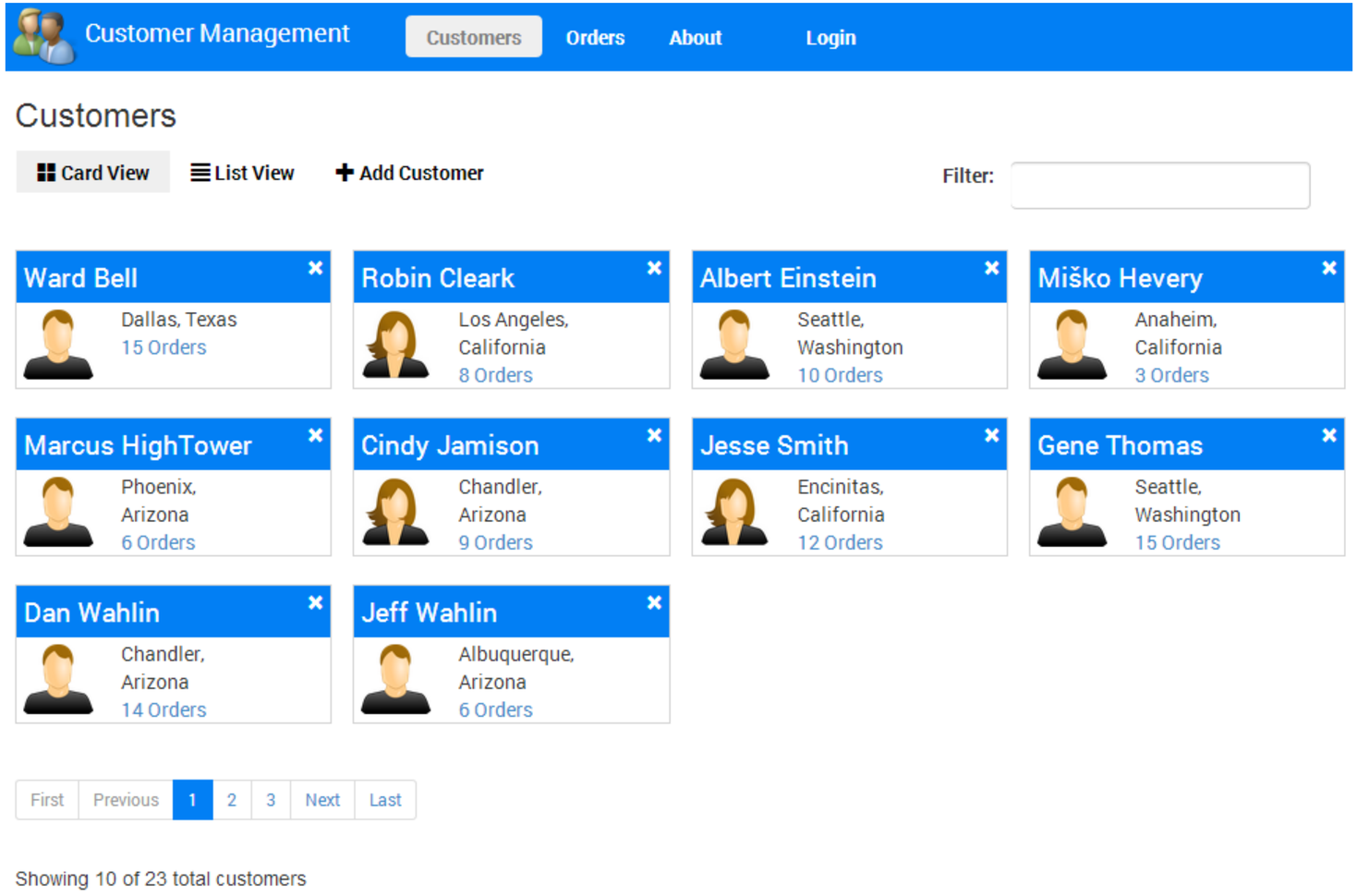
AngularJS Directives

Bonus Content

The CustomerManager Application



https://github.com/DanWahlin/CustomerManagerStandard

Building a Custom Validation Directive

Creating a Unique Value Directive

- A view named **customerEdit.html** requires a unique email address as customers are added/modified:

