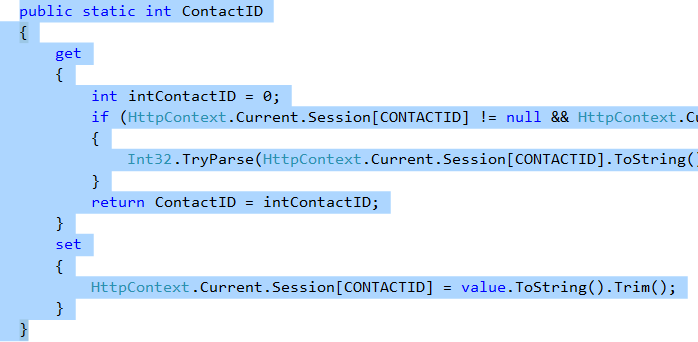
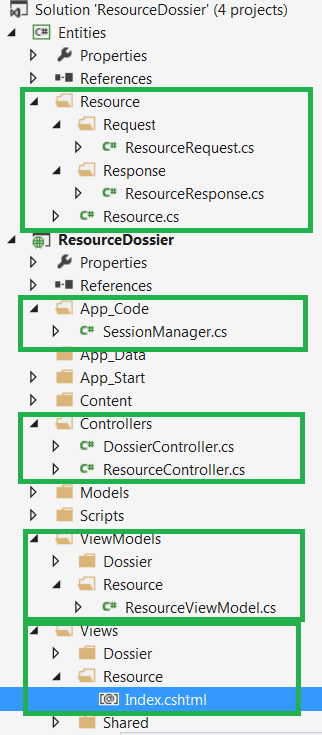
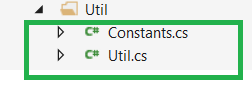
**Documentation of standards for UI component implementation:**

This document is outlined with list of styles and script to be included and creation of entity, request, response and view models.

1. Coding stands has been listed in wiki:
   1. <https://github.com/ZeroChaos/standards/wiki>
2. Below are common scripts & styles to be included:
   1. **Scripts**:
      1. jquery.js(v1.11.2)
      2. jquery-migrate.js
      3. jquery-ui.js
      4. jquery.uniform.min.js
      5. jquery.form.js
      6. chosen.jquery.min.js
      7. jquery.validate.js
      8. jquery.validate.unobtrusive.js
      9. jquery.unobtrusive-ajax.js
      10. kendo.all.min.js,
      11. kendo.aspnetmvc.min.js,
      12. kendo.grid.min.js,
      13. kendo.modernizr.custom.js
      14. bootstrap.js(Bootstrap v3.1.1),
      15. bootstrap-dialog.min.js,
      16. bootstrap-timepicker.js,
      17. bootstrap-datepicker.js
      18. common.js
      19. app-common.js
      20. dossier.js
   2. **Styles**:
      1. bootstrap.min.css(v2.3.2)
      2. kendo.common-bootstrap.min.css
      3. kendo.metro.min.css
      4. kendo.dataviz.min.css
      5. kendo.dataviz.default.min.css
      6. timepicker.css
      7. datepicker.css
      8. bootstrap-multiselect.css
      9. bootstrap-responsive.css
      10. bootstrap-modal.css
      11. font-awesome.min.css
      12. style.css
      13. uniform.default.css
      14. style-responsive.css
      15. common.css
      16. DT\_bootstrap.css
      17. jquery-ui.css
      18. bootstrap-responsive.min.css
      19. app-common.css
3. In order to create entity, request, response and view models below are the details:
   1. **Entity**: Entity class should contain all the properties related to a table. Do not add the CreatedDate, CreatedByID, ModifiedDate and ModifiedByID properties to the entity classes. These fields are processed only by stored procedures and triggers. If the entity object must map to the CreatedByID or the ModifiedByID columns, then inherit from the AuditableEntity class instead of the EntityBase class. Create this class under namespace ZeroChaos.Business.Entities.
      1. For example:
         1. public class Resource : AuditableEntity
         2. {
         3. public int ResourceID{ get; set; }
         4. public bool ResourceFirstName { get; set; }
         5. public bool ResourceLastName { get; set; }
         6. }
   2. **Response**: Based on the requirement, response object should contain all the properties and entities to bind the data in view, under namespace namespace ZeroChaos.Business.Entities.
      1. Ex:
         1. public class ResourceResponse
         2. {
         3. public int ResourceID { get; set; }
         4. public List<Languages> Languages { get; set; }
         5. }
   3. **Request**: Based on our requirement we need to create a new RequestModel as below. This request model is used to send data to service layer for get/insert data into database. Consider, a service method which expects more than two or three parameters and returns response object. In this case, create request entity which holds all these properties and make service method to expect only one parameter which is request object. Create this under namespace ZeroChaos.Business.Entities.
      1. Ex:
         1. public class ResourceRequest
         2. {
         3. public int ResourceID { get; set; }
         4. public int UserPreferredLanguageID { get; set; }
         5. public int ReqCandID { get; set; }
         6. }
         7. Service Method declaration should be:
            1. public ResourceResponse GetResourceDossierResponseByResourceID(ResourceRequest resourceRequest);
   4. **View model**: Based on our requirement we need to create a new viewmodel as below. We need to map ResourceResponse with ResourceViewModel, response object will be constructed in service layer and **business logic should be written in service layer**. Viewmodel should be directly used by view without writing any business logic in mvc side. Create this under namespace ZCW.MVC.ViewModels.
      1. Ex:
         1. public class ResourceViewModel
         2. {
         3. public int ResourceID { get; set; }
         4. public List<Languages> Languages { get; set; }
         5. }
      2. Consume service method as below:
         1. IResourceService resourceService = new ResourceService();
         2. ResourceResponse response = resourceService.GetResourceDossierResponseByResourceID(resourceRequest);
      3. Map the response to viewModel in **MappingHelper.cs**.
         1. response.ToResourceViewModel()
         2. Method signature:
            1. public static ResourceViewModel ToResourceViewModel(this ResourceResponse resourceResponse)

Assign all properties to view model from response.

* + - 1. Use this model to bind data in view.

1. **SessionManager**: Session variable will be added in this static class.
   1. 
   2. Will be accessed as SessionManager.ContactID
2. **Utils**: This class will contain generic validation checks like email address validation, phone number validation and rounding numbers. Created under namespace ZeroChaos.Utilities.
3. Find the folder structure format as highlighted below:
   1. 
   2. 
4. **Working:** Make sure to create Response and Request model for your page. Response model basically includes control visibility/data based on business. Request model includes data that needs to be sent to database for saving. Based on the table entities required for page we need to build Response model and include properties needed for a page (Only required properties needs to be included in response model). Once after sending Response model to MVC from service layer, we need to map Response model with Viewmodel and send it to view without any business logic change in MVC side. After modal binding, map the Request model from ViewModel and send this model request to service method. Based on requirement build your table entity from Request model and insert data to database.