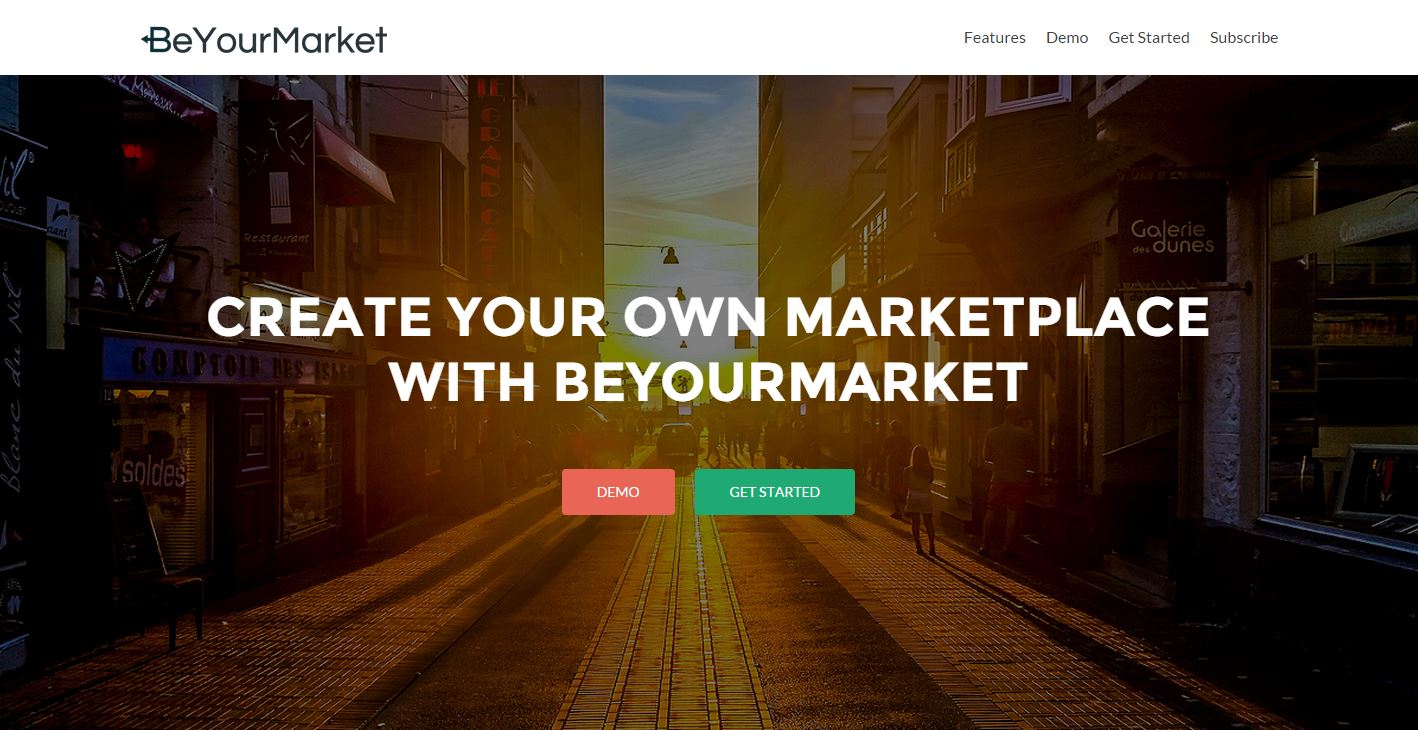
**[BeYourMarket](https://beyourmarket.atlassian.net/wiki/display/BYM/BeYourMarket)**

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/BeYourMarket#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jun 23, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491533&selectedPageVersions=6&selectedPageVersions=7)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/BeYourMarket#page-metadata-start)

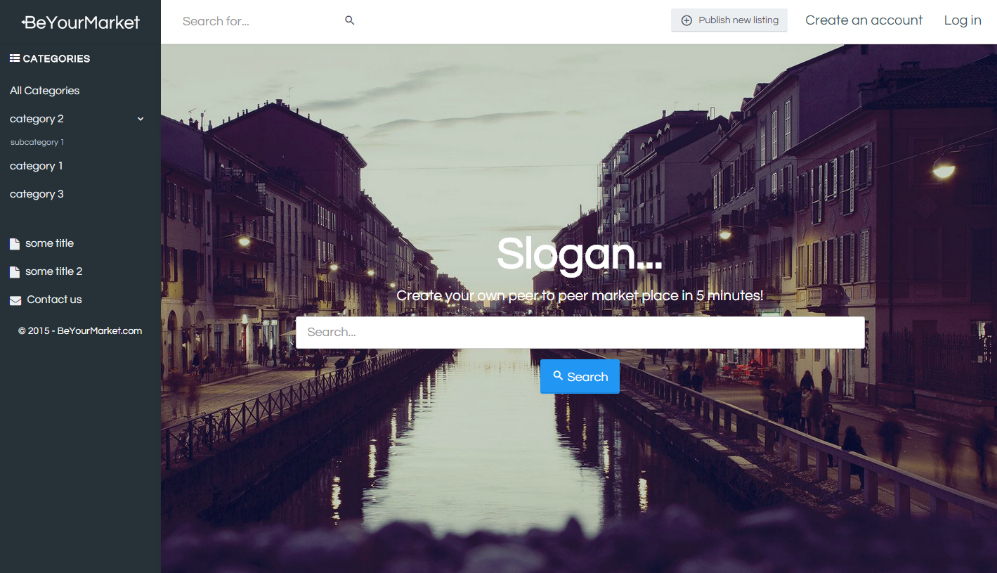
[BeYourMarket](http://beyourmarket.com) is an opensource marketplace framework based on [asp.net](http://asp.net) and mvc 5. It enables you to create and customize a marketplace quickly and easily!

[](http://beyourmarket.com/)

BeYourMarket is an opensource marketplace framework based on [asp.net](http://asp.net) and mvc 5. It enables you to create and customize a marketplace quickly and easily!

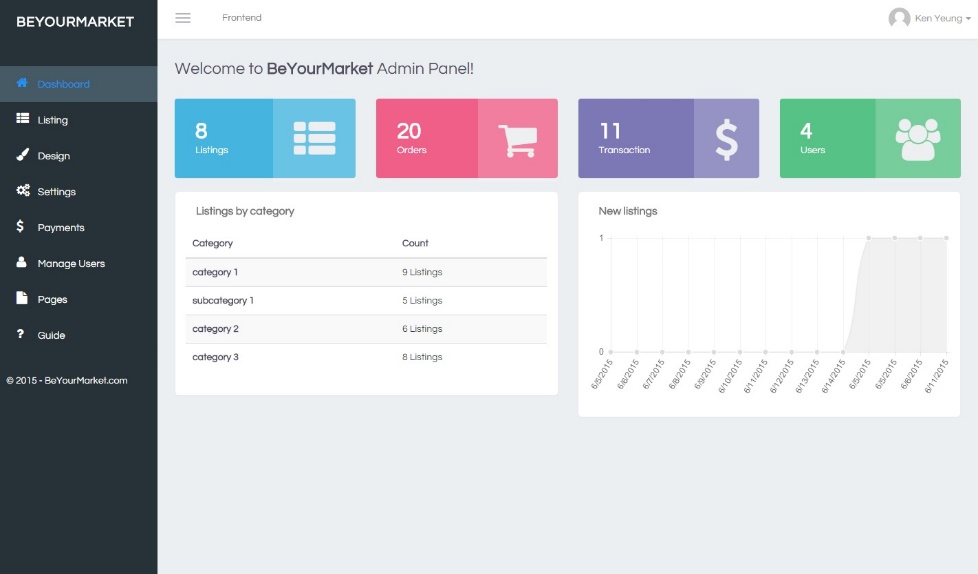
**A remarkably simple user interface**

BeYourMarket has been designed to make you as productive as possible. It allows your to focus on building up your community rather than how the technology works.



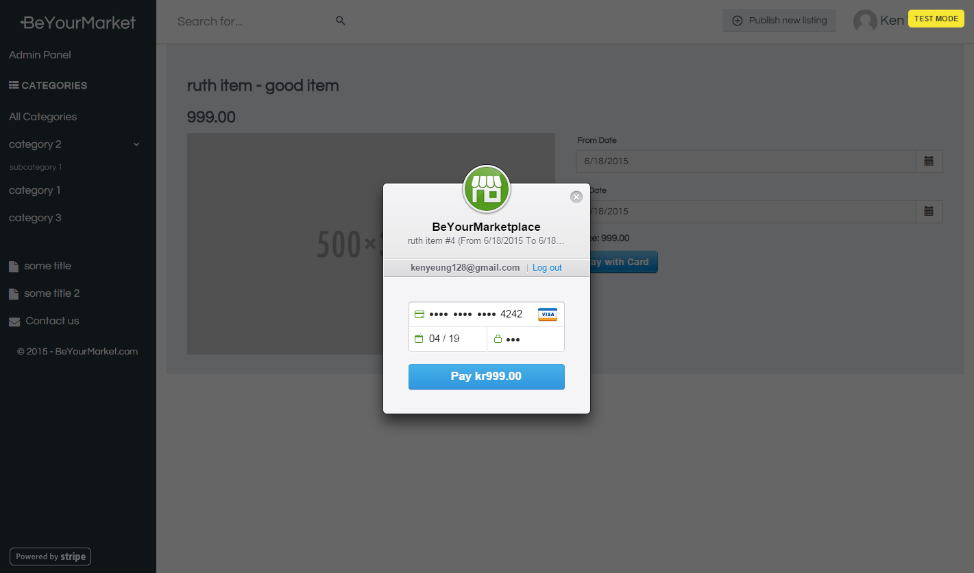
**Easy marketplace management**

Powerful dashboard management panel allow you to manage users, categories and your services/products in your marketplace.



