Angular $injector that knows how to invoke functions by analyzing their dependencies and passing the correct parameters by fetching the proper service instances. Angular creates a single $injector when it bootstraps an application and uses the single $injector to invoke controller functions, service functions, filter functions, and any other function that might need dependencies as parameters.

The above error would occur if a function asks for a service named "someService", but "someService" is not a service the injector knows about (because, perhaps, the service wasn't registered correctly).

You can use the $injector service, like any service, just by asking for the $injector by name.

```
var someFunction = function($rootScope, $http) {
    return "called!";
};

app.run(function($rootScope, $injector) {

    $rootScope.annotations = $injector.annotate(someFunction);
    $rootScope.message = $injector.invoke(someFunction);

});
```

The above code demonstrates two capabilities of the $injector.

1. You can use the annotate API to discover the dependency annotations for an injectable function (the above code would list "$rootScope" and "$http" as annotations).

2. You can use the invoke API to execute an injectable function and have the $injector pass in the proper services.

Manual Injection

In some cases you might find it useful to create your own $injector instead of using the injector created by Angular during application startup. As an example, creating your own injector is useful in unit tests where you do not want singleton service instances. You can create your own injector using the angular.injector method.

```
var injector = angular.injector(["ng"]);

var someFunction = function($http) {
        // ...
};

injector.invoke(someFunction);
```

You must pass a list of the modules the injector will work with (just the core “ng” module in the above code). You have to explicitly list the ng module if you are going to use any services from the core of Angular. Unlike the angular.module method, which assumes you have a dependency on the ng module and will silently add “ng” to your list of dependencies, the injector function makes no assumptions about dependent modules.

Comments

Comments are closed.