specialist ICT learning

GURUTEAM

H&R BLOCK

AngularJS: Model, View, Controller

GuruTeam Instructor: Sang Shin

Topics

- Templates
- Expressions
- Directives
- Controllers
- Scopes
- ControllerAs

Templates

What is a Template?

- In Angular, templates are written with HTML that contains Angular-specific elements and attributes
- Angular combines the template with
  - Data from the model
  - Controller to render the dynamic view that a user sees in the browser

Template (HTML page)Example

- A template (HTML page) with directives and expression bindings:

```
<!DOCTYPE html>
<html>
<body ng-app="myApp">

<div ng-controller="DoubleController">
Two times <input ng-model="num"> equals {{ double(num) }}
</div>
<my-own-component></my-own-component>
<my-own-directive></my-own-directive>

<script>
var myApp = angular.module('myApp', []);
myApp.controller('DoubleController', ['$scope', function ($scope) {
    $scope.double = function (value) {
          return value * 2;
     };
}]);
</script>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></script>
</body>
```

directive

expression

model

Expressions

Expressions

- Angular expressions are JavaScript-like code snippets
  - {{ expression }}

- Control flows (such as loops, if) are not allowed, however

“firstName” and “lastName” are model attrs (properties of scope object)

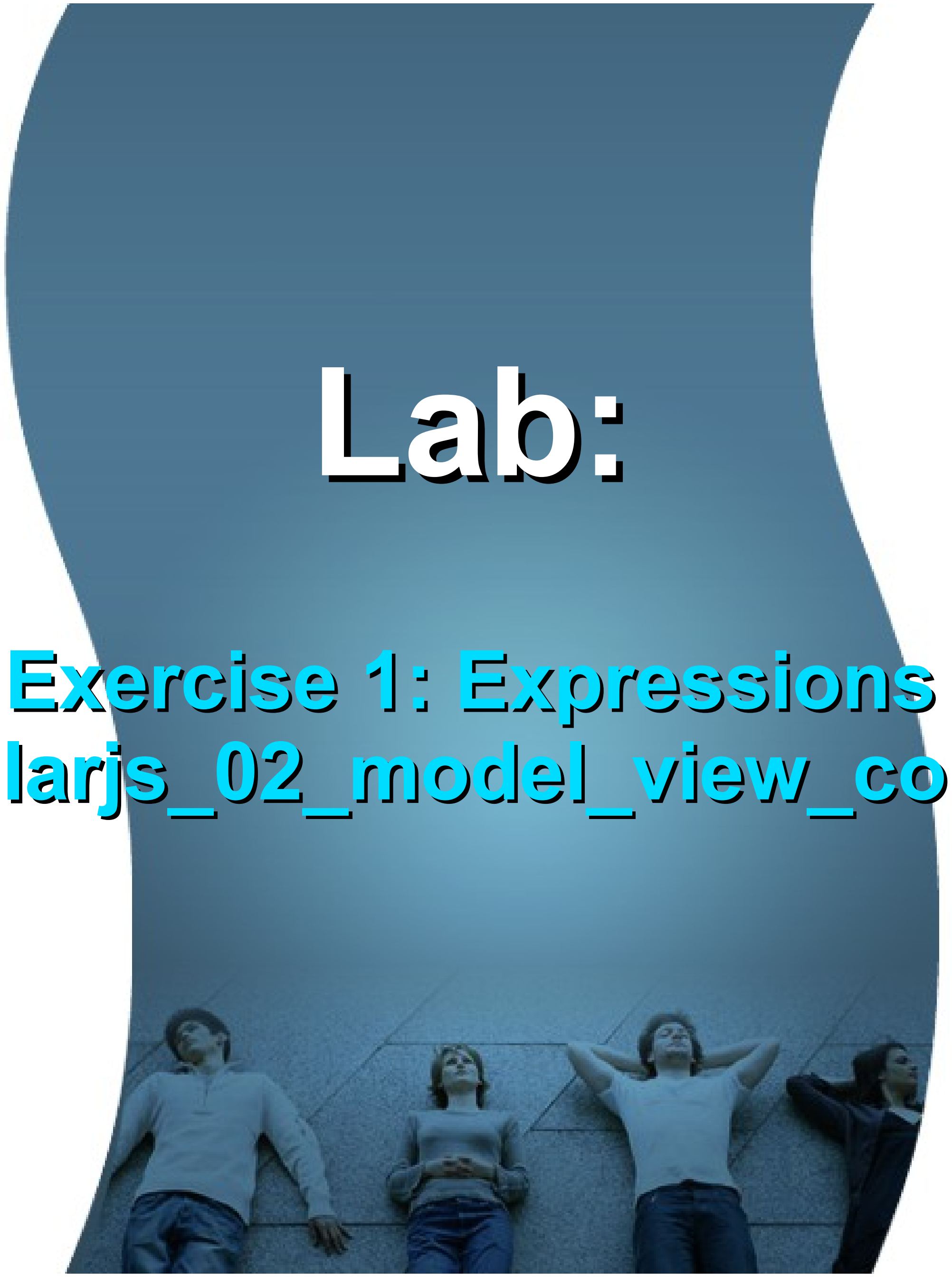
- Examples
  - {{ 2 + 5 }}
  - {{ firstName + " " + lastName }}
  - {{ user.name }}   // from “user” JavaScript object
  - {{ items[index]}}   // from “items” array

