

[Sign up for our free weekly Web Developer Newsletter.](#)**CODE
PROJECT®**
For those who code[home](#)[articles](#)[quick answers](#)[discussions](#)[features](#)[community](#)[help](#)

Design and Develop a website using ASP.NET MVC 4, EF, Knockoutjs and Bootstrap : Part - 1

**Anand Ranjan Pandey**, 16 Jan 2013

CPOL

Rate this:



4.93 (111 votes)

Design a website architecture that must be simple, easily understandable by any web designer using asp.net MVC, EF, Knockoutjs and Bootstrap

[Download Application.zip](#)

Another MVC Application: Introduction

All websites are growing faster these days, and once it grows, it is very hard to write, organize and maintain. As we add new functionality or developer to a project, any large web applications with poor design may be go out of control. So the idea behind this article is to design a website architecture that must be simple, easily understandable by any web designer (beginner to intermediate) and Search Engines. For this article I am trying to design a website for any individuals to maintain their contact details online. However, in future, the same application could be used by large community all over the world with added functionality and modules. So, the design should be easily adaptable in order to cope with the future growth of business.

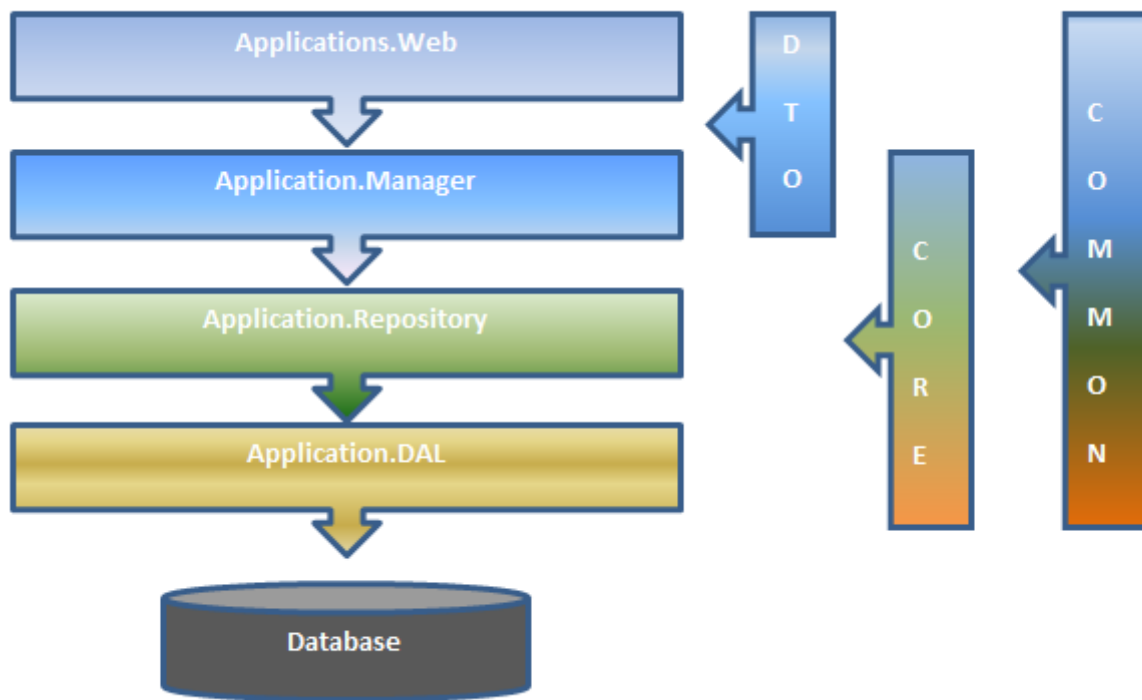
In this article I will talk about creating and designing User Interface (UI) in such a manner so that UI will be separated from the business logic, and can be created independently by any designer/developer. For this part we will use ASP.Net MVC, Knockout JQuery and Bootstrap.

The Second part of this series is more about database design and implementing business logic using structured layers using SQL Server 2008, Entity Framework, and Castle Windsor for Dependency Injection.

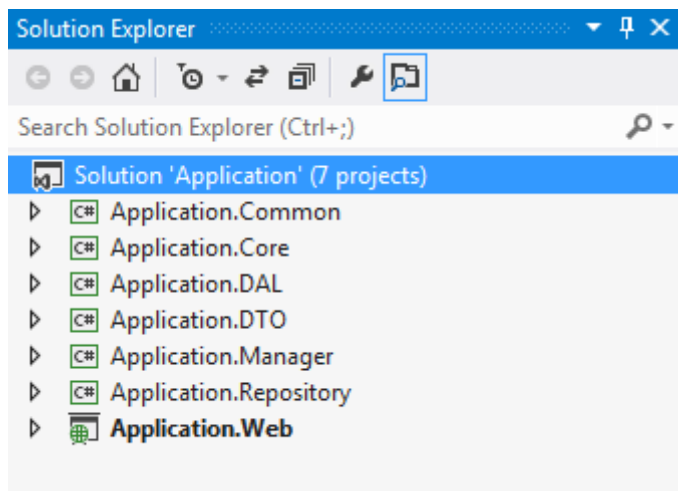
[Design and Develop a website using ASP.NET MVC 4, EF, Knockoutjs and Bootstrap : Part - 2](#)

Separation of Concern: Primary Objective

The key concept is stripping out most or all logic. Logic should not be bound to a page. What if we need to re-use the logic from one page in another? In that case we will be tempted to copy-and-paste. If we are doing this then our project will become maintainable. Another important concept is to separate data access layer from any business logic, as we are planning to use Entity Framework this is less of a problem as EF should already have this end separate. We should be able to easily move all our EF files to another project and simply add a reference to the projects that need it. Below is the high level design:



And, the final solution will look like below image in Visual Studio :



Seven Projects in One Solution : Is it required ?

It is all about your decision...□ The proposed designed will offer some rather relevant benefits, which include:

- **Separation of Concern:** Design should allow clear and defined layers; means segregate application into distinct areas whose functionality does not overlap. such that UI-designers can focus on their job without dealing with the business logic (Application.Web), and the core developer can only work on main business logics (Application.DTO or Application.Manager).
- **Productivity:** It is easier to add new features to existing software, since the structure is already in place, and the location for every new piece of code is known beforehand, so any issue can be easily identified and separated to cope with complexity, and to achieve the required engineering quality factors such as robustness, adaptability, maintainability, and reusability.
- **Maintainability:** It is easier to maintain application, due to clear and defined structure of the code is visible and known, so it's easier to find bugs and anomalies, and modify them with minimal risk.
- **Adaptability:** New technical features, such a different front ends, or adding a business rule engine is easier to achieve, as your software architecture creates a clear separation of concerns.

- **Reusability:** Reusability is another concern on designing any application, because it is one of the main factors to decrease the total cost of ownership, and our design should consider to what extent we can reuse the created Web application and different layers.

In last section of this article, we will discuss the functionality of each individual layer's in details.

Tools & Technology

To achieve the final solution, we need below tools/dll:

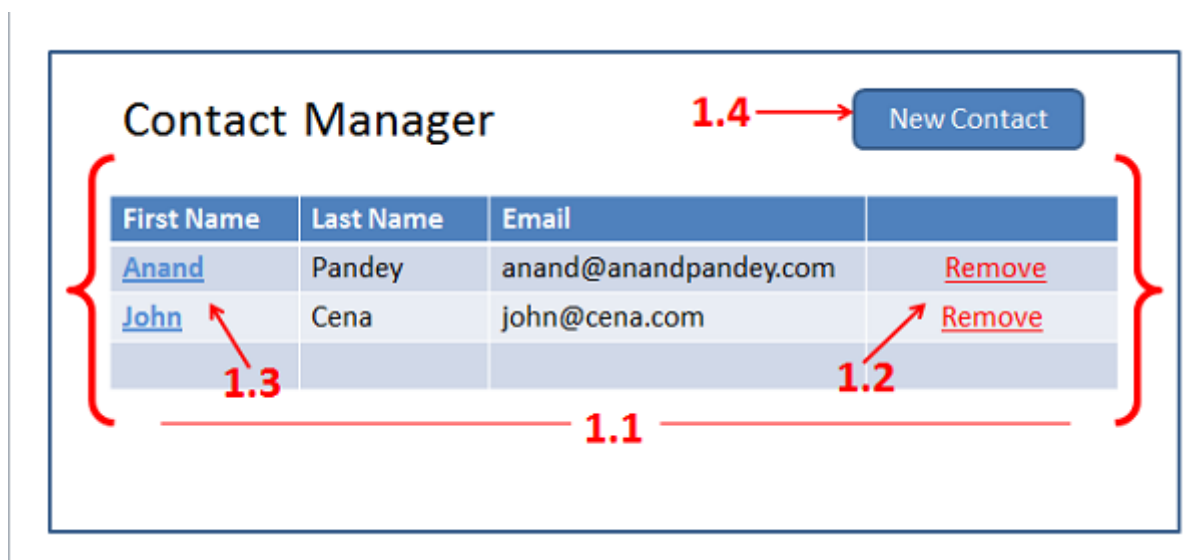
- Visual Studio 2012
- ASP.NET MVC 4 with Razor View Engine
- Entity Framework 5.0
- Castle Windsor for DI
- SQL Server 2008/2012
- Knockout.js & JQuery
- Castle Windsor for DI
- Bootstrap CSS

What we are trying to achieve: Requirement

Screen 1: Contact List - View all contacts

- 1.1 This screen should display all the contacts available in Database.
- 1.2 User should be able to delete any contact.
- 1.3 User should be able to edit any contact details.
- 1.4 User should be able to create a new contact.

Initial sketch:



Screen 2: Create New Contact

This screen should display one blank screen to provide functionalities as.

