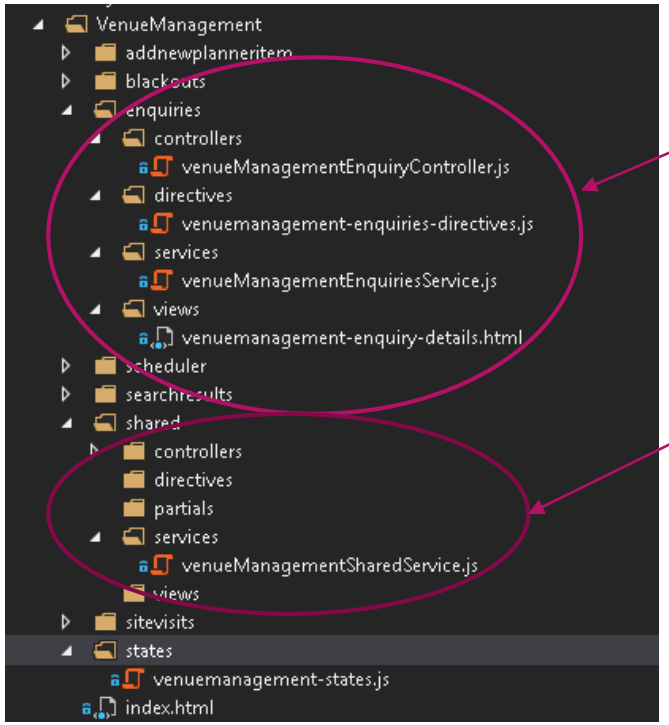
AngularJS

STYLE & CODING STANDARDS

ROBIN CULLIMORE

Folder structure



Each section or ‘sub module’ has it’s own: Controllers, Directives, Services and Views folder

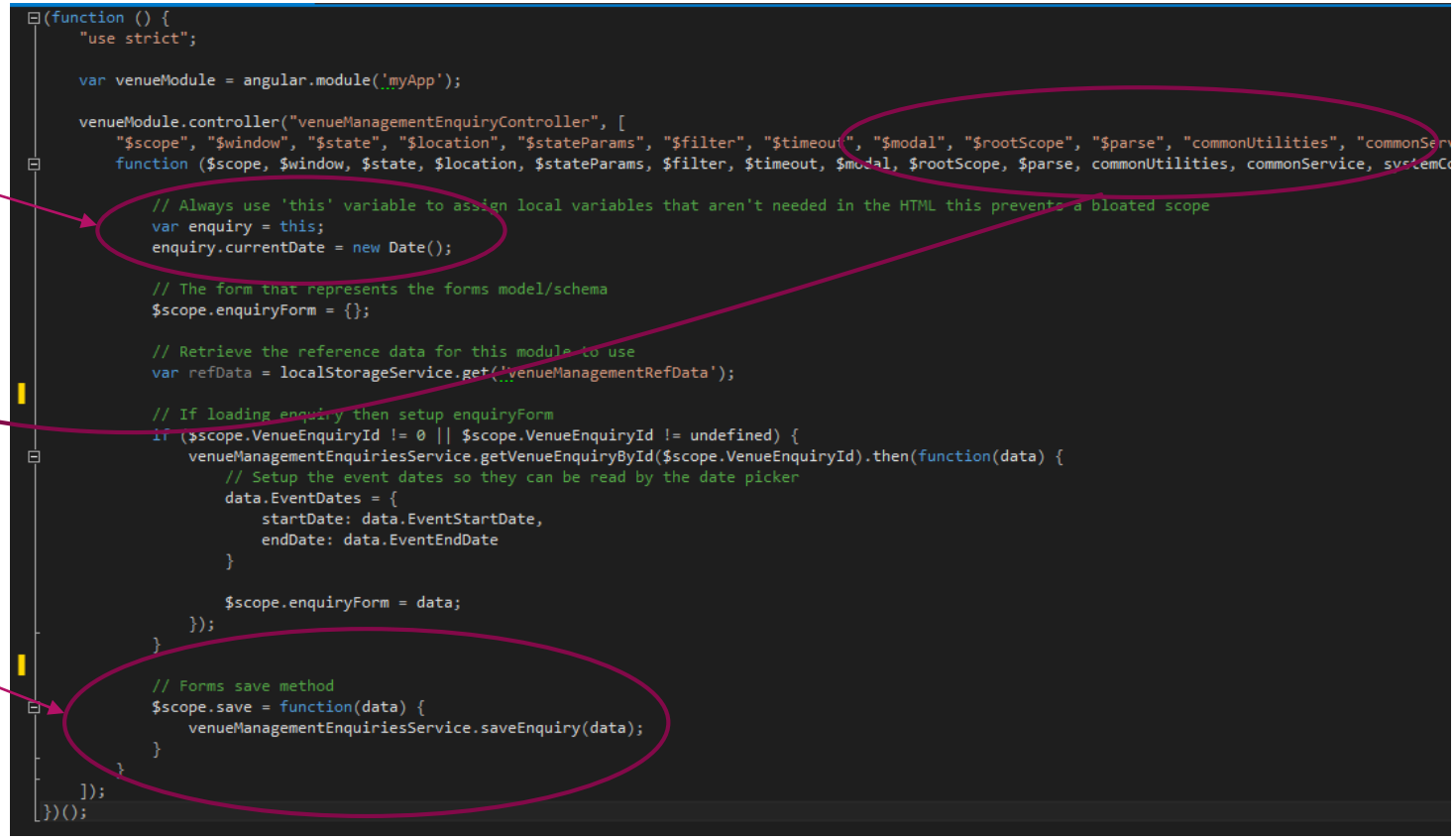
The shared folder has an additional folder for partial views. Only generic or reusable components should go into the shared folder.