Expressions & ng-bind

- Expressions works in the same way as the *ng-bind* directive - usage of Expression is simpler to code

```
<div ng-init="name='Sang Shin';age=99">
<p>Using Expression: {{ 15 + 3 }}</p>
<p>Using ng-bind: <span ng-bind="15+3"></span>
</p>

<p>{{ name + " is " + age + " years old." }}</p>
<p ng-bind="name + ' is ' + age + ' years old.'"></p>
</div>
```



Lab:

Exercise 1: Expressions

3302_angularjs_02_model_view_controller.zip

Directives

What are Directives?

- Extends HTML attributes with the prefix *ng-* or *data-ng-*
  - Attach special behavior or transform DOM element without having to write JavaScript code

- AngularJS provided directives
  - *ng-app*
  - *ng-controller*
  - *ng-init*
  - *ng-model*
  - *ng-bind*
  - *ng-if, ng-switch*
  - *ng-show, ng-hide*
  - *ng-repeat*
  - *ng-click*

ng-app (ngApp) directive

- Use this directive to auto-bootstrap an AngularJS application
  - Without the usage of “ng-app”, a template will not be “compiled” - in other words, it will not be recognized as an AngularJS template
  - The *ng-app* directive designates the AngularJS root element and is typically placed near the HTML root element of the page - e.g. on the <body> or <html> tags
- Only one ng-app directive per HTML document is allowed

ng-model (ngModel) directive

- Binds the value of HTML input controls (<input>, <select>, <textarea>) to application data (scope properties)
  - <input type="text" ng-model="name">
- Supports HTML5 validation behavior (i.e. required, number, email, url, minlength, maxlength)
  - <input type="text" ng-model="name" required>
- Supports two-way databinding