- 2.1 User should be able to Enter his/her First name, Last Name and Email Address.
- 2.2 User should be able to add any number of Phone numbers by clicking on Add numbers.
- 2.3 User should be able to remove any phone number.
- 2.4 User should be able to add any number of Addresses by clicking on Add new address.
- 2.5 User should be able to remove any address.
- 2.6 Click on save button should save Contact details in Database and user will return back in Contact List page.

2.7 Click on Back to Profile button should return back the user to Contact List page.

Initial sketch:

Create New Contact

Profile Information

First Name Last Name Email

Phone Information

Select Phone Type ... Number Remove

Add New Phone

Address Information

Select Address Type ... Remove

Address Line 1 Address Line 2 Country

State City Zip Code

Add New Address

Save Profile

Back to Profile List

Screen 3: Update Existing Contact

This screen should display screen with selected contact information details.

- 3.1 User should be able to modify his/her First name, Last Name and Email Address.
- 3.2 User should be able to modify /delete/Add any number of Phone numbers by clicking on Add numbers or delete link.
- 3.3 User should be able to modify /delete/Add any number of Addresses by clicking on Add new address or delete link.
- 3.4 Click on save button should update Contact details in Database and user will return back in Contact List page.
- 3.5 Click on Back to Profile button should return back the user to Contact List page.

>

Initial Sketch:

Edit Contact

3.5 → [Back to Profile List](#)

3.1 → **Profile Information**

Anand Pandey anand@anandpandey.com

Phone Information

Work Phone 1-832-832-8328 Remove

Personal Phone 1-238-238-2382 Remove

3.2 → [Add New Phone](#)

Address Information

Billing Address Remove

10000 Richmond Ave Apt # 1000 USA

Texas Houston 70000

3.3 → [Add New Address](#)

3.4 → [Save Profile](#)

Part 1: Create Web Application (Knockout.js, Asp.Net MVC and Bootstrap): For Designers

Before kick-off the UI part, let us see what benefits we are getting using Knockoutjs and Bootstrap along with ASP.NET MVC 4.

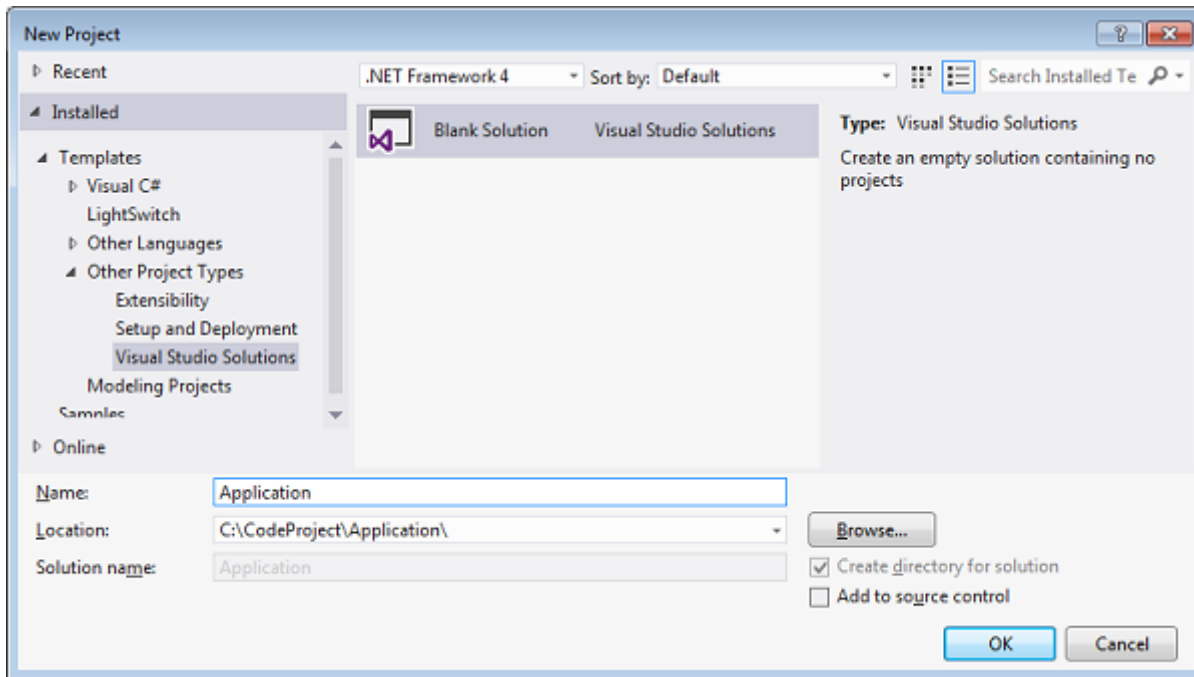
Why Knockoutjs: Knockout is an MVVM pattern that works with a javascript ViewModel. The reason this works well with MVC is that serialization to and from javascript models in JSON is very simple, and it is also included with MVC 4. It allows us to develop rich UI's with a lot less coding and whenever we modify our data, Immediate it reflect in the user interface.

Why Bootstrap: Twitter Bootstrap is a simple and flexible HTML, CSS, and Javascript for popular user interface components and interactions. It comes with bundles of CSS styles, Components and Javascript plugins. It provides Cross platform support, which eliminates major layout inconsistencies. Everything just works! Good documentation and the Twitter Bootstrap's website itself is a very good reference for real life example. And, finally, it saves me plenty of times, as it cut the development time to half with very less testing and almost zero browser issues. Some other benefits we can get by using this framework are:

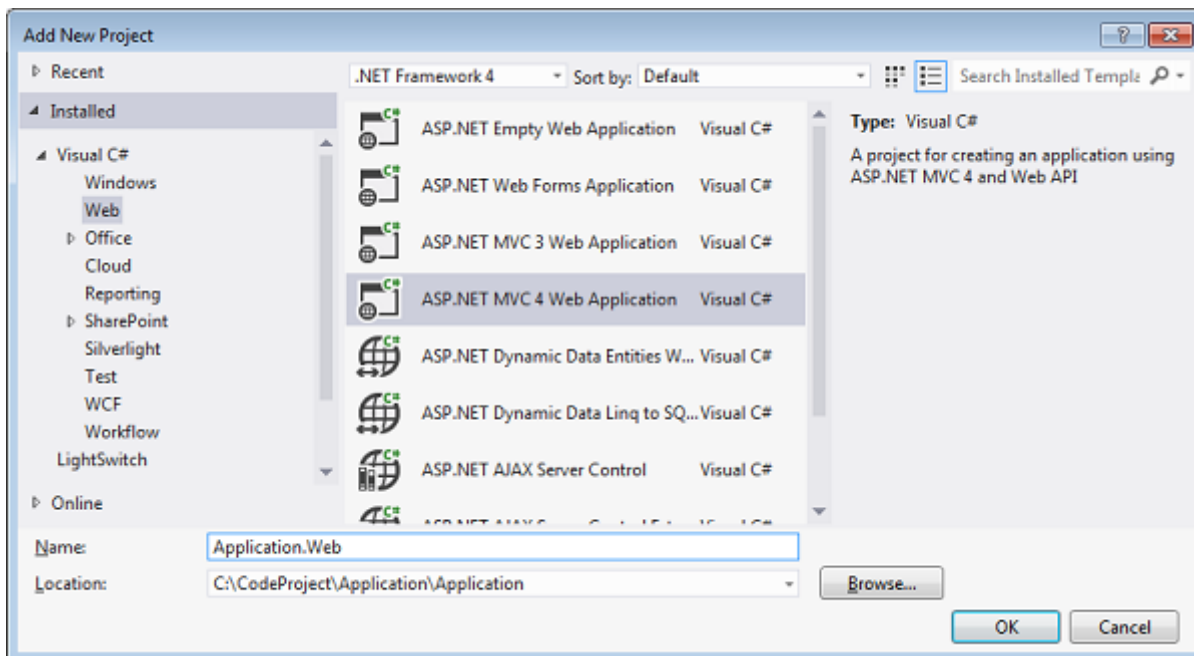
- 12-Column Grid, fixed layout, fluid or responsive layout.
- Base CSS for Typography, code (syntax highlighting with Google prettify), Tables, Forms, Buttons and uses Icons by Glyphicons .
- Web UI Components like Buttons, Navigation menu, Labels, Thumbnails, Alerts, Progress bars and misc.
- Javascript plugins for Modal, Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel and Typehead.