**Pay and accept payments easily**

Stripe is integrated into BeYourMarket so users from your community can accept and receive payment easily and securely.



[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Running+BeYourMarket+with+Web+Platform#page-metadata-end)

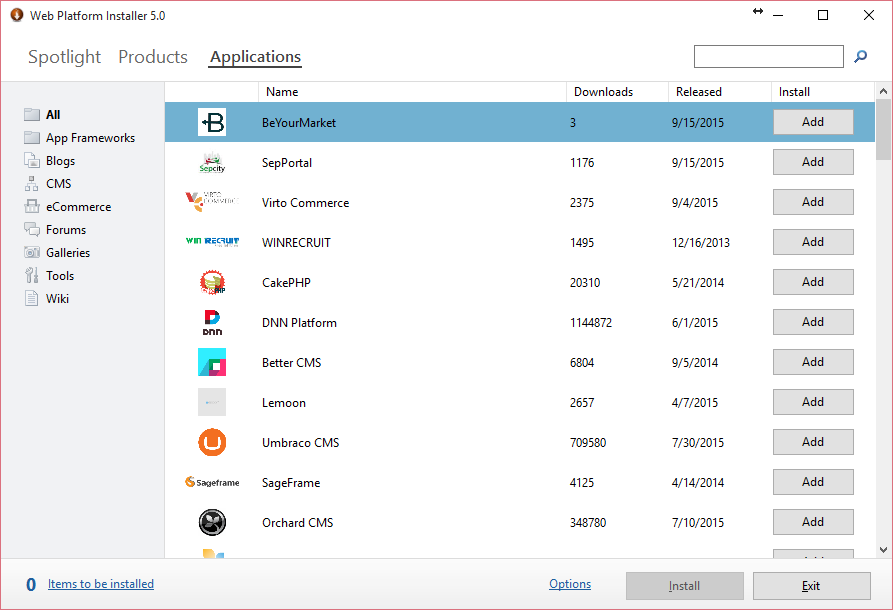
* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin) on [Sep 17, 2015](https://beyourmarket.atlassian.net/wiki/pages/viewpreviousversions.action?pageId=3538953)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Running+BeYourMarket+with+Web+Platform#page-metadata-start)

Install and run BeYourMarket on your PC. Create a new marketplace in 5 minutes!

Download [Web Platform Installer](http://www.microsoft.com/web/downloads/platform.aspx)

Search "BeYourMarket", Click install and follow the instruction.



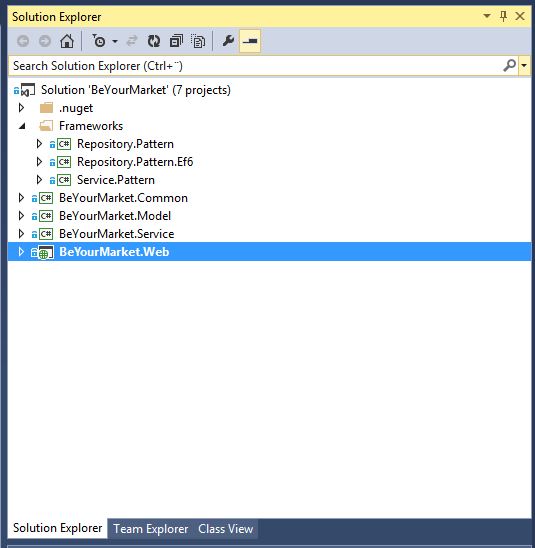
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Running+BeYourMarket+with+Visual+Studio+2013#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jun 30, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491546&selectedPageVersions=6&selectedPageVersions=7)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Running+BeYourMarket+with+Visual+Studio+2013#page-metadata-start)

Install and run BeYourMarket on your PC. Create a new marketplace in 5 minutes!

## Running BeYourMarket with Visual Studio 2013

1. Get source with git clone (See [Running BeYourMarket with Visual Studio 2013](https://beyourmarket.atlassian.net/wiki/display/BYM/Running+BeYourMarket+with+Visual+Studio+2013))
2. Compile the solution
3. Set the project **BeYourMarket.Web** as startup solution  
   
4. Run **BeYourMarket.Web**
5. An installation wizard will be launch as below, choose admin username/password, select database type, click Install button  
     
   
6. When it finished, it will be redirected to the home page of your new marketplace!

To launch the installation wizard again, simply remove the Default connection string in connectionstrings.config

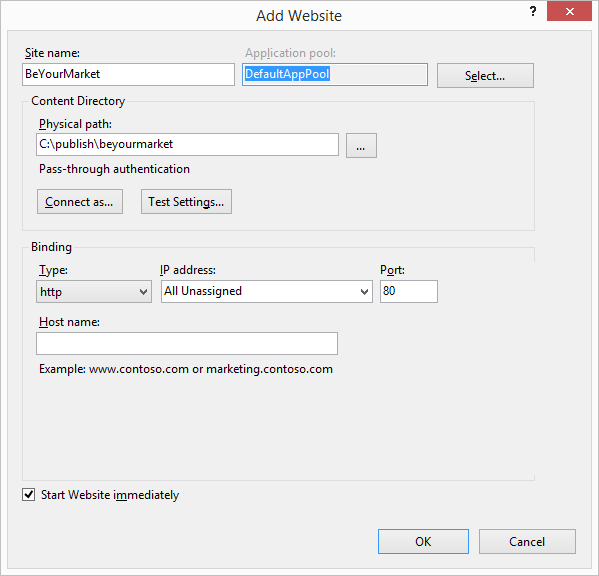
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Running+BeYourMarket+with+IIS#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jun 30, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1179651&selectedPageVersions=1&selectedPageVersions=2)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Running+BeYourMarket+with+IIS#page-metadata-start)

Install and run BeYourMarket on your PC. Create a new marketplace in 5 minutes!

## Running BeYourMarket with IIS

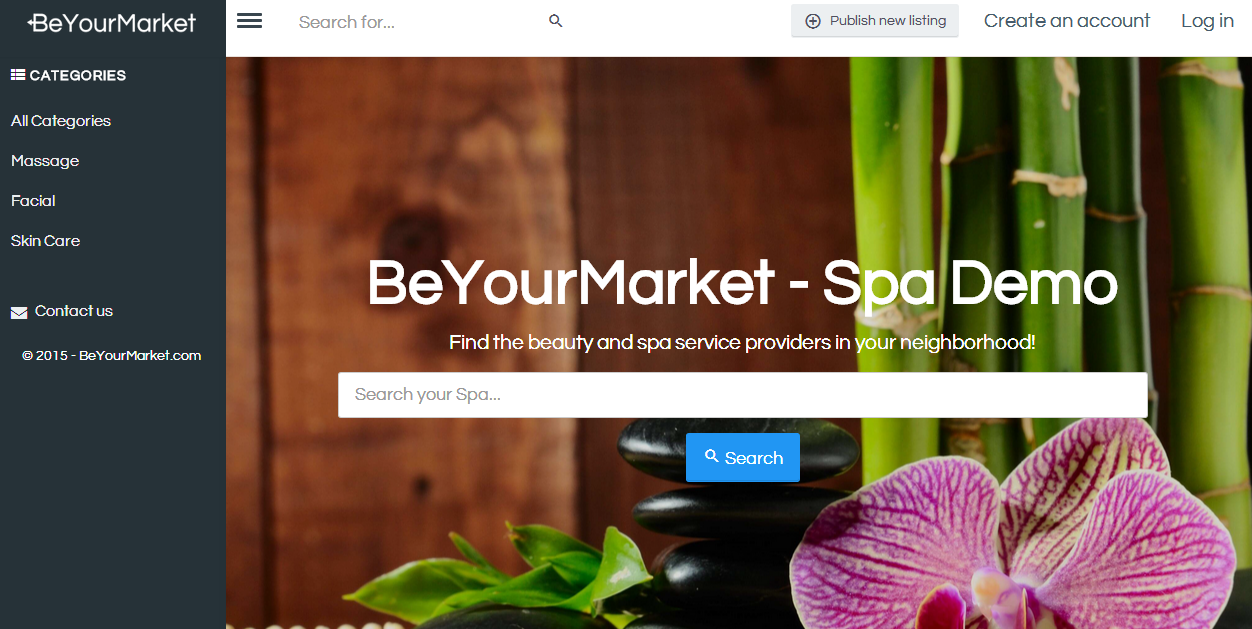
1. Publish the solution as File System using Visual Studio 2013
2. Install Application Request Routing, for URL rewrite.  
   <http://www.iis.net/downloads/microsoft/application-request-routing>
3. Open IIS and add new application, choose the Physical Path as the published solution.  
     