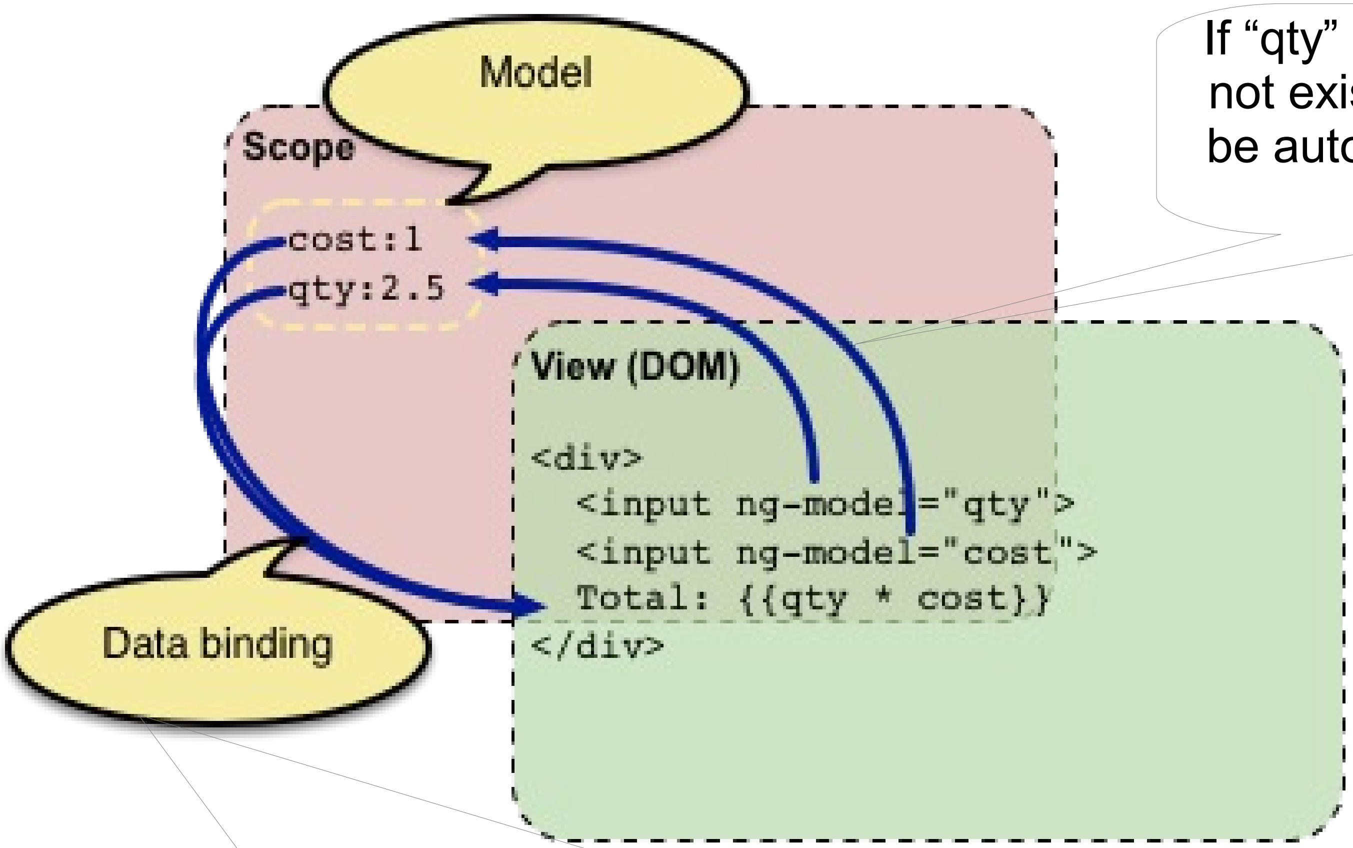
ng-model (ngModel) directive example

- ngModel will try to bind to the property given by evaluating the expression on the current scope
  - If the property doesn't already exist on this scope, it will be created implicitly and added to the scope

```
<div ng-app>
  <b>Invoice:</b>
  <div>
    Quantity: <input type="number" min="0" ng-model="qty">
  </div>
  <div>
    Costs: <input type="number" min="0" ng-model="cost">
  </div>
  <div>
    <b>Total:</b> {{qty * cost | currency}}
  </div>
</div>
```

"qty" property
is created in $rootScope

Data-binding between Model and View



```
<div>
  <input ng-model="qty">
  <input ng-model="cost">
  Total: {{qty * cost}}
</div>
```

If “qty” and “cost” properties do not exist in the scope, they will be automatically created in the scope.

