Introduction to Angular

Dylan Lane McDonald

CNM STEMulus Center
Web Development with PHP

November 12, 2015

Outline

Angular Architecture

Angular is a "superheroic" Model-View-Whatever (MVW) framework that separates the concerns of the user interface, how the front end is updated, and how the front end is connected to the backend. Angular is highly modular and consists of reusable components that can be integrated together rapdily to accomplish any use case.

- **Modularity:** Angular consists of several types of modular components that accomplish one task well.
- **Two Way Binding:** When a variable is updated by one component, it is automatically updated in both the backend and frontend components.
- **Promises:** Angular vastly simplifies grabbing data from AJAX and other dynamic components by providing a common API.
- **Templating:** Scope variables can be integrated into a template, providing a common view for all objects.

Angular Architecture

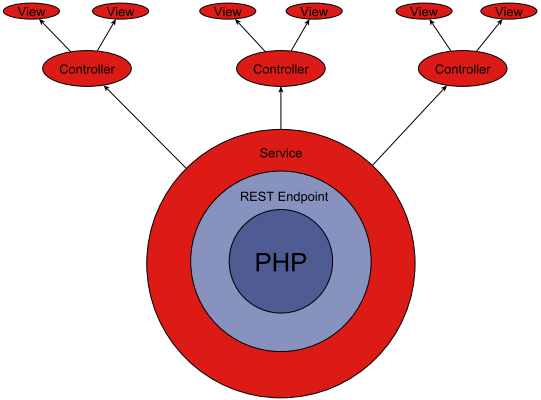

Figure 1: Angular Architecture

Representational State Transfer (REST)

Definition

REST, *a mnemonic for* **Representational State Transfer**, *is a software architectural pattern that lays out how to interchange data amongst different sites.*

REST allows sites with different backends, different languages, different use cases, etc. to easily exchange information without the complication of writing site-to-site specific integration code. REST rides on top of standard HTTP verbs to perform a standard set of operations. Using the ubiquity of HTTP, REST is an easy-to-program, agreed upon standard for exchanging data among sites.

Promises

Definition

A **promise** *is a service that helps one run a method asynchronously and use the return values (or exceptions) when they are available.*

A promise can be thought of as a "latent if" that allows for asynchronous operations to process as they happen. [1] For instance, suppose a HTTP request is made to an external API. The promise will **wait** and inform Angular of three possible outcomes:

1. **Successful Contact & Data:** Data was returned successfully
2. **Successful Contact, Failure on Data:** The server had no data
3. **Completely Unsuccessful:** The server was offline

In case 1, the promise is **accepted**. Cases 2 & 3 are **rejected**.

Services

Definition

A **service** *is a layer that rides on an asynchronous operation (e.g., a REST API endpoint) that uses a promise to handle the latency and results.*

A service acts as a bridge from an asynchronous operation to a promise object, where it can be processed and interpreted by controllers. Services are normally a REST API endpoint, but can be any asynchronous event, such as a click event, an exception, a redirect, or a timer. All services will have the following properties:

- **Input:** Parameter(s) needed to fire up the service
- **Output:** Promise containing the results when they are ready

Services standardize asynchronous events such as AJAX and DOM events and provide a common API to handle them.

Controllers

Controllers are the focal point for Angular development. The definition is identical as from the MVC design pattern. Each controller is dependency injected with the services and other dependencies it requires.

Definition

Dependency Injection *is a design pattern in which a dependent object is dynamically injected into an object's state variables.*

Dependency Injection is the center of Angular's flexibility: what one object needs can be determined on an individual case-by-case basis by the developer and dynamically loaded and instantiated by Angular on demand. This way, only the objects needed are loaded–no more, no less. Dependency Injection is a main feature of Angular, and has been a core feature of software frameworks such as Java Spring for many years. [2, 3]

Using Controllers

Once dependencies have been (easily) resolved via dependency injection, controllers act as the main hub for data flowing to and from the backend. A controller can consist of anything the developer can imagine. From a pragmatic standpoint, controllers normally utilizes $scope, which contains the state of the controller. $scope largely consists of two items:

1. **state variables:** state variables, many of which have been loaded after accepting or rejecting a promise from a service
2. **methods:** methods that respond to promises, user input, and other DOM events

Using these two items, controllers act as the bridge from the backend to the frontend.

Directives

Definition

A **directive** *is an Angularific marker in the DOM that uses custom HTML to inject an Angular customized behavior on the DOM.*

The main idea of a directive is to define a generalized behavior and inject it as necessary using standard HTML syntax. [4] Directives can be custom HTML attributes, tags, classes, or comments. Angular has many predefined directives:

- **ng-class:** Use a class if a condition is true.
- **ng-repeat:** Loop through an array in $scope, **foreach** style.
- **ng-show:** Show if a condition is true.

The power of directives is that one can specify directives to inject anything into the DOM, $scope, etc.

Angular Bootstrap

The **Angular Bootstrap** project provides a freely available set of directives that allow one to use Bootstrap's JavaScript components using Angular. Standard Bootstrap uses jQuery to drive its JavaScript components. Angular Bootstrap isolates the jQuery dependent components and eliminates the requirement to include jQuery or Bootstrap's JavaScript files. The components available include:

- Carousels
- Drop Down Menus
- Modals
- Typeaheads

. . . and more. [5]

Works Cited I


AngularJS Team.
Angularjs: Api: $q.
https://docs.angularjs.org/api/ng/service/$q, 2015.


Anand Mani Sankar.
Angularjs dependency injection - demystified.
http://anandmanisankar.com/posts/angularjs-dependency-injection-demystified/, 2014.


Pivotal Software.
The ioc container.
http://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html, 2015.

Works Cited II


Sandeep Panda.
A practical guide to angularjs directives.
http://www.sitepoint.com/practical-guide-angularjs-directives/, 2014.


AngularJS Team.
Ui bootstrap.
https://angular-ui.github.io/bootstrap/, 2015.