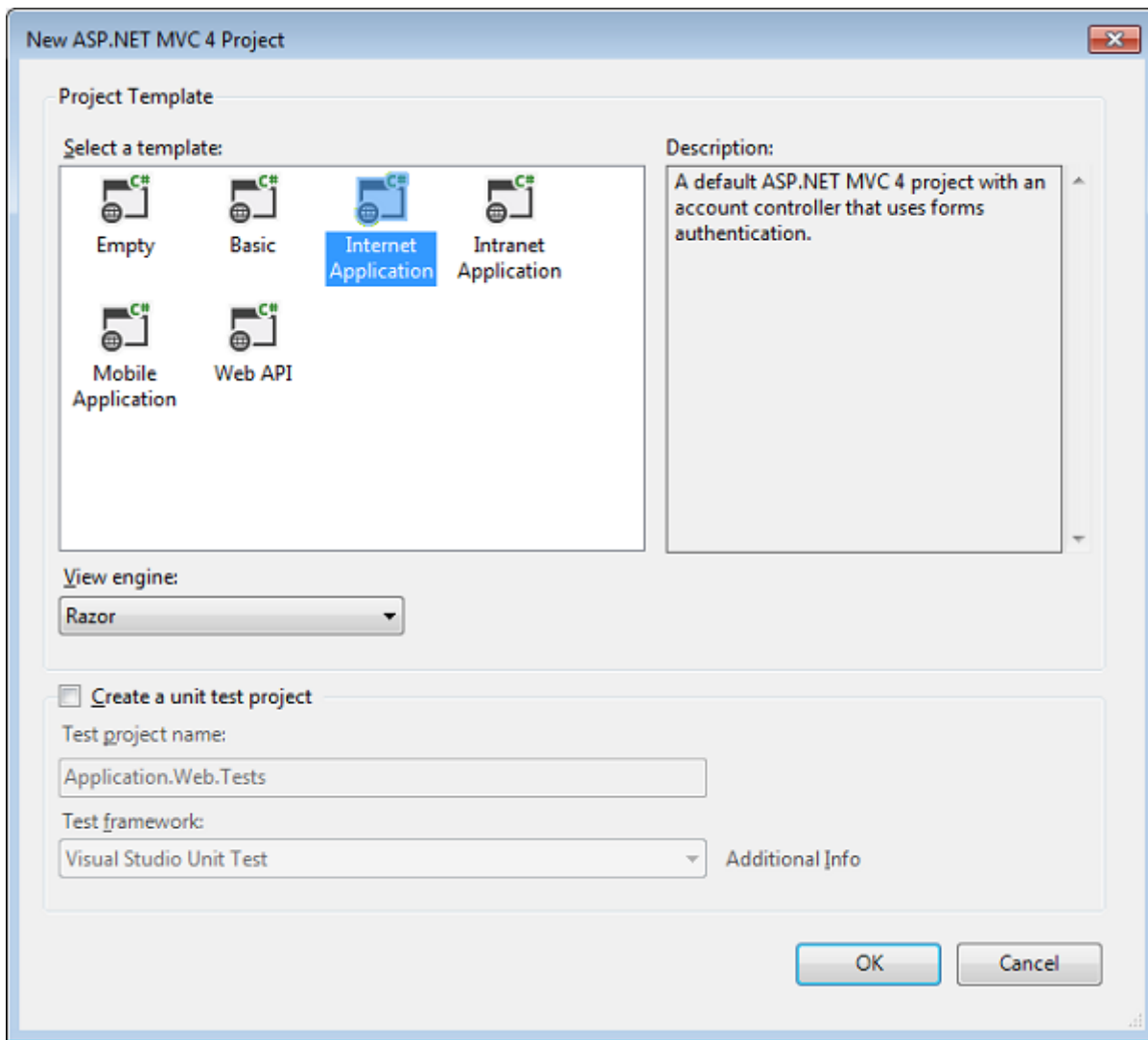
In below steps, we will work through the layout and design to build UI for above requirement by using dummy javascript data.

Step 1: Create a new project as Blank Solution; name it as "Application"

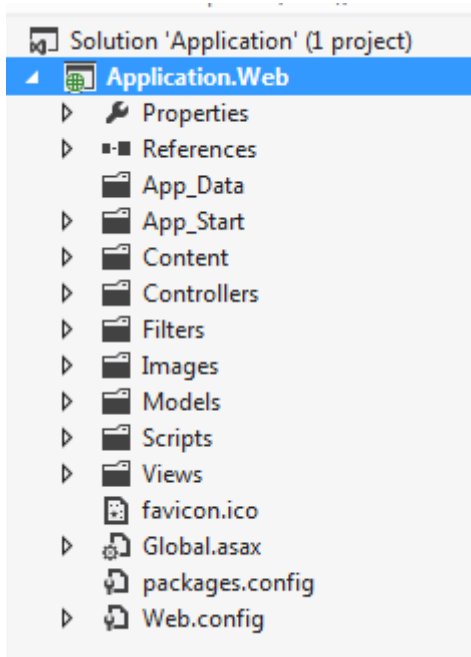


Step 2: Right Click on Solution folder and add new Project of type ASP.NET MVC 4 as an Internet Application Template with View engine as Razor.

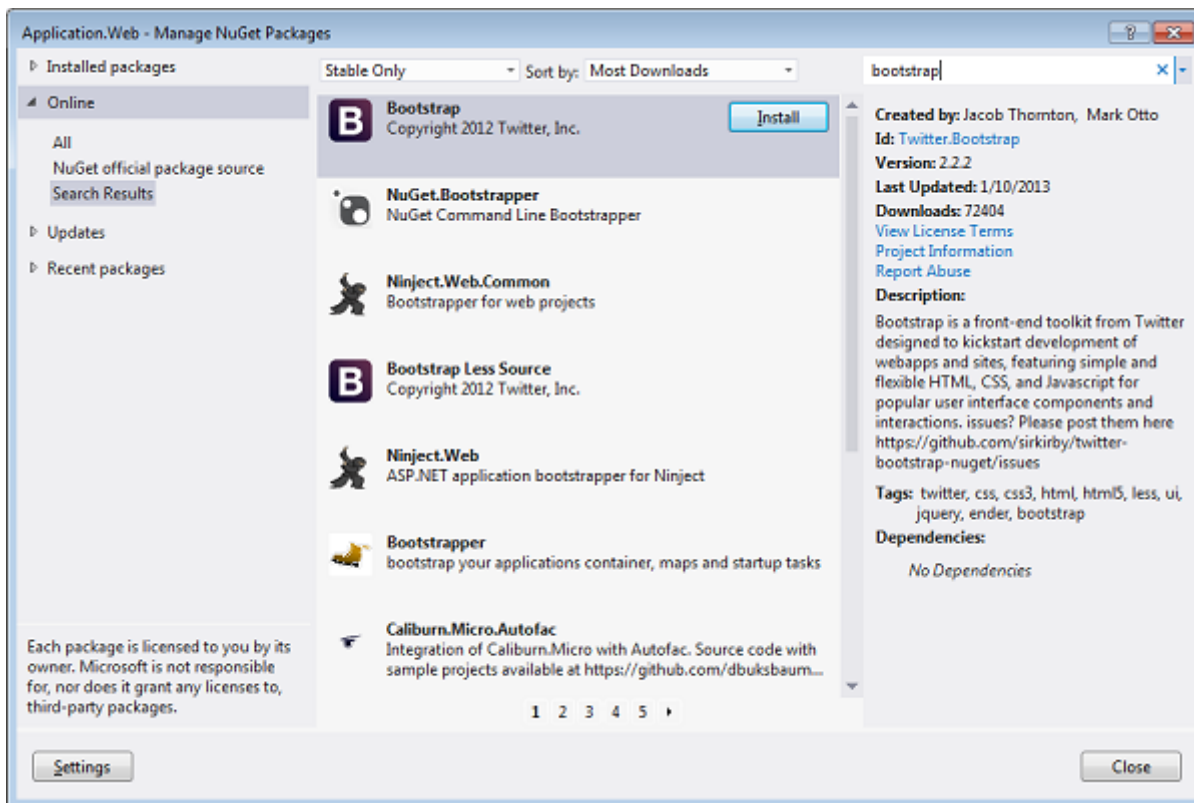




After Step 2 - the project structure should look like the below image



Step 3: Right click on References and click on Manage NuGet Packages. Type Bootstrap on search bar then click on Install button.



Step 4: add below line of code into BundleConfig.cs file under App_Start folder to add Knockoutjs and Bootstrap for every page

Hide Copy Code

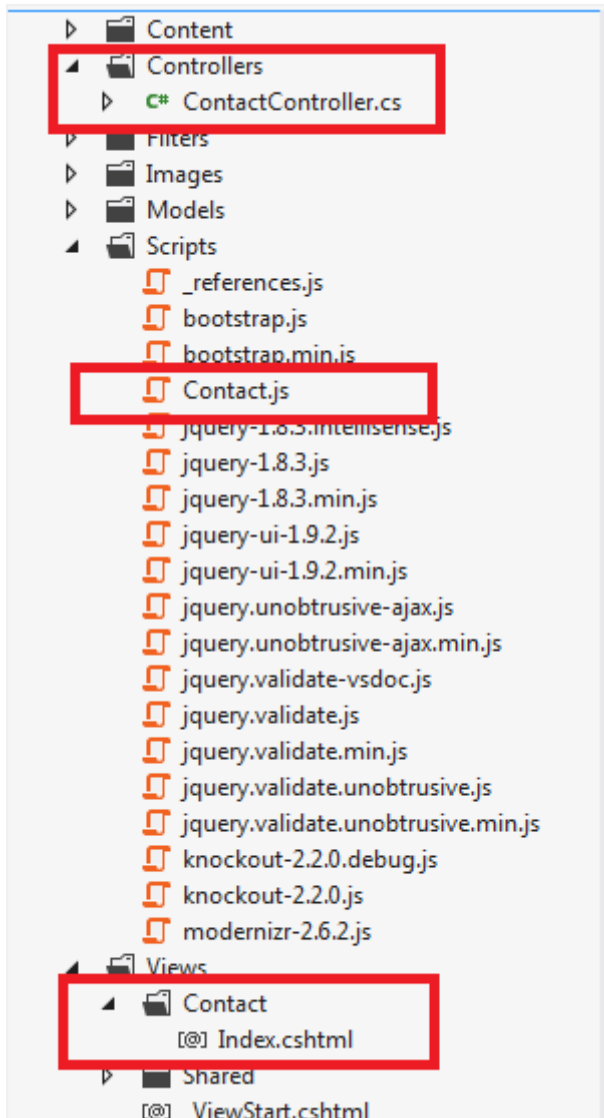
```
bundles.Add(new ScriptBundle("~/bundles/knockout").Include(
    "~/Scripts/knockout-{version}.js"));
bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/bootstrap.css"));
```

Also in _Layout.cshtml file under Views/Shared folder add below line to register knockout files as :

Hide Copy Code

```
@Scripts.Render("~/bundles/knockout")
```

Step 4: Add a new folder name as Contact inside Views, and then add Index.cshtml as new View page. Then add a new Controller name it ContactController.cs inside Controller folder, and add a new Contact.js file under Scripts folder. Refer to below image.