   
4. Open browser and run the Marketplace with setup first time!

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Intro+of+the+Frontend#page-metadata-end)

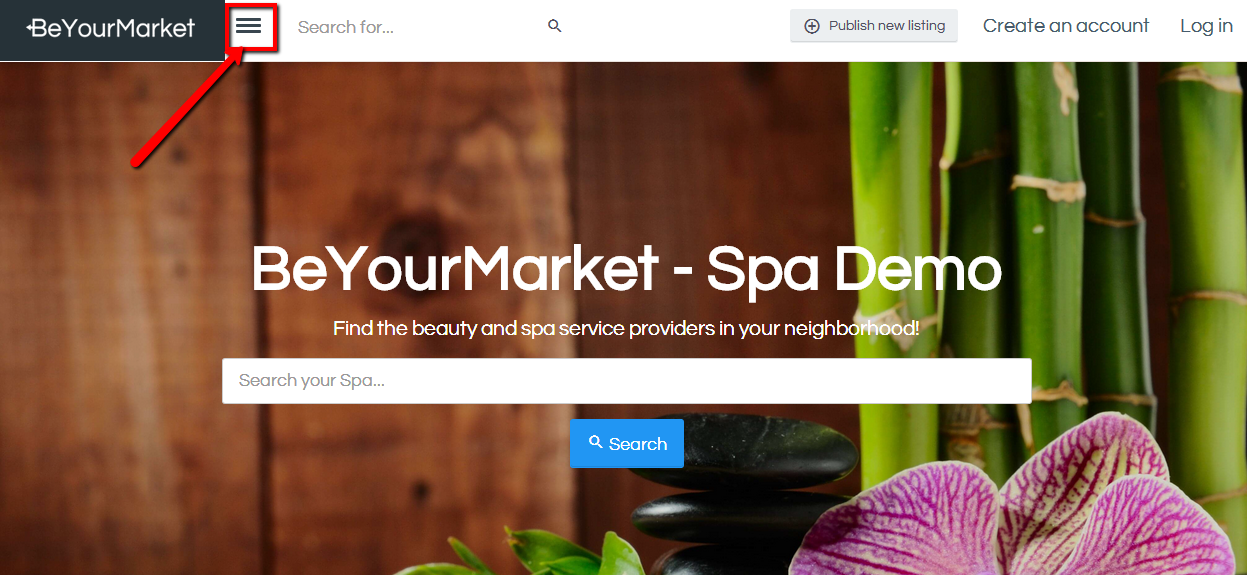
* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jun 23, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491593&selectedPageVersions=8&selectedPageVersions=9)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Intro+of+the+Frontend#page-metadata-start)

In BeYourMarket front end, you will be able to view and navigate the created marketplace.



* + [User Dashboard](https://beyourmarket.atlassian.net/wiki/display/BYM/User+Dashboard)
  + [Create Listing](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Listing)
  + [Configure Payment](https://beyourmarket.atlassian.net/wiki/display/BYM/Configure+Payment)
  + [Accept orders](https://beyourmarket.atlassian.net/wiki/display/BYM/Accept+orders)

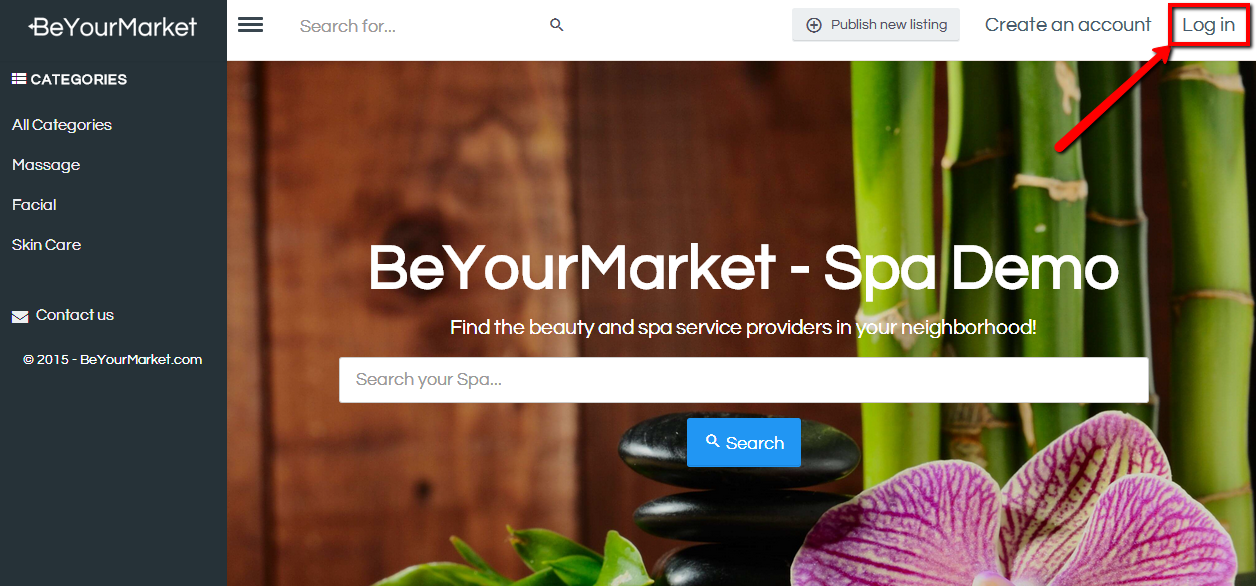
You can also hide the left bar by clicking the menu icon:  


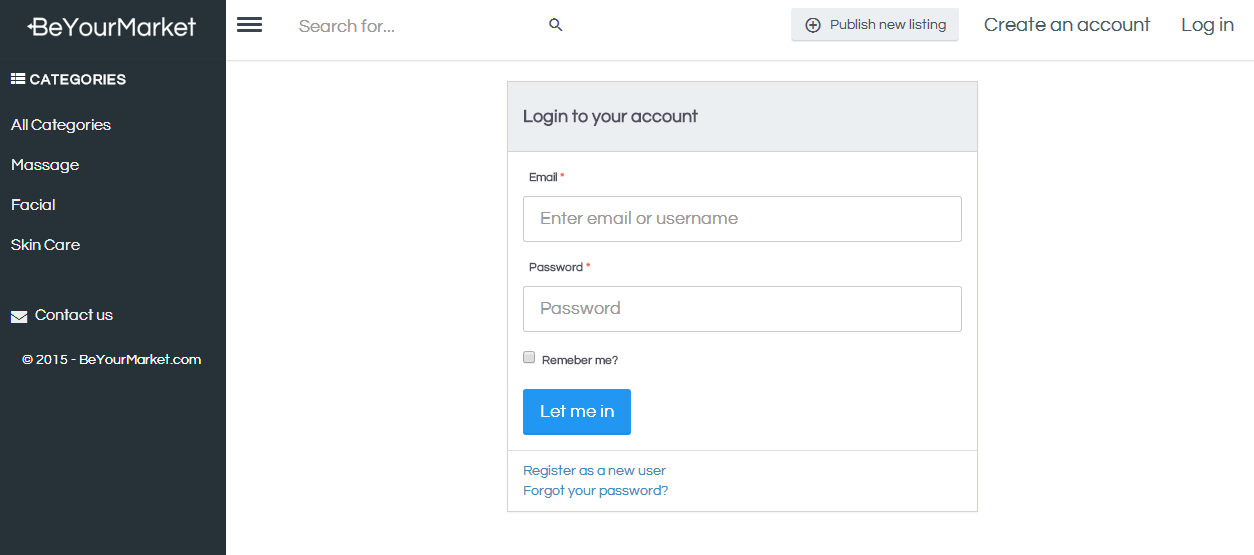
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/User+Dashboard#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jun 23, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491602&selectedPageVersions=3&selectedPageVersions=4)

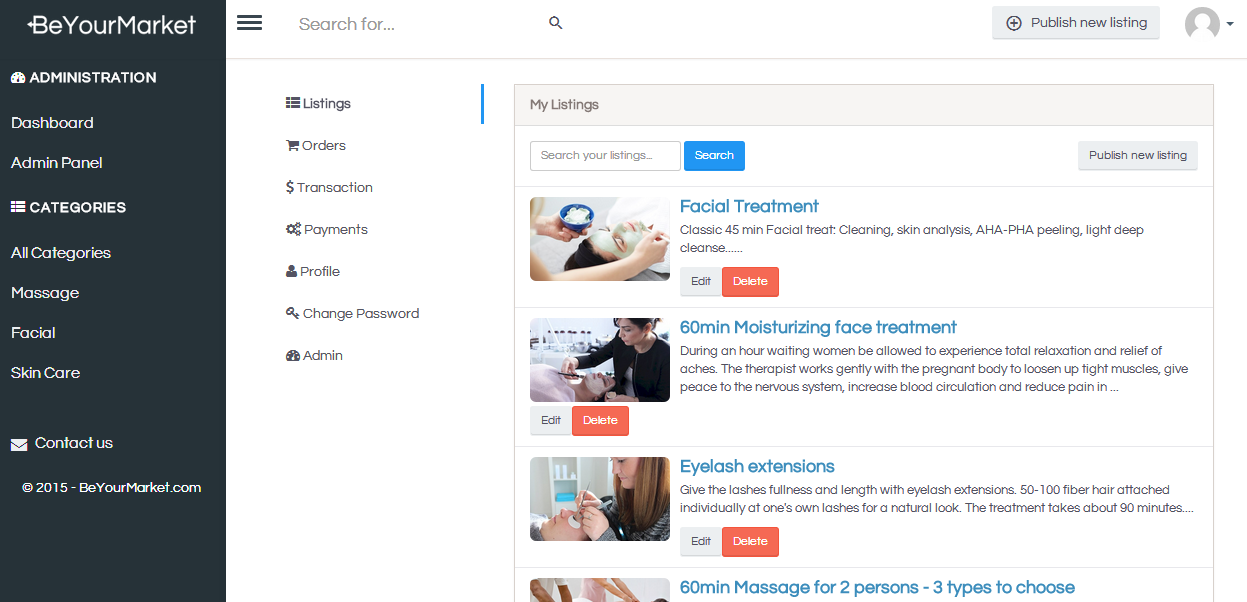
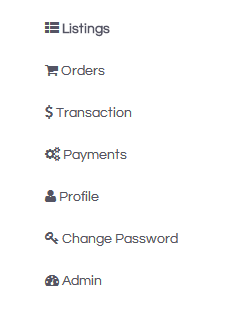
[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/User+Dashboard#page-metadata-start)

To enter User's Dashboard, find the "Log in" tab on the top right corner and type in your registered email and password.