Controllers

Using a ‘this’ variable will help reduce scope bloat

Injected names or variables will be fully qualified and not abbreviated.

The aim is lean controllers. No business logic should be implemented in the controllers – this should go into the related service



```
(function () {
    "use strict";

    var venueModule = angular.module('myApp');

    venueModule.controller("venueManagementEnquiryController", [
        "$scope", "$window", "$state", "$location", "$stateParams", "$filter", "$timeout", "$modal", "$rootScope", "$parse", "commonUtilities", "commonServ
        function ($scope, $window, $state, $location, $stateParams, $filter, $timeout, $modal, $rootScope, $parse, commonUtilities, commonService, systemCo

            // Always use 'this' variable to assign local variables that aren't needed in the HTML this prevents a bloated scope
            var enquiry = this;
            enquiry.currentDate = new Date();

            // The form that represents the forms model/schema
            $scope.enquiryForm = {};

            // Retrieve the reference data for this module to use
            var refData = localStorageService.get('venueManagementRefData');

            // If loading enquiry then setup enquiryForm
            if ($scope.VenueEnquiryId != 0 || $scope.VenueEnquiryId != undefined) {
                venueManagementEnquiriesService.getVenueEnquiryById($scope.VenueEnquiryId).then(function(data) {
                    // Setup the event dates so they can be read by the date picker
                    data.EventDates = {
                        startDate: data.EventStartDate,
                        endDate: data.EventEndDate
                    }

                    $scope.enquiryForm = data;
                });
            }

            // Forms save method
            $scope.save = function(data) {
                venueManagementEnquiriesService.saveEnquiry(data);
            }
        }
    ]);
})();
```

Directives

```
(function () {
    'use strict';

    var myApp = angular.module('venuemanagement-enquiries-directives', []);

    myApp.directive('venuemanagementEnquiriesDetail', [
        '$state', '$stateParams', '$filter', function () {
            return {
                restrict: 'AE',
                templateUrl: '../Scripts/app/VenueManagement/enquiries/views/venuemanagement-enquiry-details.html',
                scope : false,
                controller: 'venueManagementEnquiryController'
            };
        }
    ]);
}());
```

Isolate scope as much as possible or use ‘scope false’ to share the parents – we should never be using $parent functionality.
We should remember $scope is being removed in Angular V2.