Step 5: Finally modify the default map route in Route.config to point to Contact controller as:

Hide Copy Code

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Contact", action = "Index", id = UrlParameter.Optional }
);
```

And also modify the _Layout.cshtml file inside View/Shared as per the Bootstrap Syntax. Below is the modified code:

Hide Shrink ▲ Copy Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@ViewBag.Title - Contact manager</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width" />
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/knockout")
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
    @RenderSection("scripts", required: false)
</head>
<body>
```

```

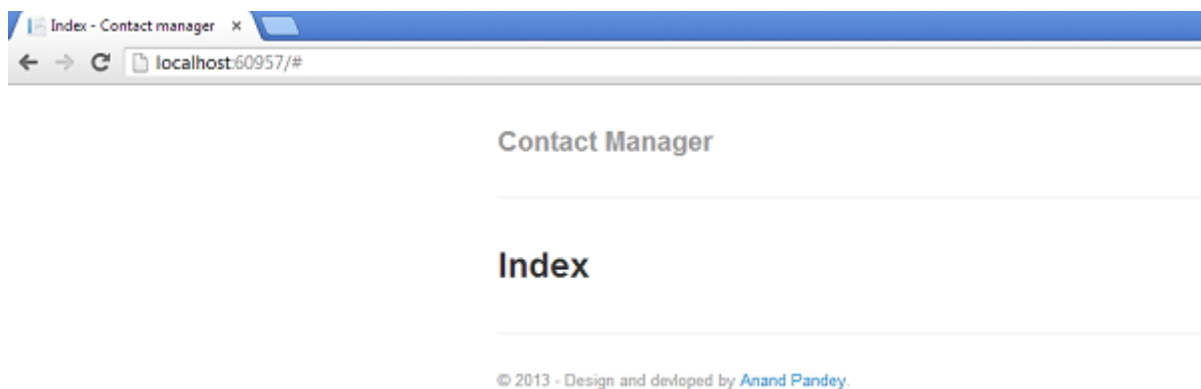
<div class="container-narrow">
  <div class="masthead">
    <ul class="nav nav-pills pull-right">

    </ul>
    <h3 class="muted">Contact Manager</h3>
  </div>
  <div id="body" class="container">
    @RenderSection("featured", required: false)
    <section>
      @RenderBody()
    </section>
  </div>
  <hr />
  <div id="footer">
    <div class="container">
      <p class="muted credit">&copy; @DateTime.Now.Year - Design and developed by <a
href="http://www.anandpandey.com">Anand Pandey</a>.</p>
    </div>
  </div>
</div>

</body>
</html>

```

Step 6: Now we are done with initial setup to run the application. The output is as below:



We will use this page to display the requirement for Screen 1 i.e. Contact List - View all contacts

Step 7: First we will create a dummy profile data as array in Contact.js (later we will fetch it from database), and then we will use this data to populate Grid.

Hide Copy Code

```

var DummyProfile = [
  {
    "ProfileId": 1,
    "FirstName": "Anand",
    "LastName": "Pandey",
    "Email": "anand@anandpandey.com"
  },
  {
    "ProfileId": 2,
    "FirstName": "John",
    "LastName": "Cena",
    "Email": "john@cena.com"
  }
]

```

Next, we will create ProfilesViewModel, a viewmodel class which hold Profiles, an array holding an initial collection of DummyProfile

data. Note that it's a `ko.observableArray`, and it is the observable equivalent of a regular array, which means it can automatically trigger UI updates whenever items are added or removed.

And finally we need to activate Knockout using `ko.applyBindings()`.

[Hide](#) [Copy Code](#)

```
var ProfilesViewModel = function () {
    var self = this;
    var refresh = function () {
        self.Profiles(DummyProfile);
    };

    // Public data properties
    self.Profiles = ko.observableArray([]);
    refresh();
};
ko.applyBindings(new ProfilesViewModel());
```

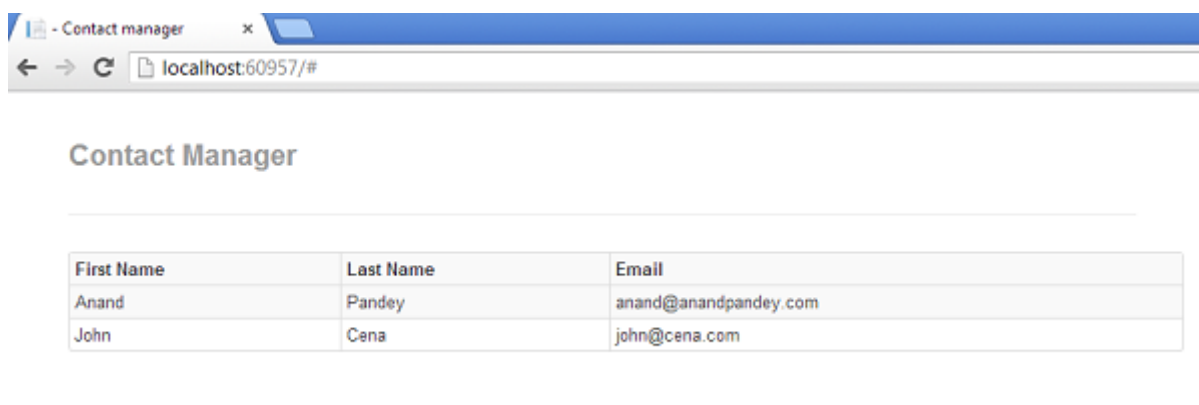
Step 8: Next we will write code in `Index.cshtml` page, that's supposed to display the Profile List. We need to use the `foreach` binding on `<tbody>` element, so that it will render a copy of its child elements for each entry in the profiles array, and then populate that `<tbody>` element with some markup to say that you want a table row (`<tr>`) for each entry.

[Hide](#) [Copy Code](#)

```
<table class="table table-striped table-bordered table-condensed">
    <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Email</th>
    </tr>
    <tbody data-bind="foreach: Profiles">
        <tr>
            <td data-bind="text: FirstName"></td>
            <td data-bind="text: LastName"></td>
            <td data-bind="text: Email"></td>
        </tr>
    </tbody>
</table>

<script src="~/Scripts/Contact.js"></script>
```

If you run the application now, you should see a simple table of profile information as:



Remember for table style we are using Bootstrap's css class. In above example it is ;

[Hide](#) [Copy Code](#)

```
<table class="table table-striped table-bordered table-condensed">
```

Step 9: Now we need to add Edit and Remove functionality for each row, and one more button at top to create a new profile. So let us do:

- Add one more <th> and <td> in our table template and bind it's click event with removeProfile function in js.
- Modify First Name row to add link for Edit Profile and then bind its click event with editProfile function.
- Add one button for Create new profile and bind it with click event using createProfile function.

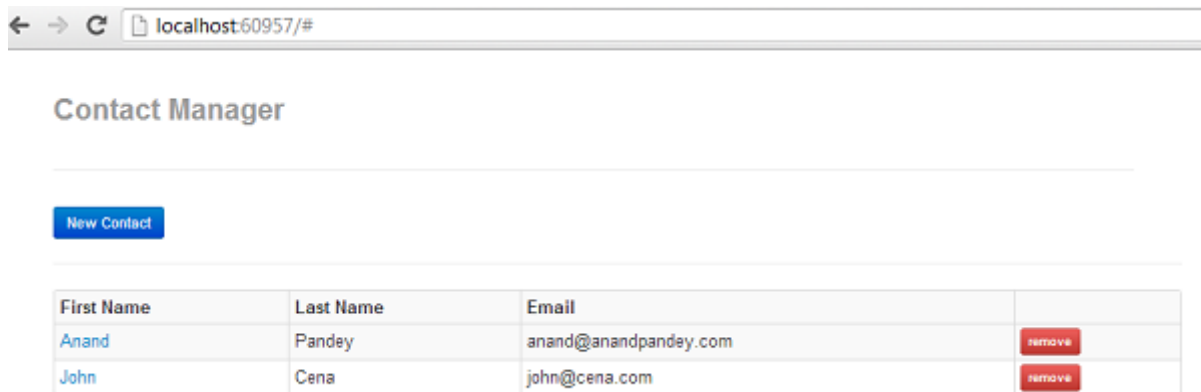
So out final code for Index.cshtml is :

[Hide](#) [Copy Code](#)

```
<input type="button" class="btn btn-small btn-primary" value="New Contact" data-
bind="click:$root.createProfile" />
<hr />
<table class="table table-striped table-bordered table-condensed">
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Email</th>
    <th></th>
  </tr>
  <tbody data-bind="foreach: Profiles">
    <tr>
      <td class="name"><a data-bind="text: FirstName, click: $parent.editProfile"></a></td>
      <td data-bind="text: LastName"></td>
      <td data-bind="text: Email"></td>
      <td><button class="btn btn-mini btn-danger" data-bind="click:
$parent.removeProfile">remove</button></td>
    </tr>
  </tbody>
</table>

<script src="~/Scripts/Contact.js"></script>
```

And output is:



None of the added button and link will work, because we have not written any code for that, so let's fix that in next step.

Step 10: Add events for createProfile, editProfile and removeProfile in Contact.js

[Hide](#) [Copy Code](#)

```
self.createProfile = function () {
  alert("Create a new profile");
};

self.editProfile = function (profile) {
  alert("Edit tis profile with profile id as :" + profile.ProfileId);
};

self.removeProfile = function (profile) {
  if (confirm("Are you sure you want to delete this profile?")) {
    self.Profiles.remove(profile);
  }
};
```

Now when we run our application and click on Remove button, then it will remove that profile from current array. As we define this array as observable, the UI will be kept in sync with changes to that array. Try it by clicking on Remove button. Edit link and Create Profile button will simply display the alert. So, let us implemented this functionality in next steps:

Step 11: Next we will add:

- A new Razor View inside Views/Contact as CreateEdit.cshtml, and register it in ContactController.cs class.