Then you will be led to the administrating page with the following to manage:

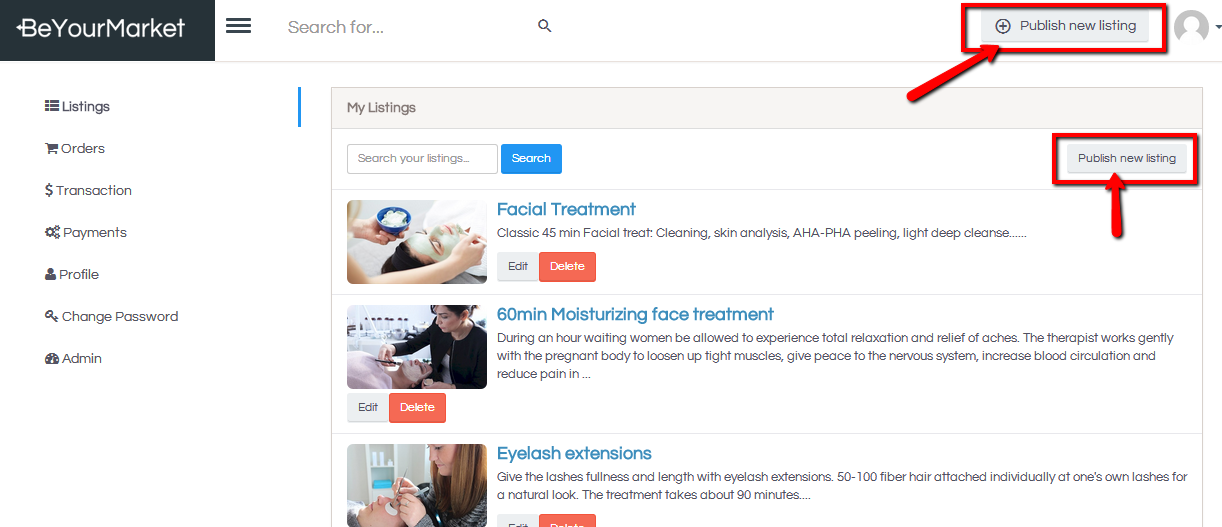
 

* **Listings**
* **Orders**
* **Transaction**
* **Payments**
* **Profile**
* **Change Password**

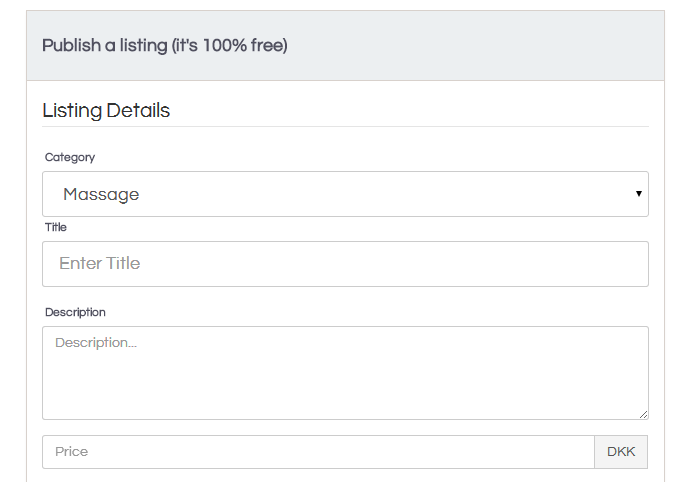
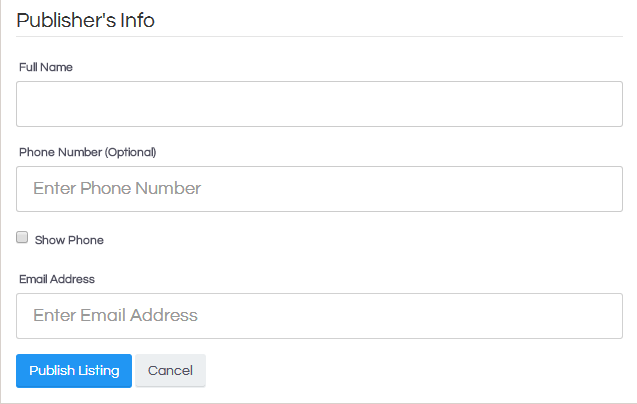
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Listing#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jun 23, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491604&selectedPageVersions=2&selectedPageVersions=3)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Listing#page-metadata-start)

You can easily create listing by clicking on the button "Publishing new listing" - both on the top right bar or when you click "listings" from User's Dashboard.  


When creating the new listing, you will need to fill in:

1. **Category**: choose the category from the menu
2. **Title**: fill in the listing title that would show on the frontend  \*tip: Keep it neat and sharp
3. **Description**: fill in more info/desciption of the listing
4. **Price**: fill in the price of this listed product/service  
   
5. Photo: You could upload photos from your drive  
   
6. Location: fill in the location and make sure it's recognizable by google map
7. Publisher's info: including name, phone number (optional and can be shown/hide on the frontend), email  
   

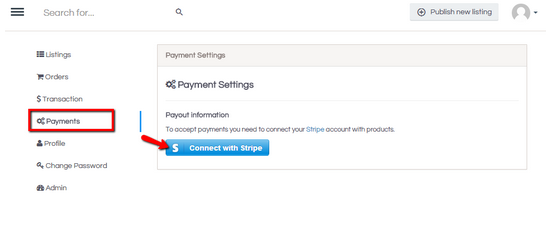
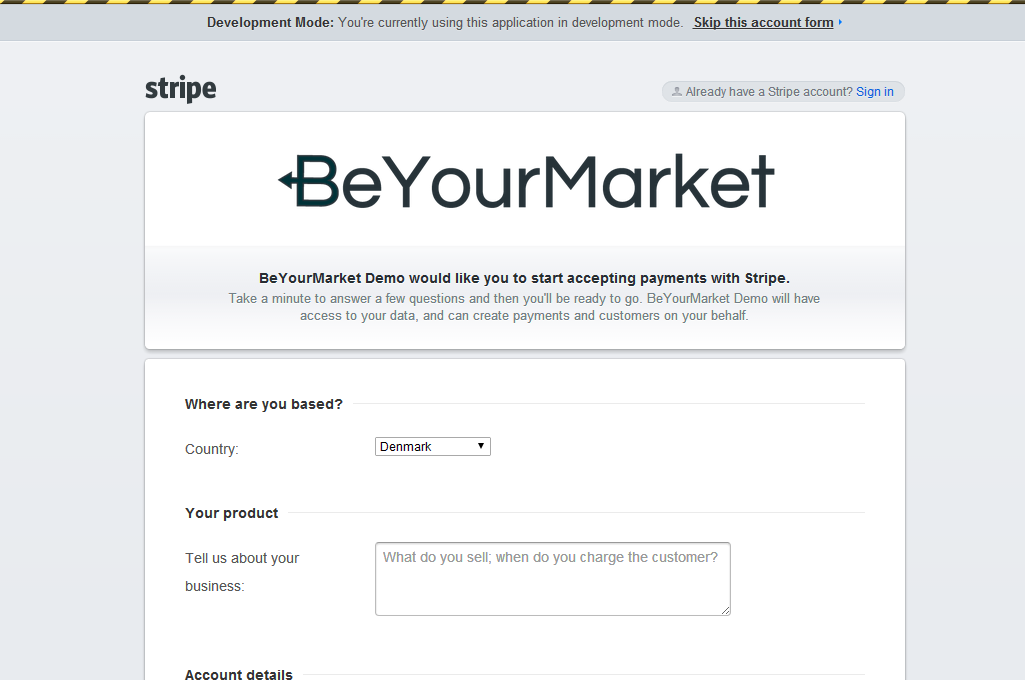
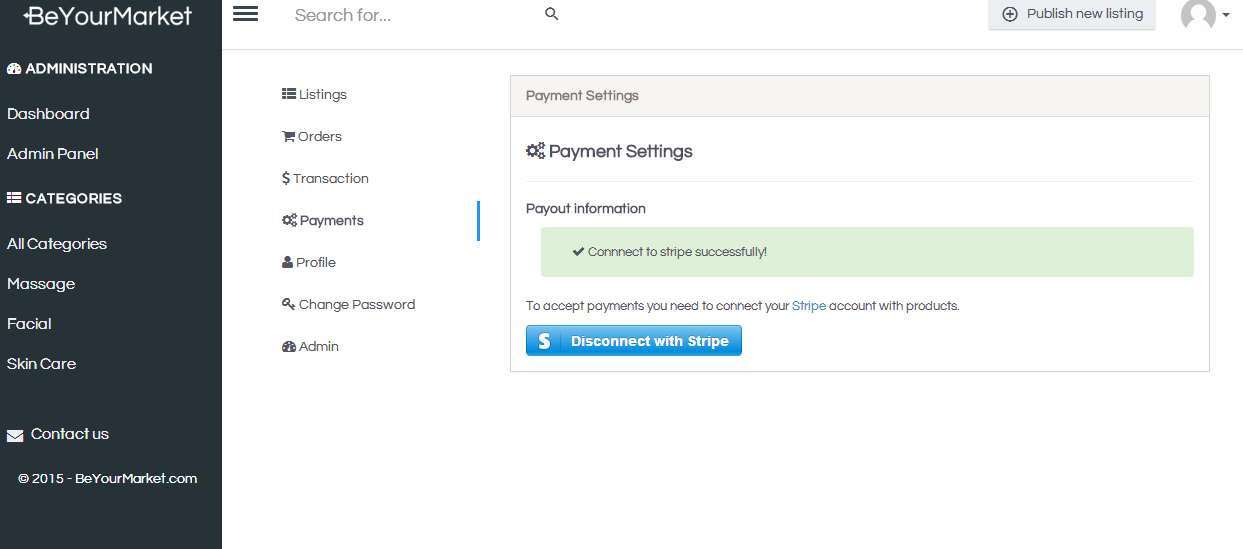
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Configure+Payment#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jul 02, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491608&selectedPageVersions=1&selectedPageVersions=2)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Configure+Payment#page-metadata-start)

To configure the payment, you need to have a Stripe account, which is an easy and safe way to accept payments.

