Guiding Principles
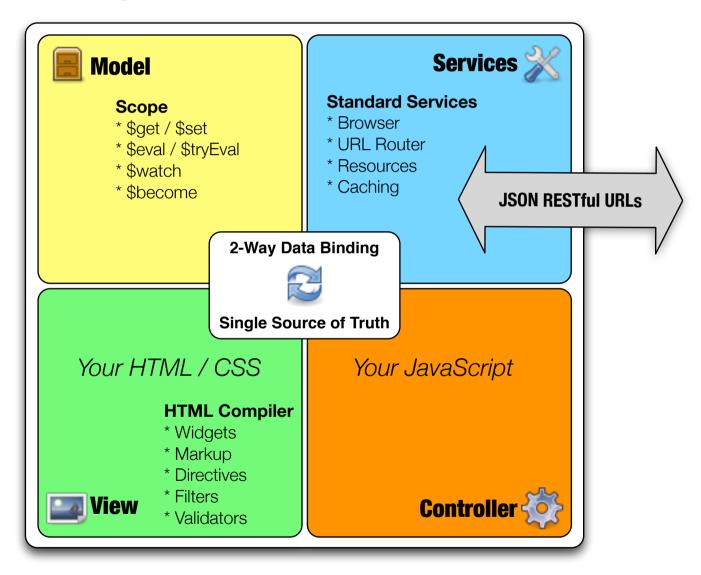* Convention over Configuration
* Declarative / Self Describing
* Testable
* DRY (Don't Repeat Yourself)
* CRUD ~ 80% -> make it trivial

<angular/> Enabled Browser

Model

Scope
* $get / $set
* $eval / $tryEval
* $watch
* $become

Services

Standard Services
* Browser
* URL Router
* Resources
* Caching

JSON RESTful URLs

2-Way Data Binding

Single Source of Truth

View

Your HTML / CSS

HTML Compiler
* Widgets
* Markup
* Directives
* Filters
* Validators

Controller

Your JavaScript

```html
<html>
  <body>
    Find by email: <input name="email" ng-validate="email"/>
    <ul>
      <li ng-repeat="person in people.$filter(email)">
        {{ person.last | uppercase }},
        {{ person.first }},
      </li>
      <ng:switch on="$location.hashPath">
        <div ng-switch-when="home">Welcome</div>
        <ng:include ng-switch-when="account" src="'account.html'"/>
      </ng:switch>
    </ul>
  </body>
</html>
```

Legend

Scope: Outter most scope which holds services and properties such as `people` and `email`.

Input Widget: binds to `email` in its scope. Changing scope changes the widget and vice versa.

Validator: an input widget may have optional validator to notify user of wrong input.

Directives: instructing the compiler to perform specific actions. In this case a repeater iterates over the list of `people` looking for the person with specific email. The DOM element is then replicated to match the number of elements.
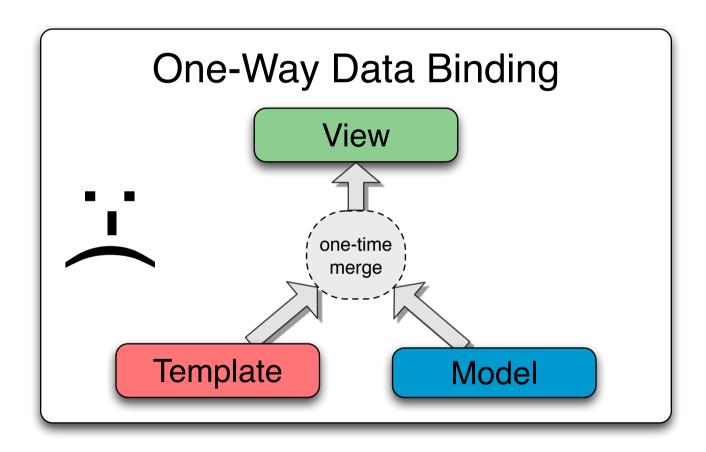
Child Scope: in this case the ng-repeat directive triggers the creation of new scopes one for each item in an iterator expression. It than assigns the 'person' to each scope. The scopes inherit from parent scopes so anything declared at higher scope is still visible.
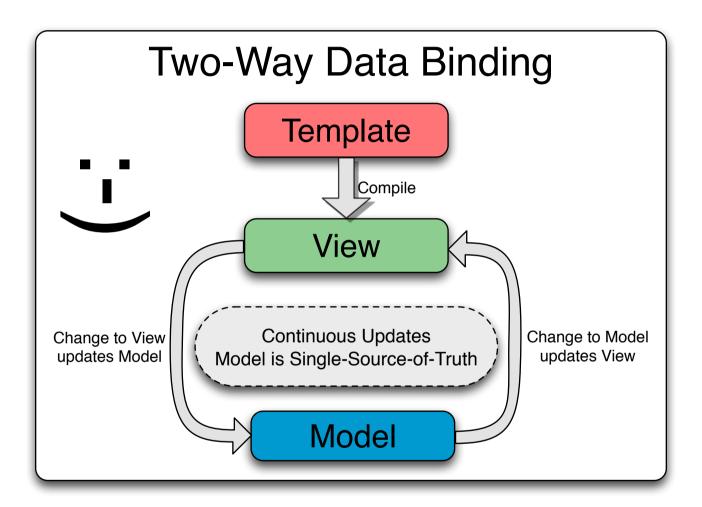
Markup: evaluates expression in the closest scope and inserts it into DOM.