[Hide](#) [Copy Code](#)

```
public ActionResult CreateEdit()
{
    return View();
}
```

- A new js file named as CreateEdit.js inside Scripts folder.
- Modify createProfile and editProfile method of Contact.js, so that it will point to the CreateEdit page.

[Hide](#) [Copy Code](#)

```
self.createProfile = function () {
    window.location.href = '/Contact/CreateEdit/0';
};

self.editProfile = function (profile) {
    window.location.href = '/Contact/CreateEdit/' + profile.ProfileId;
};
```

Running the application will now work for all the events in Contact List (screen -1). And create and Edit should redirect to CreateEdit page with required parameters.

Step 12: First we will start with adding Profile Information to this CreateEdit page. For that we need to do:

- We need to get profileId from url so, add below two lines on top of the CreateEdit.js page

[Hide](#) [Copy Code](#)

```
var url = window.location.pathname;
var profileId = url.substring(url.lastIndexOf('/') + 1);
```

- Define a dummy Profile data as array in CreateEdit.js

[Hide](#) [Copy Code](#)

```
var DummyProfile = [
    {
        "ProfileId": 1,
        "FirstName": "Anand",
        "LastName": "Pandey",
        "Email": "anand@anandpandey.com"
    },
    {
        "ProfileId": 2,
        "FirstName": "John",
        "LastName": "Cena",
        "Email": "john@cena.com"
    }
]
```

- Profile, a simple JavaScript class constructor that stores a profile's FirstName, LastName and Email selection.

[Hide](#) [Copy Code](#)

```
var Profile = function (profile) {
    var self = this;

    self.ProfileId = ko.observable(profile ? profile.ProfileId : 0);
    self.FirstName = ko.observable(profile ? profile.FirstName : '');
    self.LastName = ko.observable(profile ? profile.LastName : '');
    self.Email = ko.observable(profile ? profile.Email : '');
};
```

- ProfileCollection, a viewmodel class that holds profile (a JavaScript object providing Profile data) along with binding for

saveProfile and backToProfileList events.

[Hide](#) [Copy Code](#)

```
var ProfileCollection = function () {
    var self = this;

    //if ProfileId is 0, It means Create new Profile
    if (profileId == 0) {
        self.profile = ko.observable(new Profile());
    }
    else {
        var currentProfile = $.grep(DummyProfile, function (e) { return e.ProfileId == profileId; });
        self.profile = ko.observable(new Profile(currentProfile[0]));
    }

    self.backToProfileList = function () { window.location.href = '/contact'; };

    self.saveProfile = function () {
        alert("Date to save is : " + JSON.stringify(ko.toJS(self.profile())));
    };
};
```

- And finally, activate Knockout using ko.applyBindings().

[Hide](#) [Copy Code](#)

```
ko.applyBindings(new ProfileCollection());
```

Step 13: Next we will write code in CreateEdit.cshtml page, that's supposed to display the Profile information. We need to use the "with" binding for profile data, so that it will render a copy of its child elements for a particular profile, and then assign the appropriate values. The code for CreateEdit.cshtml is as below:

[Hide](#) [Copy Code](#)

```
<table class="table">
    <tr>
        <th colspan="3">Profile Information</th>
    </tr>
    <tr></tr>
    <tbody data-bind='with: profile'>
        <tr>
            <td>
                <input class="input-large" data-bind='value: FirstName' placeholder="First Name"/>
            </td>
            <td>
                <input class="input-large" data-bind='value: LastName' placeholder="Last Name"/>
            </td>
            <td>
                <input class="input-large" data-bind='value: Email' placeholder="Email" />
            </td>
        </tr>
    </tbody>
</table>

<button class="btn btn-small btn-success" data-bind='click: saveProfile'>Save Profile</button>
<input class="btn btn-small btn-primary" type="button" value="Back To Profile List" data-
bind="click:$root.backToProfileList" />

<script src="~/Scripts/CreateEdit.js"></script>
```

Running the application will display the below screens :

For create New:

localhost:60957/Contact/CreateEdit/0

Contact Manager

if 0, means Create New

Profile Information

First Name Last Name Email

Save Profile Back To Profile List

For existing record with profile Id 1:

localhost:60957/Contact/CreateEdit/1

Contact Manager

> 0 means existing record

Profile Information

Anand Pandey anand@anandpandey.com

Save Profile Back To Profile List

Update any existing record and click on save with give below output:

localhost:60957/Contact/CreateEdit/1

Contact Manager

Profile Information

Anand Ranjan ranjan@anandpandey.com

Save Profile Back To Profile List

© 2013 - Design and developed by

The page at localhost:60957 says:

Date to save is : {"ProfileId": 1,"FirstName":"Anand","LastName":"Ranjan","Email":"ranjan@anandpandey.com"}

OK

As per requirement for this screen, we already have done:

- 2.1 User should be able to Enter his/her First name, Last Name and Email Address
- 2.6 Click on save button should save Contact details in Database and user will return back in Contact List page.
- 2.7 Click on Back to Profile button should return back the user to Contact List page.

In next step we will try to achieve below requirement:

- 2.2 User should able to add any number of Phone numbers by clicking on Add numbers.
- 2.3 User should able to remove any phone number

Step 14: To achieve requirement no. 2.2 and 2.3, we need to do:

- Define a dummy PhoneType and PhoneDTO data as an array in CreateEdit.js. phoneTypeData will be used to bind dropdown to

define the Phone Type for a particular Phone number.

[Hide](#) [Copy Code](#)

```
var phoneTypeData = [
    {
        "PhoneTypeId": 1,
        "Name": "Work Phone"
    },
    {
        "PhoneTypeId": 2,
        "Name": "Personal Phone"
    }
];

var PhoneDTO = [
    {
        "PhoneId": 1,
        "PhoneTypeId": 1,
        "ProfileId": 1,
        "Number": "111-222-3333"
    },
    {
        "PhoneId": 2,
        "PhoneTypeId": 2,
        "ProfileId": 1,
        "Number": "444-555-6666"
    }
];
```

- Profile, a simple JavaScript class constructor that stores a PhoneLine information which include its Phone Type i.e. whether its "Work Phone" or "Home Phone" etc. along with phone number.

[Hide](#) [Copy Code](#)

```
var PhoneLine = function (phone) {
    var self = this;
    self.PhoneId = ko.observable(phone ? phone.PhoneId : 0);
    self.PhoneTypeId = ko.observable(phone ? phone.PhoneTypeId : 0);
    self.Number = ko.observable(phone ? phone.Number : '');
};
```

- Modify ProfileCollection viewmodel class to also holds phoneNumbers along with binding for addPhone and removePhone events.

[Hide](#) [Shrink](#) [Copy Code](#)

```
var ProfileCollection = function () {
    var self = this;

    //if ProfileId is 0, It means Create new Profile
    if (profileId == 0) {
        self.profile = ko.observable(new Profile());
        self.phoneNumbers = ko.observableArray([new PhoneLine()]);
    }
    else {
        var currentProfile = $.grep(DummyProfile, function (e) { return e.ProfileId == profileId; });
        self.profile = ko.observable(new Profile(currentProfile[0]));
        var currentProfilePhone = $.grep(PhoneDTO, function (e) { return e.ProfileId == profileId; });
    }

    self.phoneNumbers = ko.observableArray(ko.utils.arrayMap(currentProfilePhone, function
(phone) {
        return phone;
    }));

    self.addPhone = function () {
        self.phoneNumbers.push(new PhoneLine());
    };

    self.removePhone = function (phone) { self.phoneNumbers.remove(phone) };
};
```



```

self.backToProfileList = function () { window.location.href = '/contact'; };

self.saveProfile = function () {
    alert("Date to save is : " + JSON.stringify(ko.toJS(self.profile())));
};
};

```

Step 15: Next we will add one more section to add Phone Information in CreateEdit.cshtml page, that's supposed to display the Phone information. As one profile can have multiple phone of different types, So, we will use the "foreach" binding for Phone numbers data, so that it will render a copy of its child elements for a particular profile, and then assign the appropriate values. Add below section in CreateEdit.cshtml just after Profile Information and before Save button.