As product/service provider, you can

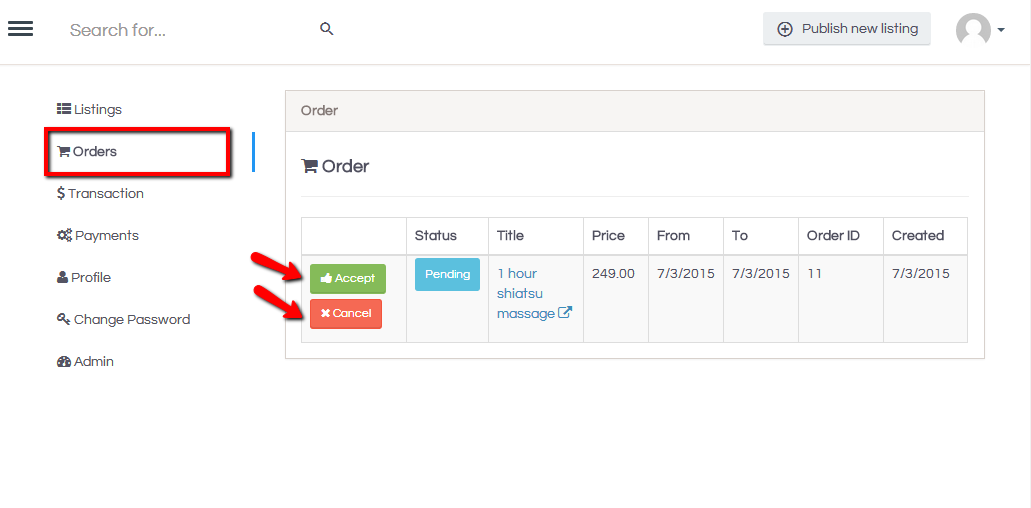
1. Click on "Payments" from the dashboard, and then "Connect with Stripe"  :  
   
2. Fill in this form that you are led to:  
   
3. Once you are connected with this one-time Stripe setup, you will be led back:  
   
4. Now you are ready to accept payment! (wink)

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Accept+orders#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jul 02, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491606&selectedPageVersions=1&selectedPageVersions=2)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Accept+orders#page-metadata-start)

As the service/product provider, you can click on "Orders" to have the overview of the ordering:



If there are orders, it means someone have ordered with a payment.  
You can choose to Accept or Cancel, so to decide if the payment will go through.

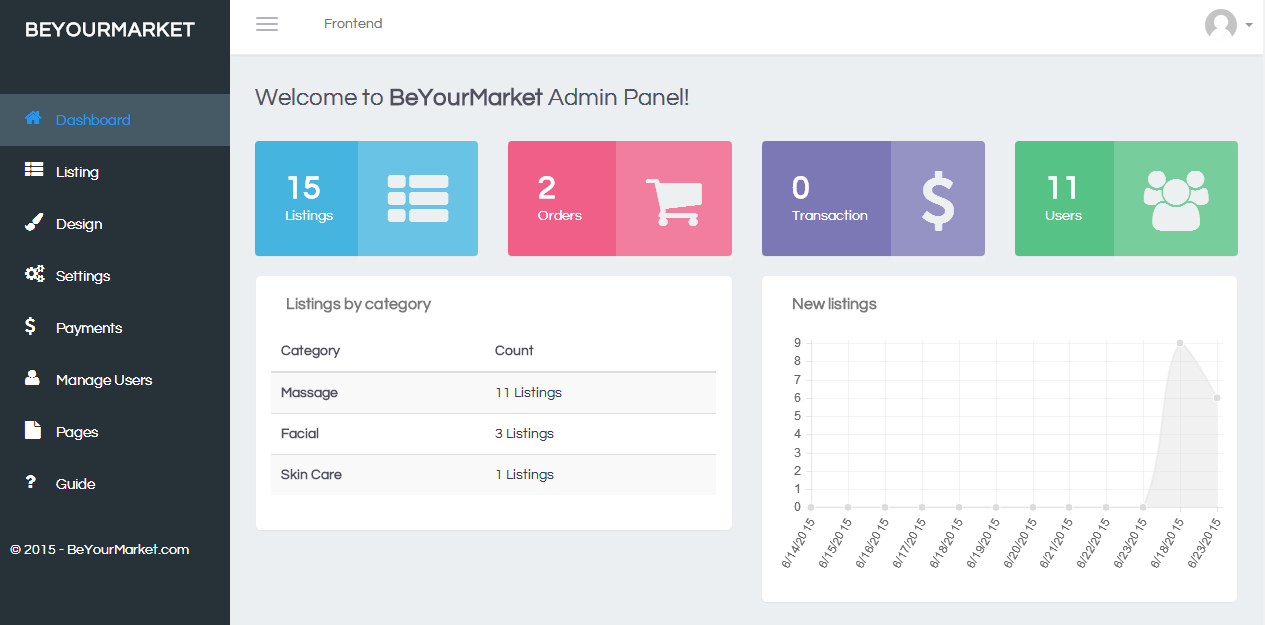
The service/product receiver may cancel the order from his/her side.  
But if the order is there, and you accept the order, then it will show "success" on the status:  


[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/pages/viewpage.action?pageId=491590#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jun 23, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491590&selectedPageVersions=3&selectedPageVersions=4)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/pages/viewpage.action?pageId=491590#page-metadata-start)

Here you would be able to administrate the back end easily with the tabs on the left bar:



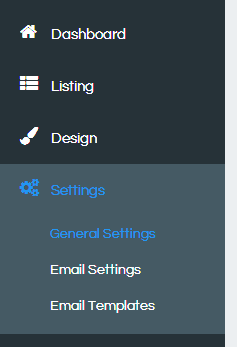
* **Dashboard**: a welcome and overview page
* **Listing**: managing categories and listings, as well as custom fields
* **Design**: customize the appearance, CSS, and javascript
* **Settings**: configure the marketplace with different settings
* **Payments**: manage payment settings and view order&transaction history
* **Manage Users**: managing the registered users
* **Pages**: add and manage new pages
* **Guide**: the link to this User Guide

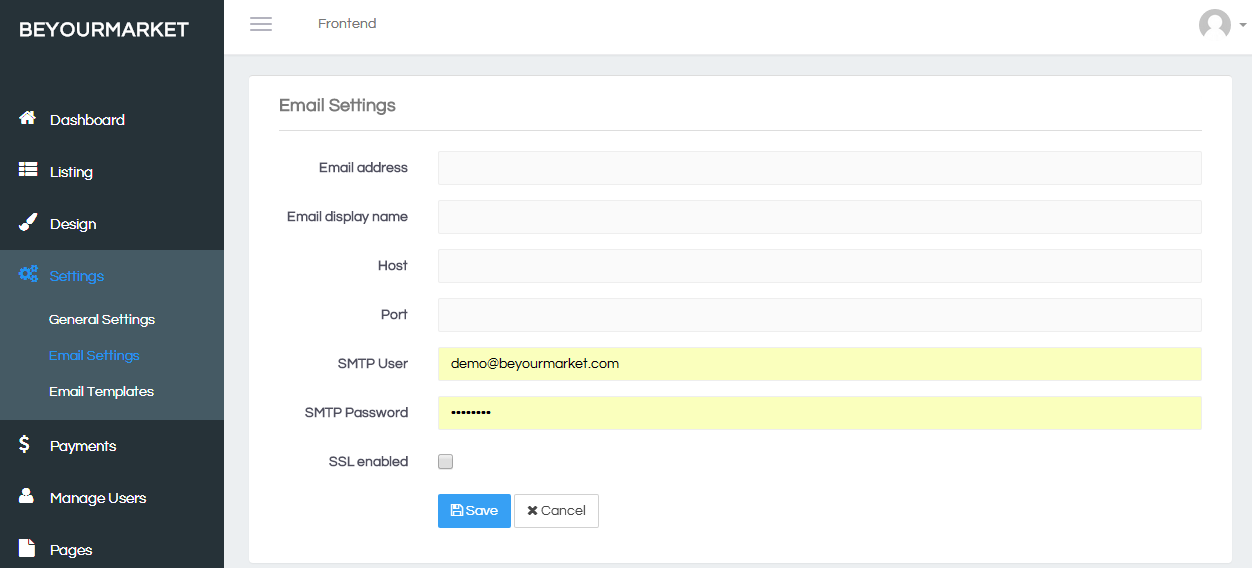
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Configure+Marketplace#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jun 23, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491615&selectedPageVersions=4&selectedPageVersions=5)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Configure+Marketplace#page-metadata-start)

Find the "Setting" tab on the left bar, then you can configure the marketplace easily with 3 sub-menu.