Two-day data binding: (1) If user change values in the view, the changed values will set the scope properties (2) If values of the properties are changed, the changed values get reflected in the view automatically

ng-repeat (ngRepeat) directive

- The *ngRepeat* directive instantiates a template once per item from a collection
- Special properties are exposed on the local scope of each template instance, including
  - *$index, $first, $middle, $last, $even, $odd*

```
<div ng-init="students=['Tom','Sang','Yunna','Mary']">
    <p>Looping names with ng-repeat:</p>
    <ul>
        <li ng-repeat="student in students">
            {{$index}}: {{ student }}
        </li>
    </ul>
</div>
```

ng-repeat two-way data binding

- When the contents of the collection change, ngRepeat makes the corresponding changes to the DOM (view)
  - When an item is added, a new instance of the template is added to the DOM
  - When an item is removed, its template instance is removed from the DOM
  - When items are reordered, their respective templates are reordered in the DOM

```
<div ng-init="students=['Tom','Sang','Yunna','Mary']">
    <p>Looping names with ng-repeat:</p>
    <ul>
        <li ng-repeat="student in students">
            {{$index}}: {{ student }}
        </li>
    </ul>
</div>
<a href="" ng-click="students.push('Jina')">Add a student</a>
```

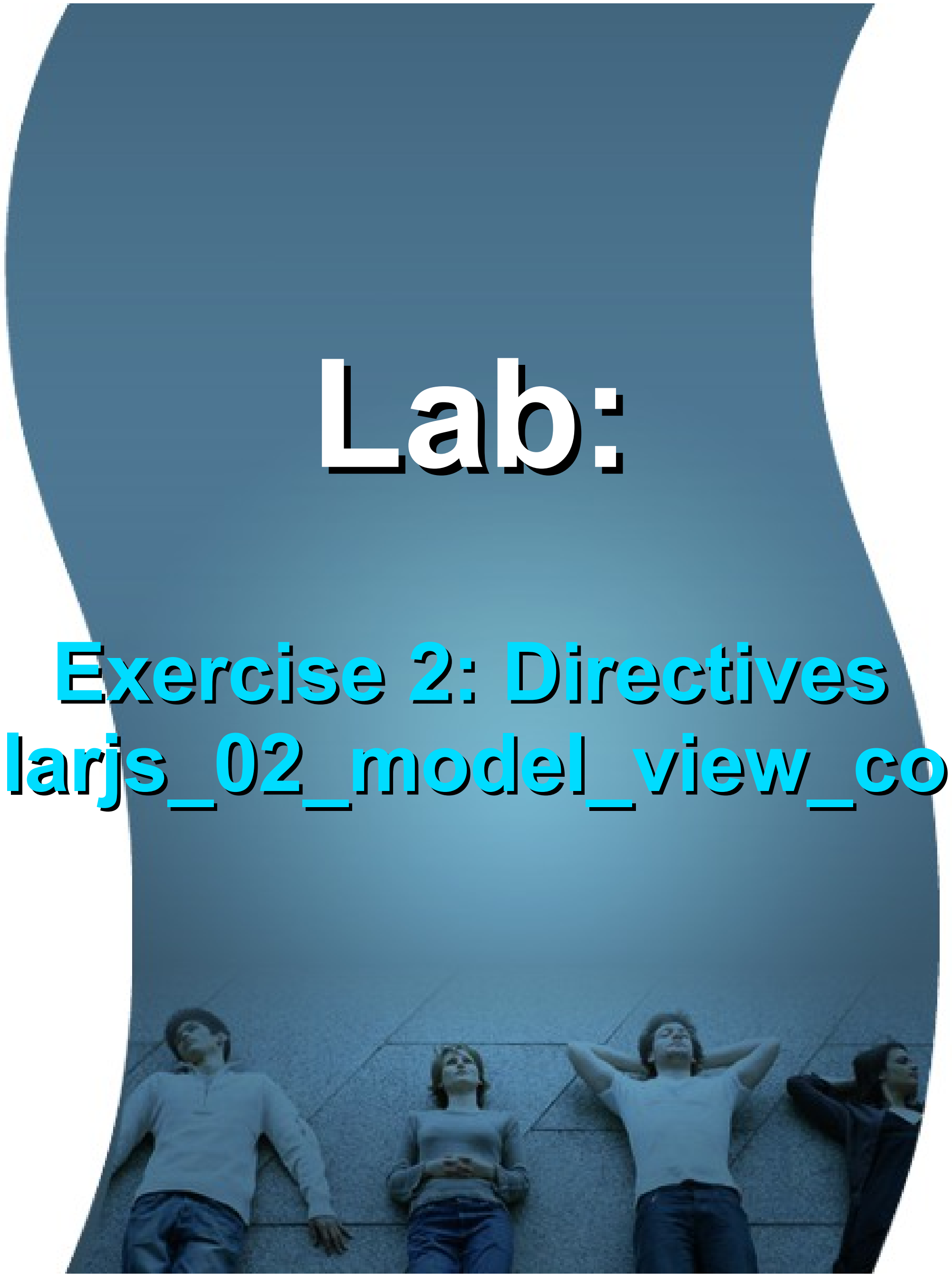
When it is clicked, the view automatically reflects the newly added student