Services

```
(function () {
    "use strict";

    var myAppModuleVM = angular.module('myApp');
    myAppModuleVM.factory('venueManagementEnquiriesService',
        ['$resource', 'commonUtilities', 'streamingUtilities', '$cacheFactory', 'localStorageService', "$route", "$filter", "venueManagementSharedService",
            function ($resource, commonUtilities, streamingUtilities, $cacheFactory, localStorageService, $route, $filter, venueManagementSharedService) {

                return {
                    saveEnquiry: function (data) {

                        // Enquiries will always be all day events so set hours to all day
                        var startDate = new Date(data.EventDates.startDate);
                        startDate.setHours(7);
                        data.EventStartDate = startDate;
                        var endDate = new Date(data.EventDates.endDate);
                        endDate.setHours(23);
                        data.EventEndDate = endDate;

                        if (data.Id == undefined || data.Id == 0) {
                            venueManagementSharedService.addData('enquiries', null, data, "Enquiry successfully added");
                        } else {
                            venueManagementSharedService.updateData('enquiries/{0}', [data.Id], data, "Enquiry successfully updated");
                        }
                    },
                    getVenueEnquiryById: function(id) {
                        return venueManagementSharedService.getData('enquiries/{0}', [id]);
                    }
                }
            }
        ]);
})();
```

Services should now contain all the business logic.

They still reach out to the generic service that deals with all the API calls. This is a much more OO way of structuring services. The functions will work just like a class methods.

The specific services can also be used to pass parameters or objects between controllers.

Views

No more than one empty line in the code, ever!!!!

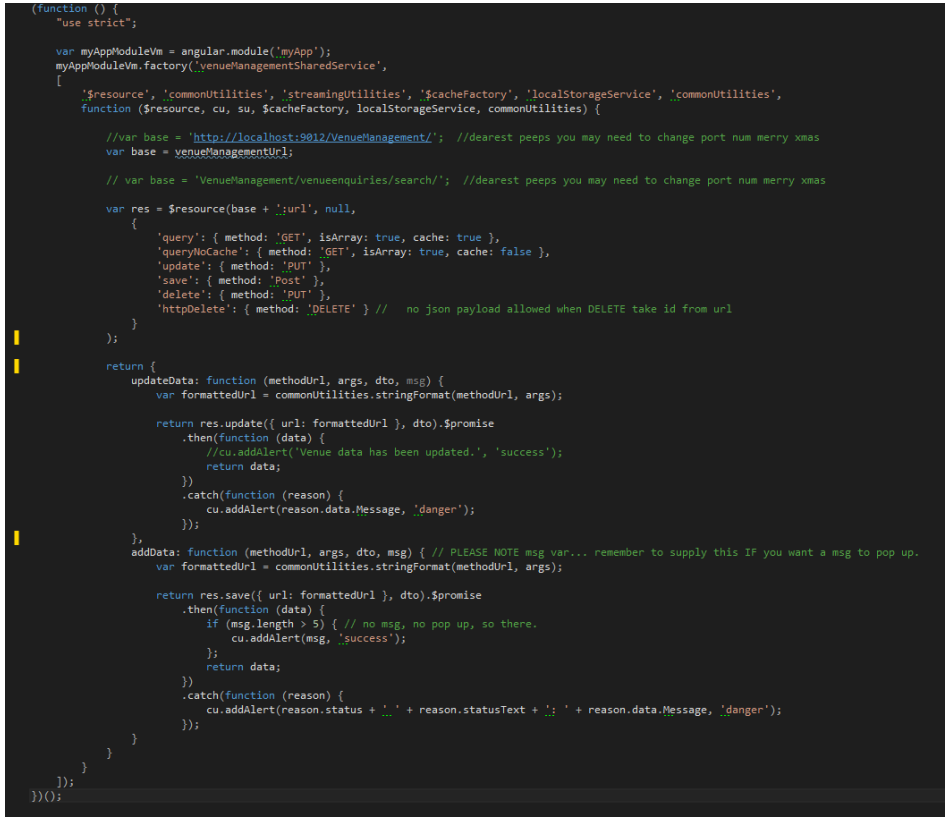
The main object will be named after the form that is being submitted.