* **General Settings**: configure the name, description, slogan, search bar text, booking, agreement...etc  
  
* **Email Settings**: configure the email  
  
* **Email Templates**: customize the 3 email templates, incl. "sign up", "forget password", and "private message"  
  

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Category#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jul 05, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491595&selectedPageVersions=2&selectedPageVersions=3)

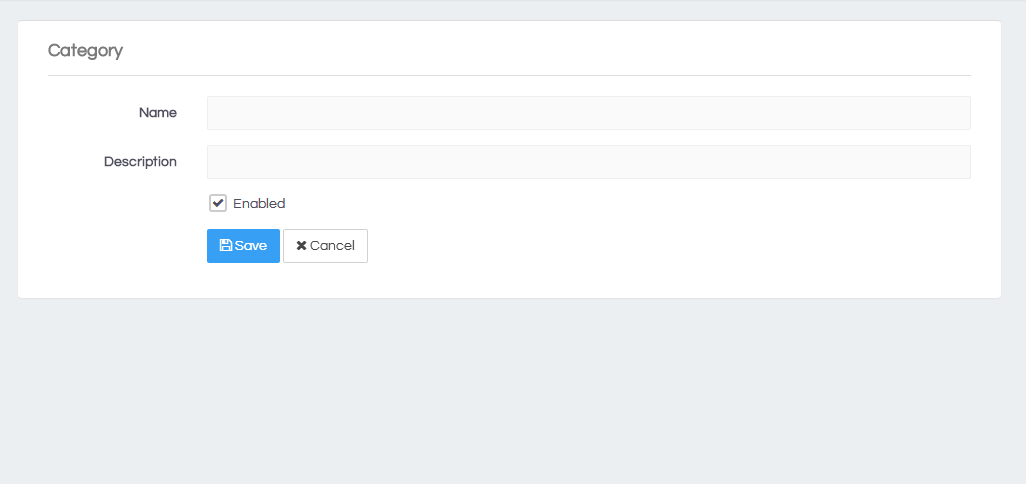
[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Category#page-metadata-start)

**Categories** is necessary for service/product providers to choose from when they create a new listing.

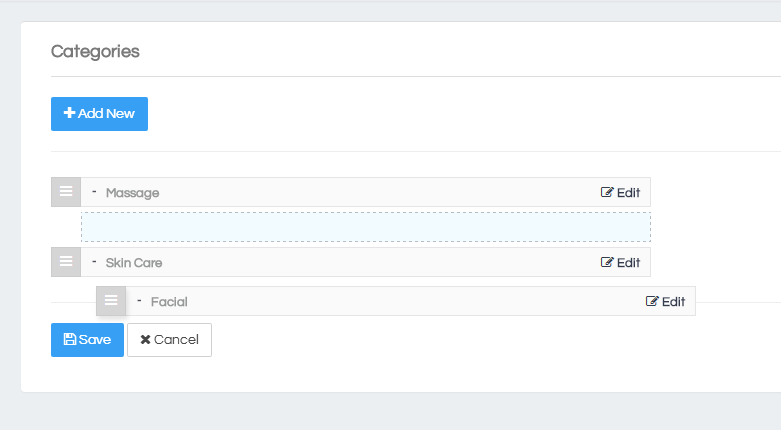
To create new Categories, you need to go to "Listing" -> "Manage Category."



When you click on "Add New," you will be directed to this page, where you can fill in the name and description of this new category:



You can also easily edit the main and sub categories or re-order the categories, simply by clicking on the icon https://beyourmarket.atlassian.net/wiki/download/attachments/491595/image2015-7-5%206%3A41%3A51.png?version=1&modificationDate=1436049715210&api=v2 and drag the bar up or down-ward.



[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Content+Page#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jul 05, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491598&selectedPageVersions=1&selectedPageVersions=2)

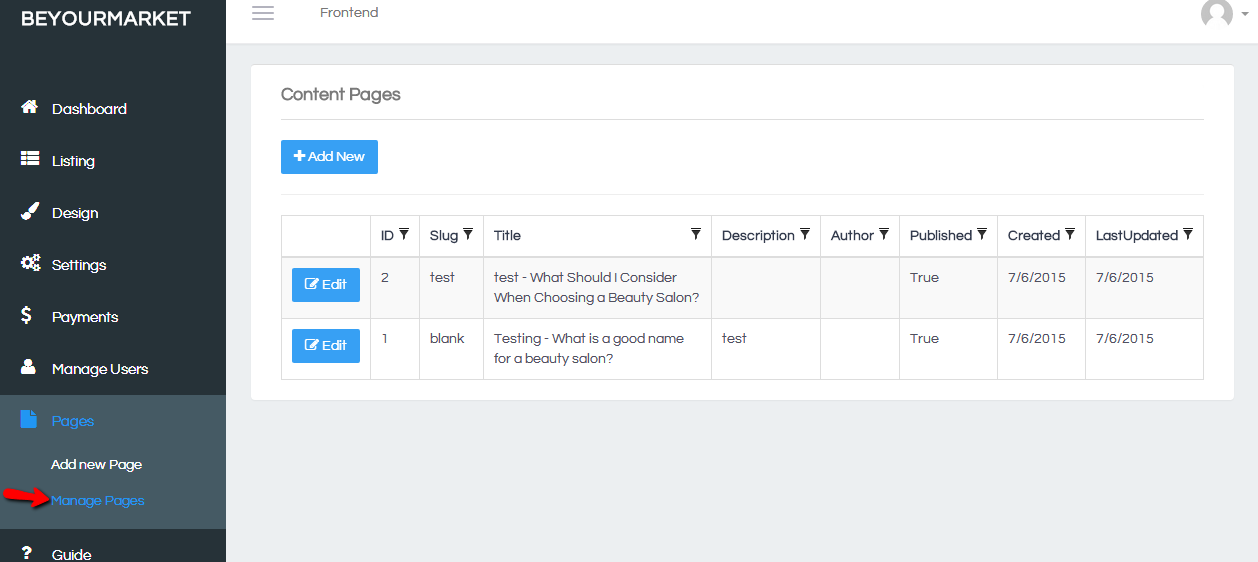
[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Content+Page#page-metadata-start)

To create a content page, go to "Pages" and you can add new ones or manage them.



Here you can fill in the page Title, Description, Slug for the unique URL, Ordering for the showing order in the Frontend, and the Content that you can easily edit.

After creating Pages, you can decide what to publish. You can also edit them again when going to "Manage Pages."



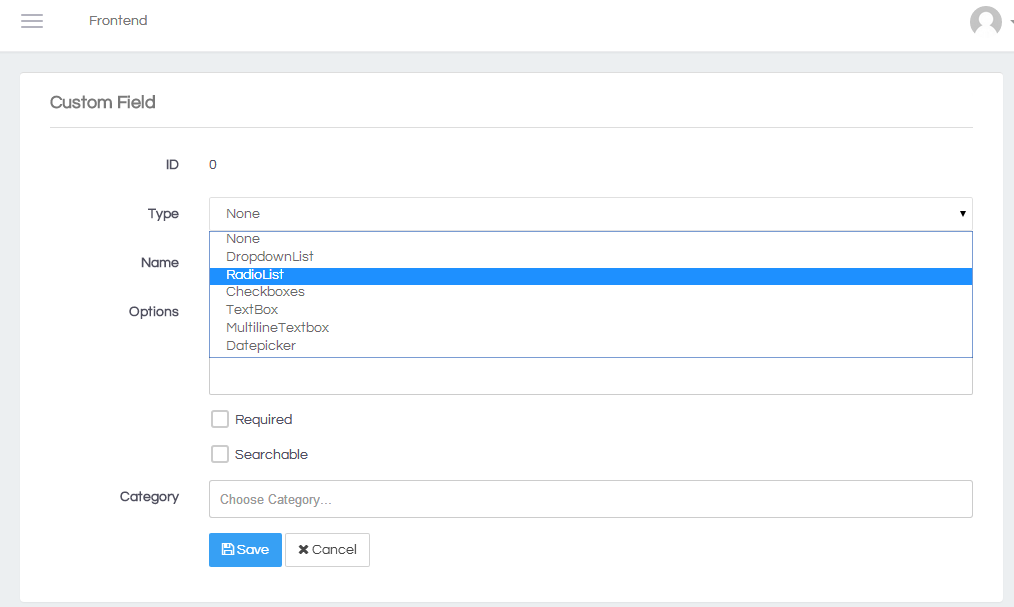
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Custom+Fields#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jul 05, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491610&selectedPageVersions=1&selectedPageVersions=2)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+Custom+Fields#page-metadata-start)

To create some custom fields for the listing creation, you should go to "Listing" -> "Custom Fields:"



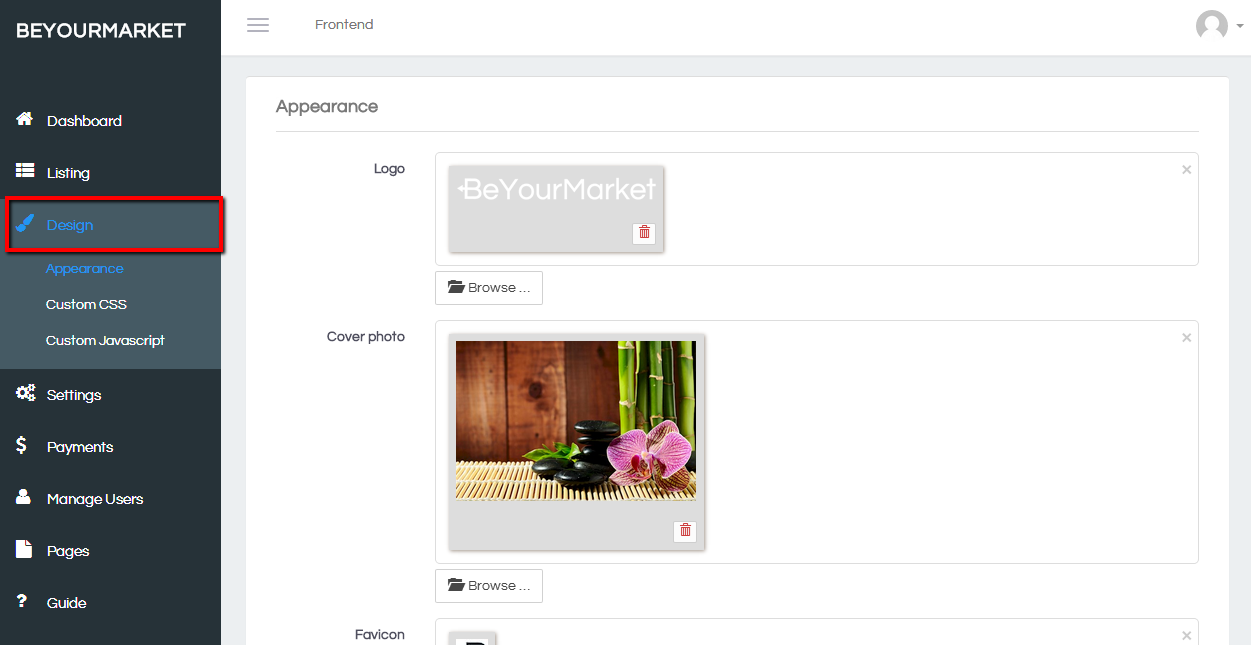
When you click on "Add New," you would be directed to this page, where you can fill in the name and options, and choose the desired field type from the drop down menu:  