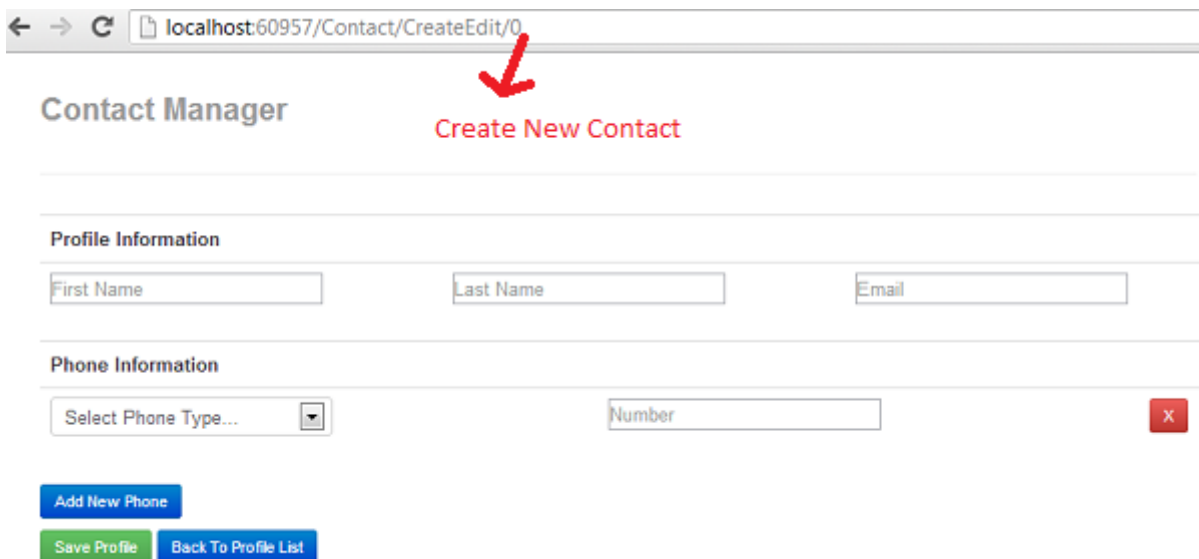
[Hide](#) [Copy Code](#)

```

<table class="table">
    <tr>
        <th colspan="3">Phone Information</th>
    </tr>
    <tr></tr>
    <tbody data-bind='foreach: phoneNumbers'>
        <tr>
            <td>
                <select data-bind="options: phoneTypeData, value: PhoneTypeId, optionsValue: 'PhoneTypeId',
optionsText: 'Name', optionsCaption: 'Select Phone Type...'></select>
            </td>
            <td>
                <input class="input-large" data-bind='value: Number' placeholder="Number" />
            </td>
            <td>
                <a class="btn btn-small btn-danger" href="#" data-bind=' click: $parent.removePhone'>X</a>
            </td>
        </tr>
    </tbody>
</table>
<p>
<button class="btn btn-small btn-primary" data-bind='click: addPhone'>Add New Phone</button>
</p>

```

Now the output to add new contact is:



And to edit existing Contact is:

localhost:60957/Contact/CreateEdit/1

Contact Manager

Edit Contact with Profile Id = 1

Profile Information

Phone Information

Work Phone X

Personal Phone X

Select Phone Type... X

Select Phone Type...
 Work Phone
 Personal Phone

Save Profile Back To Profile List

So now we left only with below requirement:

2.4 User should able to add any number of Addresses by clicking on Add new address.

2.5 User should able to remove any address

Step 16: The requirement no. 2.4 and 2.5 are similar to Phone Information, below is the final code for CreateEdit.js and CreateEdit.cshtml files:

For CreateEdit.js

Hide Shrink ▲ Copy Code

```
var url = window.location.pathname;
var profileId = url.substring(url.lastIndexOf('/') + 1);

var DummyProfile = [
  {
    "ProfileId": 1,
    "FirstName": "Anand",
    "LastName": "Pandey",
    "Email": "anand@anandpandey.com"
  },
  {
    "ProfileId": 2,
    "FirstName": "John",
    "LastName": "Cena",
    "Email": "john@cena.com"
  }
];

var PhoneTypeData = [
  {
    "PhoneTypeId": 1,
    "Name": "Work Phone"
  },
  {
    "PhoneTypeId": 2,
    "Name": "Personal Phone"
  }
];

var PhoneDTO = [
  {
```

```

        "PhoneId":1,
        "PhoneTypeId": 1,
        "ProfileId":1,
        "Number": "111-222-3333"
    },
    {
        "PhoneId": 2,
        "PhoneTypeId": 2,
        "ProfileId": 1,
        "Number": "444-555-6666"
    }
];

var AddressTypeData = [
    {
        "AddressTypeId": 1,
        "Name": "Shipping Address"
    },
    {
        "AddressTypeId": 2,
        "Name": "Billing Address"
    }
];

var AddressDTO = [
    {
        "AddressId": 1,
        "AddressTypeId": 1,
        "ProfileId": 1,
        "AddressLine1": "10000 Richmond Avenue",
        "AddressLine2": "Apt # 1000",
        "Country": "USA",
        "State": "Texas",
        "City": "Houston",
        "ZipCode": "70000"
    },
    {
        "AddressId": 2,
        "AddressTypeId": 2,
        "ProfileId": 1,
        "AddressLine1": "20000 Highway 6",
        "AddressLine2": "Suite # 2000",
        "Country": "USA",
        "State": "Texas",
        "City": "Houston",
        "ZipCode": "80000"
    }
];

var Profile = function (profile) {
    var self = this;

    self.ProfileId = ko.observable(profile ? profile.ProfileId : 0);
    self.FirstName = ko.observable(profile ? profile.FirstName : '');
    self.LastName = ko.observable(profile ? profile.LastName : '');
    self.Email = ko.observable(profile ? profile.Email : '');
    self.PhoneDTO = ko.observableArray(profile ? profile.PhoneDTO : []);
    self.AddressDTO = ko.observableArray(profile ? profile.AddressDTO : []);
};

var PhoneLine = function (phone) {
    var self = this;
    self.PhoneId = ko.observable(phone ? phone.PhoneId : 0);
    self.PhoneTypeId = ko.observable(phone ? phone.PhoneTypeId : 0);
    self.Number = ko.observable(phone ? phone.Number : '');
};

var AddressLine = function (address) {

```

```

var self = this;
self.AddressId = ko.observable(address ? address.AddressId : 0);
self.AddressTypeId = ko.observable(address ? address.AddressTypeId : 0);
self.AddressLine1 = ko.observable(address ? address.AddressLine1 : '');
self.AddressLine2 = ko.observable(address ? address.AddressLine2 : '');
self.Country = ko.observable(address ? address.Country : '');
self.State = ko.observable(address ? address.State : '');
self.City = ko.observable(address ? address.City : '');
self.ZipCode = ko.observable(address ? address.ZipCode : '');
};

var ProfileCollection = function () {
    var self = this;

    //if ProfileId is 0, It means Create new Profile
    if (profileId == 0) {
        self.profile = ko.observable(new Profile());
        self.phoneNumbers = ko.observableArray([new PhoneLine()]);
        self.addresses = ko.observableArray([new AddressLine()]);
    }
    else {
        //For Profile information
        var currentProfile = $.grep(DummyProfile, function (e) { return e.ProfileId == profileId; });
        self.profile = ko.observable(new Profile(currentProfile[0]));
        //For Phone number
        var currentProfilePhone = $.grep(PhoneDTO, function (e) { return e.ProfileId == profileId; });
        self.phoneNumbers = ko.observableArray(ko.utils.arrayMap(currentProfilePhone, function (phone) {
            return phone;
        }));
        //For Address
        var currentProfileAddress = $.grep(AddressDTO, function (e) { return e.ProfileId == profileId; });
        self.addresses = ko.observableArray(ko.utils.arrayMap(currentProfileAddress, function (address) {
            return address;
        }));
    }

    self.addPhone = function () { self.phoneNumbers.push(new PhoneLine()) };

    self.removePhone = function (phone) { self.phoneNumbers.remove(phone) };

    self.addAddress = function () { self.addresses.push(new AddressLine()) };

    self.removeAddress = function (address) { self.addresses.remove(address) };

    self.backToProfileList = function () { window.location.href = '/contact'; };

    self.saveProfile = function () {
        self.profile().AddressDTO = self.addresses;
        self.profile().PhoneDTO = self.phoneNumbers;
        alert("Date to save is : " + JSON.stringify(ko.toJS(self.profile())));
    };
};

ko.applyBindings(new ProfileCollection());

```

and, for CreateEdit.cshtml

Hide Shrink ▲ Copy Code

```

<table class="table">
    <tr>
        <th colspan="3">Profile Information</th>
    </tr>
    <tr></tr>
    <tbody data-bind='with: profile'>
        <tr>
            <td>
                <input class="input-large" data-bind='value: FirstName' placeholder="First Name"/>
            </td>
        </tr>
    </tbody>
</table>

```

```

        </td>
        <td>
            <input class="input-large" data-bind='value: LastName' placeholder="Last Name"/>
        </td>
        <td>
            <input class="input-large" data-bind='value: Email' placeholder="Email" />
        </td>
    </tr>
</tbody>
</table>

<table class="table">
    <tr>
        <th colspan="3">Phone Information</th>
    </tr>
    <tr></tr>
    <tbody data-bind='foreach: phoneNumbers'>
        <tr>
            <td>
                <select data-bind="options: PhoneTypeData, value: PhoneTypeId, optionsValue: 'PhoneTypeId',
optionsText: 'Name', optionsCaption: 'Select Phone Type...'"></select>
            </td>
            <td>
                <input class="input-large" data-bind='value: Number' placeholder="Number" />
            </td>
            <td>
                <a class="btn btn-small btn-danger" href='#' data-bind=' click: $parent.removePhone'>X</a>
            </td>
        </tr>
    </tbody>
</table>
<p>
<button class="btn btn-small btn-primary" data-bind='click: addPhone'>Add New Phone</button>
</p>
<hr />
<table class="table">
    <tr><th colspan="5">Address Information</th></tr>
    <tbody data-bind="foreach: addresses">
        <tr>
            <td colspan="5">
                <select data-bind="options: AddressTypeData, value: AddressTypeId, optionsValue:
'AddressTypeId', optionsText: 'Name', optionsCaption: 'Select Address Type...'"></select>
            </td>
        </tr>
        <tr>
            <td>
                <input class="input-large" data-bind='value: AddressLine1' placeholder="Address Line1" />
                <p style="padding-top: 5px;"><input class="input-large" data-bind='value: State'
placeholder="State" /></p>
            </td>
            <td>
                <input class="input-large" data-bind=' value: AddressLine2' placeholder="Address Line2" />
                <p style="padding-top: 5px;"><input class="input-large" data-bind='value: Country'
placeholder="Country" /></p>
            </td>
            <td>
                <input class="input-large" data-bind='value: City' placeholder="City" />
                <p style="padding-top: 5px;"><input class="input-large" data-bind='value: ZipCode'
placeholder="Zip Code" />
                <a class="btn btn-small btn-danger" href='#' data-bind='click: $root.removeAddress'>X</a>
            </td>
        </tr>
    </tbody>
</table>
<p>
<button class="btn btn-small btn-primary" data-bind='click: addAddress'>Add New Address</button>
</p>
<hr />

```

```
<button class="btn btn-small btn-success" data-bind='click: saveProfile'>Save Profile</button>
<input class="btn btn-small btn-primary" type="button" value="Back To Profile List" data-
bind="click:$root.backToProfileList" />

<script src="~/Scripts/CreateEdit.js"></script>
```

So finally, Application will display screens as per the requirement:

Screen 1: Contact List - View all contacts

The screenshot shows a web browser at `localhost:60957/contact`. The page is titled "Contact Manager". There is a "New Contact" button. Below it is a table with columns: First Name, Last Name, Email, and a remove button.