The wcUnique Directive

```
<input type="email" name="email"
        class="form-control"
        ng-model="customer.email"
        ng-model-options="{ updateOn: 'blur' }"
        wc-unique
        wc-unique-key="{{customer.id}}"
        wc-unique-property="email"
        required />
```

wcUnique Directive Shell Code

```
(function () {

    var injectParams =['$q', 'dataService'];
    var wcUniqueDirective = function ($q, dataService) {
        return {
            restrict: 'A',
            require: 'ngModel',
            link: function (scope, element, attrs, ngModel) {
                ...
            }
        };
    };

    wcUniqueDirective.$inject = injectParams;

    angular.module('customersApp').directive('wcUnique', wcUniqueDirective);

}());
```

Using $asyncValidators

```
link: function (scope, element, attrs, ngModel) {
    ngModel.$asyncValidators.unique = function (modelValue,
        viewValue) {
        var deferred = $q.defer(),
            currentValue = modelValue || viewValue,
            key = attrs.wcUniqueKey,
            property = attrs.wcUniqueProperty;

         ...
    };
}
```

Resolving or Rejecting

```
dataService.checkUniqueValue(key, property, currentValue)
  .then(function (unique) {
      if (unique) {
          deferred.resolve(); //It's unique
      }
      else {
          deferred.reject(); //Add unique to $errors
      }
});

return deferred.promise;
```

Building an Ajax Overlay Directive

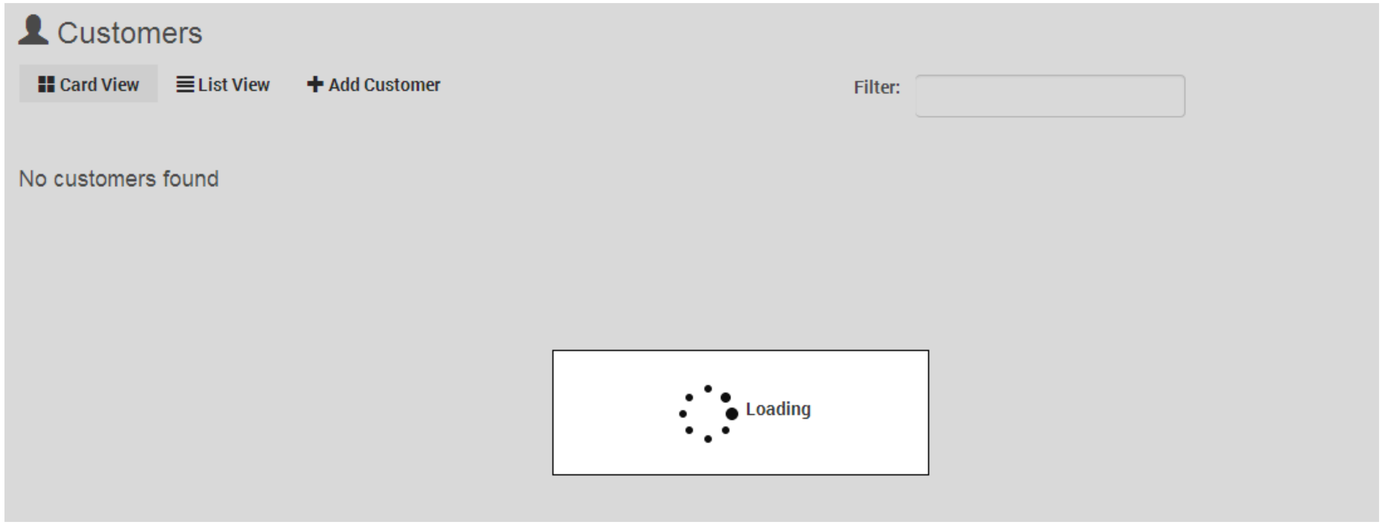
Using $http Interceptors

- How do you intercept XHR requests or responses?
  - Handle errors globally
  - Check returned authentication status codes
  - Pre-process or post-process requests

- $httpProvider can be used to register custom $http interceptors

Using $http Interceptors in a Directive

- As XHR requests are made an overlay can be shown:



- **wcOverlay** directive captures XHR requests/responses and shows/hides an overlay

The wcOverlay Directive

- The wcOverlay directive performs several tasks:
  - Monitors XHR requests and responses
  - Displays an overlay as a request is started
  - Hides the overlay as the response returns
  - Accesses the DOM to show/hide overlay content

```
<div wc-overlay wc-overlay-delay="300">
    <br /><img src="/Content/images/spinner.gif" />
    Loading
</div>
```