[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Appearance#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified by [Ruth](https://beyourmarket.atlassian.net/wiki/display/%7Eruth0109) on [Jul 05, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491612&selectedPageVersions=3&selectedPageVersions=4)

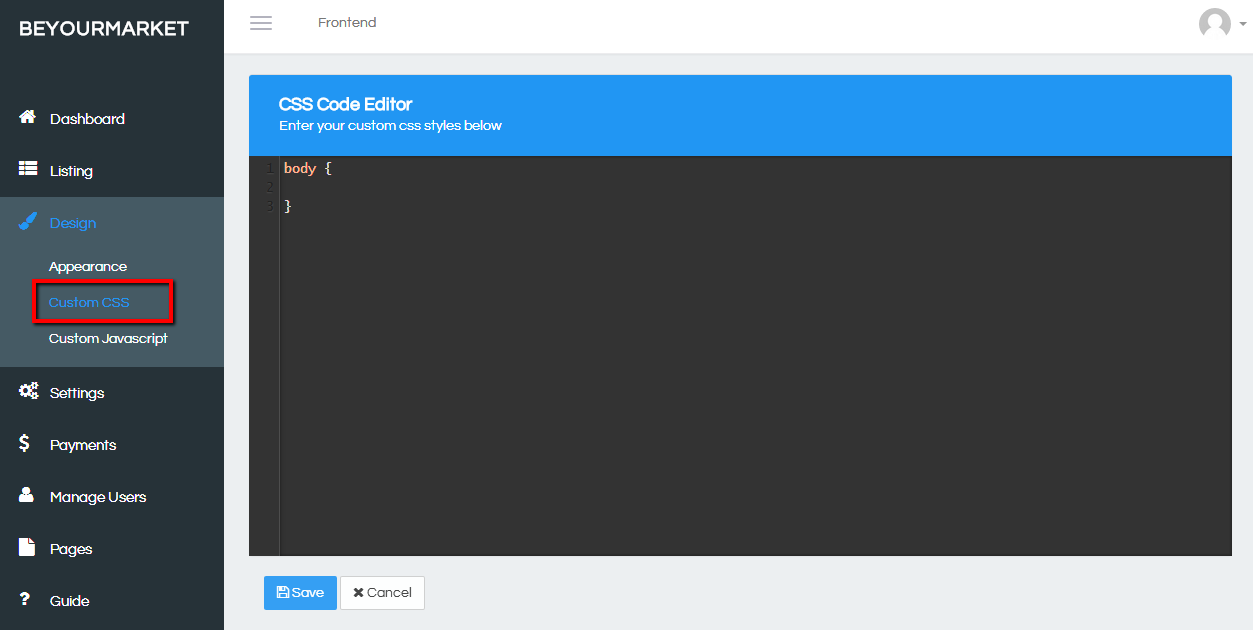
[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Appearance#page-metadata-start)

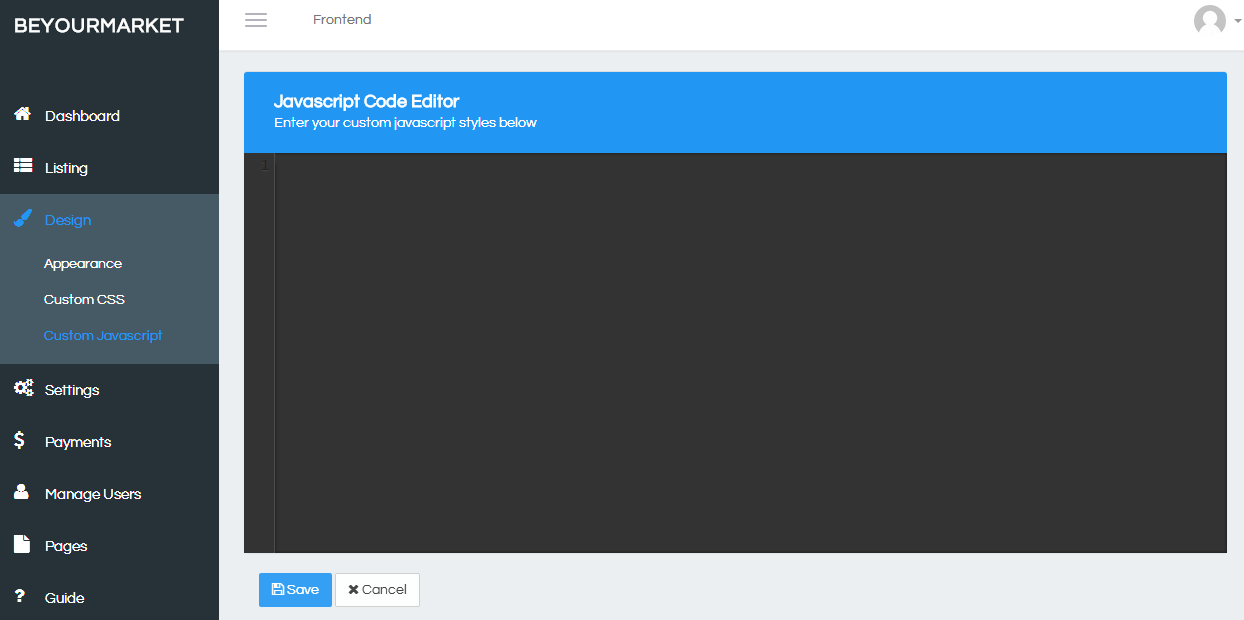
You can easily custom the appearance of the front end. Find "**Design**" from the left bar and see 3 sub-categories for customizing:



Choose **Appearance**, you will be able to custom:

* **Logo**: The logo that shows on the top of the left bar in the front end  
  **Cover photo**: The background photo in the front page
* **Favicon**: The favicon shown on the site

If you are a developer, you can also custom **CSS** and **Javascript.  
**



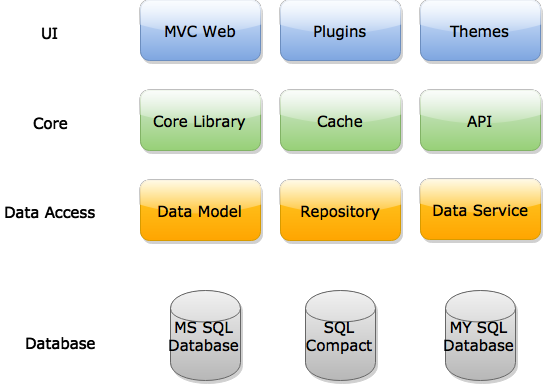
[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Architecture+overview#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1540104&selectedPageVersions=3&selectedPageVersions=4)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Architecture+overview#page-metadata-start)

BeYourMarket is a MVC web application, where it can be extended and customized with MVC pattern, Plugin and Theme.

Basically, there are 3 layers, UI, Core and Data Access(where the underlying database can be MS SQL database, SQL Compact file or even MySQL database)



From the solution point of view, it's contains:

1. BeYourMarket.Web

It's the top layer Web Application contains all the UI logic.

2. BeYourMarket.Service

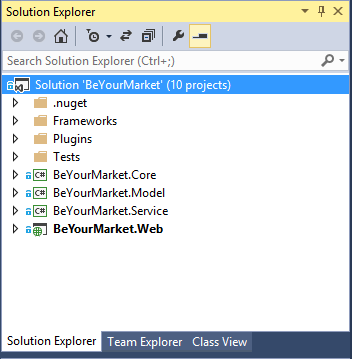
It's the data access layer with all the logic with reading/writing to the underlying database

3. BeYourMarket.Model

It's the data model layer with all the models mapped to the database

4. BeYourMarket.Core

it's the core layer with the plugin backbone components and other components that used across the library and the web application.



[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Development+environment#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Sep 16, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=491543&selectedPageVersions=6&selectedPageVersions=7)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Development+environment#page-metadata-start)

**Using the code**

The code can be download from github <https://github.com/beyourmarket/beyourmarket>

In command line, to clone the solution from github

|  |
| --- |
| git clone <https://github.com/beyourmarket/beyourmarket.git> |

This framework is ASP.NET MVC 5 based, Microsoft :NET 4.5 Framework is required.

**Visual Studio 2015**

For MS Visual Studio 2015,  [Web Essentials 2015](https://visualstudiogallery.msdn.microsoft.com/ee6e6d8c-c837-41fb-886a-6b50ae2d06a2) is highly recommended to be installed.

[Visual Studio 2015 Community](https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx) is free of charge.

**Visual Studio 2013**

For MS Visual Studio 2013, [Web Essentials 2013](https://visualstudiogallery.msdn.microsoft.com/56633663-6799-41d7-9df7-0f2a504ca361) and [Entity Framework Power Tools](https://visualstudiogallery.msdn.microsoft.com/72a60b14-1581-4b9b-89f2-846072eff19d) are highly recommended to be installed.