No more RHS-Forms or ‘ob’ / ‘cd’.

```
<h2>Enquiry</h2>


<div class="col-md-12 row">
    <ng-form role="form" class="" name="forms.enquiryForm" style="margin-bottom: 20px; display: block">
        <fieldset>
            <div>
                <div class="form-group col-sm-6" ng-class="{'has-error': forms.enquiryForm.EnquirerEmailAddress.$invalid}">
                    <label for="EnquirerEmailAddress" class="control-label">Enquirer Email Address</label>
                    <div>
                        <input type="email"
                               class="form-control"
                               name="EnquirerEmailAddress"
                               ng-minlength="10"
                               ng-model="enquiryForm.EnquirerEmailAddress"
                               ng-required="EnquirerTelephoneNumber == null" />
                    </div>
                    <div class="col-sm-2">
                        <span ng-show="forms.enquiryForm.EnquirerEmailAddress.$error.required" class="help-block">Required</span>
                    </div>
                </div>

                <div class="form-group col-sm-12" ng-class="{'has-error': forms.enquiryForm.EventNotes.$invalid}">
                    <label for="EventNotes" class="control-label">Event Notes</label>
                    <div>
                        <textarea type="text"
                                  ng-minlength="30"
                                  class="form-control rhs-expand"
                                  name="EventNotes"
                                  placeholder="Enter notes here ..."
                                  data-ng-model="enquiryForm.EventNotes" />
                    </div>
                </div>
            </div>
        </fieldset>
        <div class="modal-footer text-right">
            <button class="btn btn-primary" ng-disabled="forms.enquiryForm.$pristine"
                    ng-click="save(enquiryForm); forms.detailenquiryFormsForm.$setPristine()">
                Save Details
            </button>

            <button class="btn btn-warning" ng-click="closeWindow()">Cancel</button>
        </div>
    </ng-form>
</div>
```

Shared Services

Shared service remains unchanged



```
(function () {
    "use strict";

    var myAppModuleVm = angular.module('myApp');
    myAppModuleVm.factory('venueManagementSharedService',
    [
        '$resource', 'commonUtilities', 'streamingUtilities', '$cacheFactory', 'localStorageService', 'commonUtilities',
        function ($resource, cu, su, $cacheFactory, localStorageService, commonUtilities) {

            //var base = 'http://localhost:9012/VenueManagement/';  //dearest peeps you may need to change port num merry xmas
            var base = venueManagementUrl;

            // var base = 'VenueManagement/venueenquiries/search/';  //dearest peeps you may need to change port num merry xmas

            var res = $resource(base + ':url', null,
                {
                    'query': { method: 'GET', isArray: true, cache: true },
                    'queryNoCache': { method: 'GET', isArray: true, cache: false },
                    'update': { method: 'PUT' },
                    'save': { method: 'Post' },
                    'delete': { method: 'PUT' },
                    'httpDelete': { method: 'DELETE' } //   no json payload allowed when DELETE take id from url
                }
            );

            return {
                updateData: function (methodUrl, args, dto, msg) {
                    var formattedUrl = commonUtilities.stringFormat(methodUrl, args);

                    return res.update({ url: formattedUrl }, dto).$promise
                        .then(function (data) {
                            //cu.addAlert('Venue data has been updated.', 'success');
                            return data;
                        })
                        .catch(function (reason) {
                            cu.addAlert(reason.data.Message, 'danger');
                        });
                },
                addData: function (methodUrl, args, dto, msg) { // PLEASE NOTE msg var... remember to supply this IF you want a msg to pop up.
                    var formattedUrl = commonUtilities.stringFormat(methodUrl, args);

                    return res.save({ url: formattedUrl }, dto).$promise
                        .then(function (data) {
                            if (msg.length > 5) { // no msg, no pop up, so there.
                                cu.addAlert(msg, 'success');
                            };
                            return data;
                        })
                        .catch(function (reason) {
                            cu.addAlert(reason.status + ' ' + reason.statusText + ': ' + reason.data.Message, 'danger');
                        });
                }
            }
        }
    ]);
})();
```

Aims

- Lean controllers
- More components, smaller components
- Code simplicity ahead of “fewer lines of code”
- Scope isolation with truly “reusable” components
- Design patterns and coding standards just as important on the front-end as the back-end

Sources

- https://github.com/mgechev/angularjs-style-guide
- http://artandlogic.com/2013/05/ive-been-doing-it-wrong-part-1-of-3/