Directive Syntax

- AngularJS HTML compiler supports multiple formats
- *ng-model, ng-bind, ng-click …*
  - Recommended format
- *data-ng-model, data-ng-bind, data-ng-click …*
  - Recommended format when HTML5 validation is important
- *ng_model, ng:model, x-ng-model*
  - Legacy, not recommended

Directive Name Normalization

- Directives are referenced by their case-sensitive camelCase normalized name
  - For example, ngModel (for ng-model), ngApp (for ng-app), ngBind (for ng-bind), etc
- However, since HTML is case-insensitive, we refer to directives in the DOM by lower-case forms, typically using dash-delimited attributes on DOM elements
  - For example, ng-model, ng-app, ng-bind, etc
- Angular normalizes an element's tag and attribute name to determine which elements match which directives
  - For example, “ng-model” or “data-ng-model” notation is normalized to “ngModel” directive



Lab:

Exercise 2: Directives
3302_angularjs_02_model_view_controller.zip

Controllers

What is a Controller?

- In Angular, a Controller is a JavaScript constructor function that is used to augment the Angular Scope
  - When a Controller is attached to the DOM via the *ng-controller* directive, Angular will instantiate a new Controller object, using the specified Controller's constructor function
  - A new child scope will be available as an injectable parameter, represented by $scope, to the Controller's constructor function

```
<div ng-controller="DoubleController">
Two times <input ng-model="num"> equals {{ double(num) }}
</div>

<script>
var myApp = angular.module('myApp', []);
myApp.controller('DoubleController', ['$scope', function ($scope) {
    $scope.double = function (value) {
        return value * 2;
    };
}]);
```

Create and use a controller

- In order to create a GreetingController, you have to have a reference to Application module - Application module object needs to be created first

```
// Create an Angular module called “myApp”
var myApp = angular.module('myApp',[]);

// Add the controller's constructor function to the module using the .controller()
// method.  This keeps the controller's constructor function out of the global scope.
myApp.controller('GreetingController', ['$scope', function($scope) {
  $scope.greeting = 'Hola!';
}]);
```

- We attach our controller to the DOM using the *ng-controller* directive - The *greeting* property below can now be data-bound to the template:

```
<div ng-controller="GreetingController">
  {{ greeting }}
</div>
```

When to use (and not to use) Controller

- Use controllers to
  - Set up the initial state of the $scope object.
  - Add behavior to the $scope object.
- Do not use controller to
  - Manipulate DOM — Controllers should contain only business logic. Putting any presentation logic into Controllers significantly affects its testability. Angular has databinding for most cases and directives to encapsulate manual DOM manipulation.
  - Format input — Use angular form controls
  - Filter output — Use angular filters
  - Share code or state across controllers — Use angular services
  - Manage the life-cycle of other components (for example, to create service instances)

Lab:

Exercise 3: Controllers
3302_angularjs_02_model_view_controller.zip

Scopes

Scope Object

- Typically, when you create a controller, you need to set up the initial state for the Angular $scope
  - You set up the initial state of a scope by attaching properties to the $scope object