Filter: Markup may include optional filter to transform value before it is displayed.

Widget: Allows the execution of custom code which can transform the DOM.

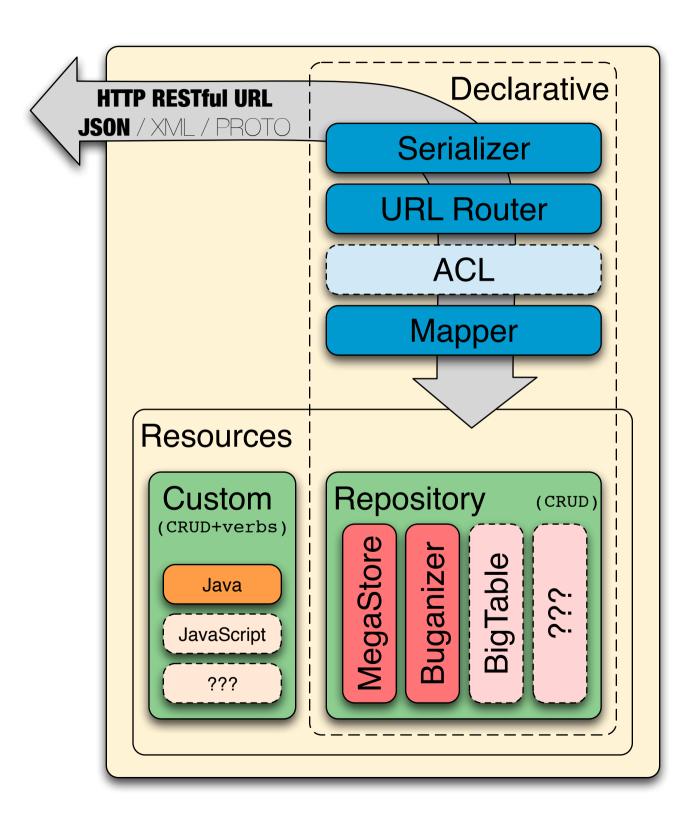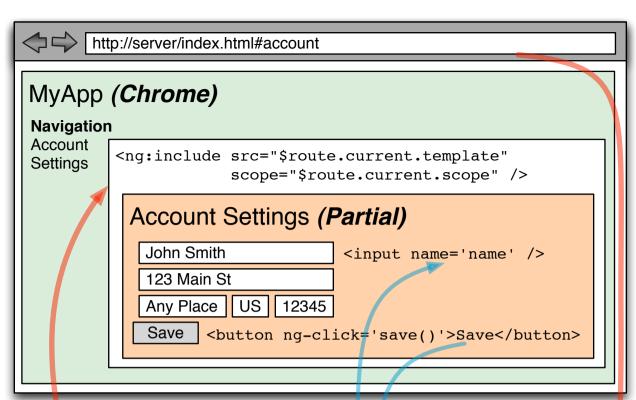**Nested Widgets:** widgets can be nested for added expressivness.

Two-Way Data Binding

Template

Compile

View

Continuous Updates
Model is Single-Source-of-Truth

Change to View
updates Model

Change to Model
updates View

Model

RESTful URLs

`http://server/data/`***Collection[/ID]***

	Verb	URL	Send	Receive
Create	POST	/Book	{ name:'Moby' }	{ id:123, name:'Moby' }
	POST	/Book	{ name:'Gatsby'}	{ id:456, name:'Rye' }
Read	GET	/Book/123		{ id:123, name:'Moby' }
	GET	/Book/456		{ id:456, name:'Gatsby' }
	GET	/Book		[{ id:123, name:'Moby' }, { id:456, name:'Rye' }]
Update	POST	/Book/456	{ id: 456, name:'Catch' }	{ id:456, name:'Catch' }
Delete	DELETE	/Book/456		

RESTy: Declarative Data Storage

HTTP RESTful URL
JSON / XML / PROTO

Declarative

Serializer

URL Router

ACL

Mapper

Resources

Custom
(CRUD+verbs)

Java

JavaScript

???

Repository
(CRUD)

MegaStore

Buganizer

BigTable

???