First Name	Last Name	Email	
Anand	Pandey	anand@anandpandey.com	remove
John	Cena	john@cena.com	remove

Screen 2: Create New Contact - This screen should display one blank screen to provide functionality as.

The screenshot shows a web browser at `localhost:60957/Contact/CreateEdit/0`. The page is titled "Contact Manager". The form is divided into sections: Profile Information, Phone Information, and Address Information.

Profile Information

First Name: Last Name: Email:

Phone Information

Select Phone Type... Number: X

Add New Phone

Address Information

Select Address Type...

Address Line1: Address Line2: City:

State: Country: Zip Code: X

Add New Address

Save Profile **Back To Profile List**

Screen 3: Update Existing Contact - This screen should display screen with selected contact information details.

← → ↻

Contact Manager

Profile Information

Phone Information

Work Phone

Personal Phone

Address Information

Shipping Address

Billing Address

Validation:

We are almost done with designing part of our application .The only thing left now, is to manage validation when the user clicks on "Save" button. Validation is the major requirement and now a day's most ignorant part for any web application. By proper validation, user can know what data needs to be entered. Next in this article, I am going to discuss KnockoutJS Validation library which can be downloaded using NuGet. Let us check some of the most common validation scenarios, and how to achieve it using knockout validation.

Scenario 1: First Name is a required field in form

Hide Copy Code

```
this.FirstName = ko.observable().extend({ required: true });
```

Scenario 2: Maximum number of character for First Name should not exceed 50 and should not be less than 3 character

Hide Copy Code

```
this.FirstName = ko.observable().extend({ maxLength: 50, minLength: 3 });
```

Scenario 3: First Name is a required field in form and maximum number of character for First Name should not exceed 50 and should not be less than 3 character.

[Hide](#) [Copy Code](#)

<p>Scenario 4: Age is a required field in form, and should be always greater than 18 and less than 100</p>

Scenario 5: Email is a required field and should be in proper email format

[Hide](#) [Copy Code](#)

```
this.Email = ko.observable().extend({ required: true, email: true });
```

Scenario 6: Date of Birth is a required field and should be in proper date format

[Hide](#) [Copy Code](#)

```
this.DateOfBirth = ko.observable().extend({ required: true, date: true });
```

Scenario 7: Price is a required field and should be in proper number or decimal format

[Hide](#) [Copy Code](#)

```
this.Price = ko.observable().extend({ required: true, number: true });
```

Scenario 8: Phone Number is a required field and should be in proper US format

Note: A valid US phone number has the following format: 1-xdd-xdd-dddd

The "1-" at the beginning of the string is optional and so are the dashes. x is any digit between 2 and 9 while d can be any digit.

[Hide](#) [Copy Code](#)

```
this.Phone = ko.observable().extend({ required: true, phoneUS: true });
```

Scenario 9: ToDate field must be greater than FromDate field

[Hide](#) [Copy Code](#)

```
this.ToDate = ko.observable().extend({
    equal: function () { return (val > $('#FromDate').val()) ? val : val + "|" }
});
```

Scenario 9: Phone number should accept only +- () 0-9 from users

[Hide](#) [Copy Code](#)

```
var regex = /\(?([0-9]{3})\)?([ .-]?)([0-9]{3})\2([0-9]{4})/
this.PhoneNumber = ko.observable().extend({ pattern: regex });
```

So, far we have seen different type of validation scenarios and their syntax; now let us implement it in our application. For that first download the library knockout.validation.js using NuGet. Right now our validation script is fully completed and should look like this :

[Hide](#) [Shrink](#) [Copy Code](#)

```
var Profile = function (profile) {
    var self = this;
    self.ProfileId = ko.observable(profile ? profile.ProfileId : 0).extend({ required: true });
    self.FirstName = ko.observable(profile ? profile.FirstName : '').extend({ required: true, maxLength: 50 });
    self.LastName = ko.observable(profile ? profile.LastName : '').extend({ required: true, maxLength: 50 });
    self.Email = ko.observable(profile ? profile.Email : '').extend({ required: true, maxLength: 50, email: true });
    self.PhoneDTO = ko.observableArray(profile ? profile.PhoneDTO : []);
    self.AddressDTO = ko.observableArray(profile ? profile.AddressDTO : []);
};

var PhoneLine = function (phone) {
    var self = this;
    self.PhoneId = ko.observable(phone ? phone.PhoneId : 0);
    self.PhoneTypeId = ko.observable(phone ? phone.PhoneTypeId : undefined).extend({ required: true });
    self.Number = ko.observable(phone ? phone.Number : '').extend({ required: true, maxLength: 25, phoneUS: true });
};
```



```

};

var AddressLine = function (address) {
    var self = this;
    self.AddressId = ko.observable(address ? address.AddressId : 0);
    self.AddressTypeId = ko.observable(address ? address.AddressTypeId : undefined).extend({ required: true });
};
self.AddressLine1 = ko.observable(address ? address.AddressLine1 : '').extend({ required: true,
maxLength: 100 });
self.AddressLine2 = ko.observable(address ? address.AddressLine2 : '').extend({ required: true,
maxLength: 100 });
self.Country = ko.observable(address ? address.Country : '').extend({ required: true, maxLength: 50 });
self.State = ko.observable(address ? address.State : '').extend({ required: true, maxLength: 50 });
self.City = ko.observable(address ? address.City : '').extend({ required: true, maxLength: 50 });
self.ZipCode = ko.observable(address ? address.ZipCode : '').extend({ required: true, maxLength: 15 });
};

```

After validation our final solution looks like below screen after click on "Save" button:

The screenshot shows a web browser at the URL `localhost:60957/Contact/CreateEdit/0`. The page is titled "Contact Manager" and contains three main sections: Profile Information, Phone Information, and Address Information.

Profile Information: Contains fields for First Name, Last Name, and Email. The First Name and Last Name fields have a red message "This field is required". The Email field has a red message "Please enter a proper email address".

Phone Information: Contains a dropdown for Work Phone and a text field for the phone number. The phone number field has a red message "Please specify a valid phone number". There is a red "X" icon next to the message.

Address Information: Contains a dropdown for Select Address Type... with a red message "This field is required". Below it are fields for Address Line1, Address Line2, City, State, and Country. Address Line1, Address Line2, City, State, and Country all have a red message "This field is required". There is a red "X" icon next to the City field.

At the bottom of the form, there are two buttons: "Add New Phone" and "Add New Address". At the very bottom, there are two buttons: "Save Profile" and "Back To Profile List".

Conclusion:

We've done it. Well, you have done it! I hope you enjoyed this tutorial and learned something.

In this article, we talked about achieving our UI without knowing any actual implementation (databases interaction) i.e. UI is created independently by any designer/developer without knowing the actual business logic. !!!That's great!!!

In [next part](#) of this article, I will talk about how to design database and how to implement business logic using structured layers.

If you have any questions feel free to ask; I will be glad to talk more in the comments. Thanks for your time!

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOl\)](#)

Share



About the Author



Anand Ranjan Pandey

  Software Developer (Senior)
United States 

No Biography provided

You may also be interested in...

[Design and Develop a website using ASP.NET MVC 4, EF, Knockoutjs and Bootstrap : Part - 2](#)

[Microsoft's Guide to Modern Dev/Test](#)

[Developing, Architecting and Testing Web Applications with MVC 5, Web API 2, KnockoutJS, Ninject and NUnit](#)

[F12 Edge Developer Tools](#)

[KnockoutJS vs. Silverlight](#)

[Strong Authentication and the Road to FIDO](#)

Comments and Discussions

You must [Sign In](#) to use this message board.

Search Comments

Go

[First](#) [Prev](#) [Next](#)

Could u please explain how to retriive images from the database

and show in the grid view in mvc4 by using aspx engein

[Sign In](#) · [View Thread](#) · [Permalink](#)

Great Article 5 out of 5 or if I was a Spinal Tap member 11

This was easy to follow and use and gave you a great example of how to use a combo of JS restful/ restless services in conjunction with .net architecture! I had some hiccups but mostly my own errors, but all in all great and easy to follow!

[Sign In](#) · [View Thread](#) · [Permalink](#)

ProfilesViewModel

In which directory does ProfilesViewModel reside and where does the followung code belong?

[Hide](#) [Copy Code](#)

```
var ProfilesViewModel = function () {  
    var self = this;  
    var refresh = function () {  
        self.Profiles(DummyProfile);  
    };  
  
    // Public data properties  
    self.Profiles = ko.observableArray([]);  
    refresh();  
};  
ko.applyBindings(new ProfilesViewModel());
```

[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: ProfilesViewModel

paste it in content.js just below to array intaialization



[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 5

Excellent Article!

[Sign In](#) · [View Thread](#) · [Permalink](#)

Java Script Not Triggering

Hi Anand,


Nice and cool tutorial. Enjoyed it alot till the end. But for me if I Click on the New Contact button or remove the Javascript is not triggering. Since I am new to MVC and Knockout.js cant able to sort it out. Could you please help me out on this???