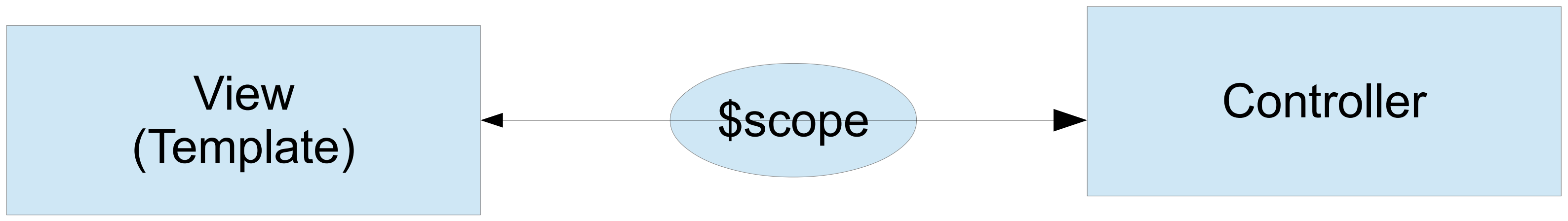
```
myApp.controller('GreetingController', ['$scope', function($scope) {
  $scope.greeting = 'Hola!';
  $scope.sayGreeting = function (){
    // some code
   }
}]);
```

- All the $scope properties will be available to the template at the point in the DOM where the Controller is registered.

```
<div ng-controller="GreetingController">
    <div>{{greeting}}</div>
    <button ng-click=”sayGreeting()”>Click me</button>
</div>
```

View, Controller, and Scope

- $scope is an object that can be used to allow communication between View and Controller - separation of roles.



```
// Controller
myApp.controller('GreetingController', ['$scope', function($scope) {
  $scope.greeting = 'Hello!';
  $scope.sayGreeting = function (){
    // some code
   }
}]);

// View (template)
<div ng-controller="GreetingController">
    <div>{{greeting}}</div>
    <button ng-click=”sayGreeting()”>Click me</button>
</div>
```

Separation of roles:
View does not know anything about the Controller and Controller does not anything about the view.

Scope as Data Model

- Scopes provide context against which expressions are evaluated
  - For example {{username}} expression is meaningless, unless it is evaluated against a specific scope which defines the username property
- Both controllers and views have reference to the scope, but not to each other
  - This makes the controllers view agnostic, which greatly improves the testability of the application

Scope Hierarchies

- Each Angular application has exactly one root scope ($rootScope), but may have several child scopes
  - Each controller has its own scope, which is a child scope of the $rootScope
- The application can have multiple scopes
  - When new scopes are created, they are added as children of their parent scope
  - This creates a tree structure which parallels the DOM where they're attached.

Lab:

Exercise 4: Scopes
3302_angularjs_02_model_view_controller.zip

ControllerAs

ControllerAs

- Introduced from Angular 1.2+
- Reduces the developer confusion regarding the usage of $scope
- Recommended to be used moving forward instead of $scope – Angular 2 removed $scope

```
myApp.controller('GreetingController', function() {
  this.greeting = 'Hola!';
  this.sayGreeting = function (){
    // some code
   }
 });

 <div ng-controller="GreetingController as ctrl">
    <div>{{ctrl.greeting}}</div>
    <button ng-click=”ctrl.sayGreeting()”>Click me</button>
 </div>
```

Lab:

Exercise 5: ControllerAs
3302_angularjs_02_model_view_controller.zip

Who are GuruTeam?

- Specialist onsite training in Linux, Cloud, Database, Architecture, Software and Web Development Technologies
- Accredited by the LPI, CompTIA, Hortonworks and the Cloud Credential Council to deliver training, examinations and certifications.
- Over 230 courses available
- All GuruTeam instructors have extensive real-world experience in their technologies
- Clients are indigenous Irish Companies and Multinationals
- We can bring high spec preconfigured equipment for deliveries in Ireland, the UK and Europe.

Exceed your expectations...
GURUTEAM