The wcOverlay Shell Code

```
(function () {
    var injectParams =['$q', '$timeout', '$window', 'httpInterceptor'];
    var wcOverlayDirective = function ($q, $timeout, $window, httpInterceptor) {
        return {
            restrict: 'EA',
            transclude: true,
            scope: { wcOverlayDelay: "@" },
            template: '<div id="overlay-container" class="overlayContainer">' +
                            '<div id="overlay-background" class="overlayBackground"></div>' +
                            '<div id="overlay-content" class="overlayContent" data-ng-transclude>' +
                            '</div>' +
                        '</div>',
            link: function (scope, element, attrs) { }
        };
    };
    wcOverlayDirective.$inject = injectParams;

    var wcDirectivesApp = angular.module('wc.directives', []);

    wcDirectivesApp.directive('wcOverlay', wcOverlayDirective]);
}());
```

Monitoring XHR Requests/Responses

```
wcDirectivesApp.factory('httpInterceptor', function () {
    return {}; //Directive will hook-in request/response handlers
});

//Hook httpInterceptor factory into the $httpProvider interceptors
//so that we can monitor XHR calls
wcDirectivesApp.config(['$httpProvider', function ($httpProvider)
{
    $httpProvider.interceptors.push('httpInterceptor');
}]);
```

Handling Request/Response/Error

```
function wireUpHttpInterceptor() {
    httpInterceptor.request = function (config) {
        processRequest();
        return config || $q.when(config);
    };
    httpInterceptor.response = function (response) {
        processResponse();
        return response || $q.when(response);
    };
    httpInterceptor.responseError = function (rejection) {
        processResponse();
        return $q.reject(rejection);
    };
}
```

Hook factory into $httpProvider interceptors

Building a Menu Highlighter Directive

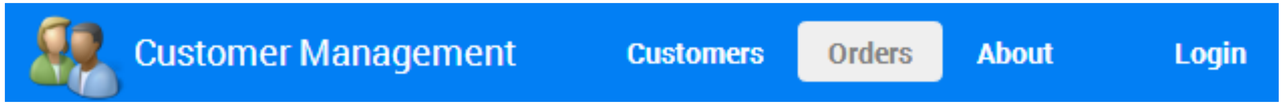
Highlighting Navbar Menu Items

- The Navbar view handles highlighting menu items as they're clicked using the highlight() function:

```
<ul class="nav navbar-nav nav-pills navBarPadding">

    <li data-ng-class="{'active':highlight('/orders')}">
        <a href="#/orders">Orders</a>
    </li>

    ...

</ul>
```

NavbarController

```
vm.highlight = function (path) {
    return $location.path().substr(0, path.length) === path;
};
```



Simplifying Navbar Menu Highlighting

- The menuHighlighter directive simplifies highlighting menu items:

```
<ul class="nav navbar-nav nav-pills navBarPadding"
    menu-highlighter highlight-class-name="active">
    <li><a href="#/customers">Customers</a></li>
    <li><a href="#/orders">Orders</a></li>
    <li><a href="#/about">About</a></li>
    <li id="nav-login" data-ng-click="loginOrOut()">
        <a data-href="#/login">{{ loginLogoutText }}</a>
    </li>
</ul>
```

The menuHighlighter Directive

```
(function () {

    var injectParams = ['$location'];
    var menuHighlighter = function ($location) {
        return {
            restrict: 'A',
            scope: {
                highlightClassName: '@'
            },
            link: function (scope, element) { ... }
        }
    }
    menuHighlighter.$inject = injectParams;
    angular.module('wc.directives')
      .directive('menuHighlighter', menuHighlighter);

}());
```

The menuHighlighter link() Function

```
var link = function (scope, element) {
    function setActive() {
        var path = $location.path();
        var className = scope.highlightClassName || 'active';
        if (path) {
            angular.forEach(element.find('li'), function (li) {
                var anchor = li.querySelector('a');
                var href = (anchor && anchor.href) ? anchor.href :
                            anchor.getAttribute('data-href').replace('#', '');
                var trimmedHref = href.substr(href.indexOf('#/') + 1, href.length);
                var basePath = path.substr(0, trimmedHref.length);
                if (trimmedHref === basePath) {
                    angular.element(li).addClass(className);
                } else {
                    angular.element(li).removeClass(className);
                }
            });
        }
    }
    setActive();
    scope.$on('$locationChangeSuccess', setActive);
};
```

Building a Custom Validator Directive