Thanks in Advance

Suresh


[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: Java Script Not Triggering

You need to put the create/remove/edit functionality into ProfilesViewModel() (below the refresh function). 

[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: Java Script Not Triggering

May be your button is firing the required event, please double check it. 
Hope that helps

[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: Java Script Not Triggering

Hello! 

Try to use Developer Mode disabling cache and reloading the page in order to check if the Script reference is really Working. If your using Firefox, you should disable cache in order to check this kind of things.

Best Regards,
Jeoxs

[Sign In](#) · [View Thread](#) · [Permalink](#)

Step 13

Hi I followed up to step 13, when I click on to the link in the list it goes to createEdit page but the contact info is not passing through and when I click on to back to profile list button it does not do anything.

When I run the application from the your source code it works fine. I tried to copy the code from your file and still the same error in the application I worked. Can you help?

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

Step 10

I followed your instructions through step 10 but after adding the code you provided, the clicking of the buttons did nothing. Please respond!!!

[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: Step 10

Hello!



Try disabling cache and reloading the page in Developer Mode in order to get sure that the script's reference is working

[Sign In](#) · [View Thread](#) · [Permalink](#)

Step 7

Every step works great till step 7. In step 7 how to create ProfilesViewModel, where to add below mentioned code .

[Hide](#) [Copy Code](#)

```
var ProfilesViewModel = function () {
    var self = this;
    var refresh = function () {
        self.Profiles(DummyProfile);
    };

    // Public data properties
    self.Profiles = ko.observableArray([]);
    refresh();
};
ko.applyBindings(new ProfilesViewModel());
```

[Sign In](#) · [View Thread](#) · [Permalink](#)

Database process

How to Complete This process with DataBase.....

[Sign In](#) · [View Thread](#) · [Permalink](#)

1.00/5 (1 vote)

NavigationRouteFilterExamples

Sir how can i insert navigationroutefilterexamples in the project? because this file is causing an error .

[Sign In](#) · [View Thread](#) · [Permalink](#)

Awesome Article

It give practical knowledge.

[Sign In](#) · [View Thread](#) · [Permalink](#)

step 10

after i am adding the events in step 10 to the Contact.js

i am trying to click on remove button but nothing happen, any idea why?

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

Re: step 10

Did you ever get a response? I made it up through step ten and clicking the buttons did nothing for me either.



[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 5

One of best article for MVC4, EF, Knockoutjs. Thanks Anand for sharing.

[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 5

Very nice article, like the way your covered every details step by step.

[Sign In](#) · [View Thread](#) · [Permalink](#)

About this articale

I am new in asp.net.But how i will create profileViewmodel in visual studio 2012. Please help me

[Sign In](#) · [View Thread](#) · [Permalink](#)

1.00/5 (1 vote)

Validations on DTO or Entity

Dear Anand,

I understand that you have given the validations on entities. Is there any specific reason for this? How about giving validations on DTO?

Please clarify.

Thanks,
Dhananjay

[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 5

Excelletn

[Sign In](#) · [View Thread](#) · [Permalink](#)

Didn't discuss about bundling of bootstrap

The article was good ,you discussed about knockout bundling in bundle.config file but you missed bootstrap bundling. I would like to know how to bundle and render it.

thank you.

[Sign In](#) · [View Thread](#) · [Permalink](#)

1.00/5 (1 vote)

Save data to database

Thanks 🙌

ME store those data in database (Sql server).

Please help 😊

Give me some link Or Some Code

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

Re: Save data to database

There is a 2nd Part of this article.

Please follow the following link

[^]

You can also get the whole article of codeplex.com too.

But 2nd article is not not easy to follow (at least for me).

Zia

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

Re: Save data to database

ok Done

Thanks

Try Zia....try & try will be Successssssssss 🙌

-- modified 30-Sep-13 7:51am.

[Sign In](#) · [View Thread](#) · [Permalink](#)

Step 6: does not work well

Dear,

I followed the example step by step through step 6. to run the application, it shows the page without styles, without the appearance of bootstrap. Not if I miss to add something to the project, I hope I can help.
regards

PD: I speak Spanish, sorry for the translation.

ATTE.

Mario Cortés

[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: Step 6: does not work well

I am facing the same issue as well. All that I'm getting is a blank screen. Am I missing something?



[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 5

good tutorial. I learn alot from this artical. Thanks.

[Sign In](#) · [View Thread](#) · [Permalink](#)

سعدی همتی

thank you so much.this is the best sample aap i have seen

[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 5

Best example I've seen on the subject of Knockout and MVC

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

Validation of newly added objects

Hi

First of all, great tutorial!

I would like to point out that in the validation part of this tutorial you are adding validations: Validation wont trigger on newly added items.

To reproduce the bug:

Create new profile

Add new Address

Press Create Profile

The validation wont trigger on the newly added address fields.

To remedy this error I added subscriptions to the phonenumbers and addresses fields, declared them right after I declared self.phoneErrors and self.addressErrors like so:

[Hide](#) [Copy Code](#)

```
self.phoneErrors = ko.validation.group(self.phoneNumbers(), { deep: true });
self.addressErrors = ko.validation.group(self.addresses(), { deep: true });

self.phoneNumbers.subscribe(function () {
    self.phoneErrors = ko.validation.group(self.phoneNumbers(), { deep: true });
});
self.addresses.subscribe(function () {
    self.addressErrors = ko.validation.group(self.addresses(), { deep: true });
});
```


Please correct me if I'm doing it the wrong way or if there is a better way of achieving the same result 😊

cheers,
Gísli

[Sign In](#) · [View Thread](#) · [Permalink](#)

Excellent

I'm planning to start my journey through asp.net MVC follow your acticle.
from China.

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

My vote of 5

Excellent, very different to what Ive found, this is very very interesting to try.

Thanks for the good work. From Honduras

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

My vote of 5

This is an excellent demonstration of MVC excellence

[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 5

Excellent. It's clean, concise, and contains valuable pieces of information and not just some "Hello world" introduction like we often see. I mean this is serious stuff. Thank's for sharing.

[Sign In](#) · [View Thread](#) · [Permalink](#)

Create first Contact/Index

Do not forget to reference Layout = "~/Views/Shared/_Layout.cshtml"

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (3 votes)

My vote of 5

One word: Excellent!

[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote of 4

I have searched article like this and i found one on this web, it's awesome man !

[Sign In](#) · [View Thread](#) · [Permalink](#)

Complete project missing

Hi I downloaded the application to have a look at the various projects and the download only contains the application.

Can you update this to have all the various projects so people look at the actual code to have a play / investigate what's in each area ?

[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: Complete project missing

Please check next part of this article :

[Design and Develop a website using ASP.NET MVC 4, EF, Knockoutjs and Bootstrap : Part - 2\[^\]](#)

Cheers by,

Anand Ranjan

[Sign In](#) · [View Thread](#) · [Permalink](#)

Search engines

Hi Anand,

Coming from WPF/SL I'm really thrilled about the way MVVM is being implemented by ko and the way you've integrated it with MVC4.

My only concern remains SEO. Are search engines nowadays capable of interpreting the ajax calls and ko logic?

Or have do you have a fallback solution available for your approach?

Thank you

[Sign In](#) · [View Thread](#) · [Permalink](#)

Re: Search engines

By following certain steps you can achieve it, Please refer below links:

<https://developers.google.com/webmasters/ajax-crawling/docs/getting-started>

<http://www.webbizideas.com/images/SEO-Friendly-Design-Guidelines.pdf>

Cheers by,

Anand Ranjan

[Sign In](#) · [View Thread](#) · [Permalink](#)

My vote 5+ How to save and bind data with MVC model rather dummy

I must say, this is excellent and easy to understand.

I got this article after 5 days struggling with knockout.js

if you could just explain:

1. How to bind data with MVC model rather dummy model
 2. and how to save the changed model.
 3. how to edit contact inline using partial view
- e.g. clicking on contact will open edit view just below that row.

I can do it using partial view but not sure how to implement it here as in a different view we are passing the contact id. apology for such stupid question 😞

Answer :

1. Passing viewmodel from controller and use below line to fetch it in .js file
- ```
var viewModel = $.parseJSON('@Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(Model)))');
```

Regards,

*modified 24-Apr-13 5:04am.*

[Sign In](#) · [View Thread](#) · [Permalink](#)

---

## where is style masthead defined?

where is style masthead defined? Used in \_Layout.cshtml ??? 😞

[Sign In](#) · [View Thread](#) · [Permalink](#)

---

## My vote of 5

Great article!!

[Sign In](#) · [View Thread](#) · [Permalink](#)

---

## Great article - small clarification

"Finally modify the default map route in Route.config to point to Contact controller as:"  
should that be RouteConfig.cs instead of Route.config?

Tony

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

---

## My vote of 5

Excellent

[Sign In](#) · [View Thread](#) · [Permalink](#)

5.00/5 (1 vote)

---

## My vote of 5

good one, waiting on next parts to see what you do with top 6 layers.  
Application.web already got 3 layers m,v and c. wondering how will you put 6 layers on top of that without repeating the tasks.

thanks

[Sign In](#) · [View Thread](#) · [Permalink](#)


Last Visit: 31-Dec-99 18:00   Last Update: 5-Jul-16 14:32

[Refresh](#)

[1](#) [2](#) [3](#) [Next »](#)

 [General](#)  [News](#)  [Suggestion](#)  [Question](#)  [Bug](#)  [Answer](#)  [Joke](#)  [Praise](#)  [Rant](#)  [Admin](#)

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | [Mobile](#)  
Web02 | 2.8.160621.1 | Last Updated 16 Jan 2013

 Select Language | ▼  
Layout: [fixed](#) | [fluid](#)

Article Copyright 2013 by Anand Ranjan Pandey  
Everything else Copyright © [CodeProject](#), 1999-2016