**Database**

Currently, BeYourMarket supports SQL Compact 4.0 and SQL Server 2014 database.

\*Microsoft SQL Server Compact 4.0 is a free, embedded database that software developers can use for building ASP.NET websites and Windows desktop applications.

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Add+a+new+data+property#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1540106&selectedPageVersions=5&selectedPageVersions=6)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Add+a+new+data+property#page-metadata-start)

In most cases, you could extend the functionalities with [Plugins](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+a+BeYourMarket+Plugin) in BeYourMarket. However, in certain scenario, you might want to modify the core functionalities and extend the pre-defined data model by adding a column or table in the database. There are 2 ways to do that.

**To add a new property on an existing model**

1. EF Code first  
     
   Go to project **BeYourMarket.Model**, find the entity that you would like to add a property. e.g. Category.cs. Then add the property to the model and update the corresponding mapping class CategoryMap.cs in the Mapping folder. Finally, update the database with migrations.  
     
   For more info on EF code first, please visit [Code First to a New Database](https://msdn.microsoft.com/en-us/data/jj193542.aspx).
2. Database first  
     
   It's a easier way but require [Entity Framework Power Tools](https://visualstudiogallery.msdn.microsoft.com/72a60b14-1581-4b9b-89f2-846072eff19d) in Visual Studio. Open Sql Management Studio, add new column to the table. Then in the Visual Studio, right click on the project **BeYourMarket.Model**, choose Entity Framework->Reverse Engineer Code First. It would then update the model accordingly.  
     
   

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+a+BeYourMarket+Plugin#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1540102&selectedPageVersions=5&selectedPageVersions=6)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Create+a+BeYourMarket+Plugin#page-metadata-start)

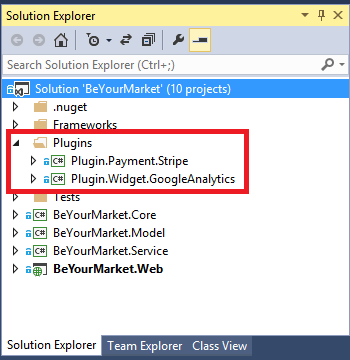
Plugins are class library which extend the functionalities in BeYourMarket. It allows easy modification, customization, and enhancement in BeYourMarket. Instead of changing the core programming of BeYourMarket, you can add functionality with BeYourMarket Plugins. For example, Payment Plugin to accept/pay using Stripe, or Widget Plugin to add GoogleAnalytics script.

**Plugin Name**

Plugin name should be simple and short to describe what the Plugin will do. The naming convention should be **Plugin.{Group}.{Name}**. {Group} is your plugin group (for example, "Payment"). {Name} is your plugin name (for example, "Stripe")

**Plugin Location**

To create a plugin, first thing is to create a new **Class Library** project with the Plugin Name in the solution. The Plugin shall be placed under Plugins directory in the solution (NOT the Plugins subdirectory located in \BeYourMarket.Web directory which is used for already deployed plugins).



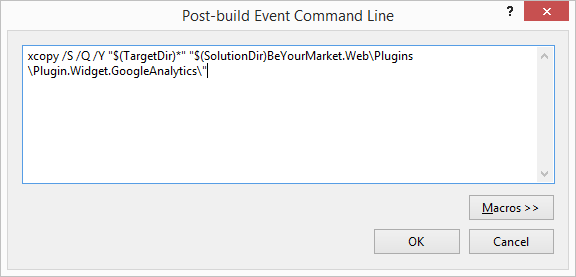
**Plugin build path**

In order for the BeYourMarket.Web to detect the plugin, the compiled dll and corresponding View files needs to be copied to the Plugins directory in BeYourMarket.Web.

Post-build Event can be configured to copy the files to the Plugins directory.

For example, to copy compiled GoogleAnalytics plugin to BeYourMarket.Web\Plugins, set the Post-build event with the following line.

|  |
| --- |
| xcopy /S /Q /Y "$(TargetDir)\*" "$(SolutionDir)BeYourMarket.Web\Plugins\Plugin.Widget.GoogleAnalytics\" |



**Plugin manifest**

For each plugin, manifest.json is required to describe the meta information of the plugin. It's json formatted file used by Plugin Manager in BeYourMarket.Web to get the information of the plugin. Ensure that "Copy to Output Directory" property of this file is set to "Copy if newer".

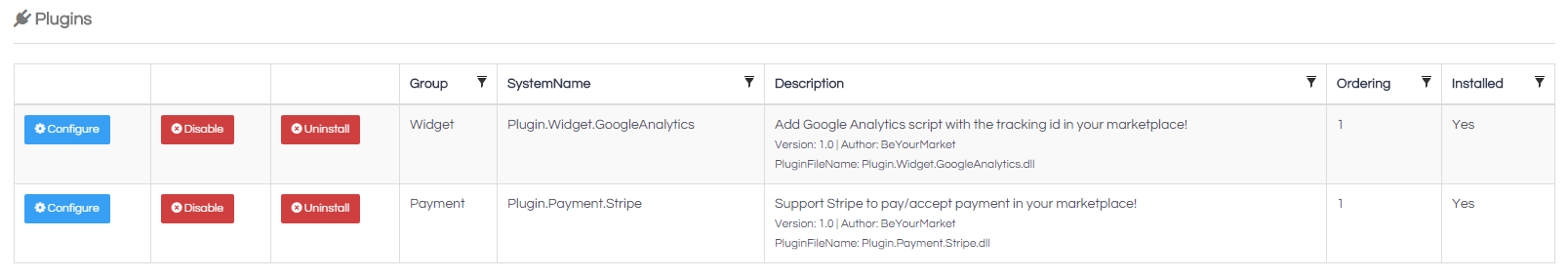
For example, manifest.json for Plugin.Widget.GoogleAnalytics

|  |
| --- |
| {      "Group": "Widget",      "SystemName": "Plugin.Widget.GoogleAnalytics",      "FriendlyName": "Google Analytics",      "Version": "1.0",      "SupportedVersions": ["1.0"],      "Author": "BeYourMarket",      "DisplayOrder": "1",      "PluginFileName": "Plugin.Widget.GoogleAnalytics.dll",      "Description":  "Add Google Analytics script with the tracking id in your marketplace!"  } |

**Plugin Manager**

To verify plugin are created correctly, compile the solution and start the application BeYourMarket.Web. Go to Admin Panel->Plugins->Manage Plugins

All Plugins detected shall be shown in the list.



Next: [Learn how to Programming Plugin](https://beyourmarket.atlassian.net/wiki/display/BYM/Programming+BeYourMarket+Plugin)

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Programming+BeYourMarket+Plugin#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1540145&selectedPageVersions=5&selectedPageVersions=6)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Programming+BeYourMarket+Plugin#page-metadata-start)

This section describes how to make your Plugin actually do something. This section contains some general ideas about Plugin development.

**Plugin programming pattern**

Plugin is based on MVC pattern, where it contains model, view and controller

**Plugin Hooks**

In order to use Plugin together with the existing application, it requires some way to hook the plugin together with the existing BeYourMarket web application.

Plugin Hooks is based on the concept where a set of callback function is defined and would be called based on certain hooks (via routes). BeYourMarket provide a HookBasePlugin class which already implements standard way to tell how to links the callback with routes. Dependency Injection is also used in order to discover plugins interface and implementation.

Here is an example:

We defined a class **GoogleAnalyticsPlugin** derived from **HookBasePlugin** where it would insert a piece script in the header on each page to track website traffic.

|  |
| --- |
| public class GoogleAnalyticsPlugin : HookBasePlugin      {          ...      } |

**Plugin Routes**

Plugin Routes define what callback should be executed on certain hook.

For example, in the code below, it defines action *Index* on controller *GoogleAnalytics* shall be called on Hook *HookName.Head.*The controller action would return a *View* with a piece of script.

|  |
| --- |
| public class GoogleAnalyticsPlugin : HookBasePlugin  {      public const string SettingTrackingID = "GoogleAnalytics\_TrackingID";      private readonly ISettingDictionaryService \_settingDictionaryService;      private readonly IUnitOfWorkAsync \_unitOfWorkAsync;      public GoogleAnalyticsPlugin(          ISettingDictionaryService settingDictionaryService,          IUnitOfWorkAsync unitOfWorkAsync)      {          \_settingDictionaryService = settingDictionaryService;          \_unitOfWorkAsync = unitOfWorkAsync;          AddRoute(HookName.Head, new RouteValueDictionary          {              { "action", "Index" },              { "controller", "GoogleAnalytics" },              { "namespaces", "Plugin.Widget.GoogleAnalytics.Controllers"},              { "area", null},              { "hookName", HookName.Head}          });     }  ...  } |

**Plugin Administration**

In most cases, the plugin would need to be configured by user with some specific information. For example, in the GoogleAnalytics plugin, user needs to configure the TrackingID in order to generate the script.

In the Admin Panel, there is a Configure button on each plugin, which will then be redirected to a configuration page on that plugin. It can be done by adding the route to the Configuration hook, as below. It would link the action *Configure* on the controller *GoogleAnalytics*.

|  |
| --- |
| AddRoute(HookName.Configuration, new RouteValueDictionary {      { "action", "Configure" },      { "controller", "GoogleAnalytics" },      { "namespaces", "Plugin.Widget.GoogleAnalytics.Controllers" },      { "area", null }  }); |

**Plugin Install/Uninstall**

When a Plugin is installed or uninstalled, you might want to insert/remove some data afterwards. To achieve this, **HookBasePlugin** provide Install/Uninstall method to be overridden which would be called after installing/uninstalling plugin.

Here is an example, GoogleAnalyticsPlugin insert a google analytics setting after installing and remove it when uninstalling.

|  |
| --- |
| /// <summary>  /// Install plugin  /// </summary>  public override void Install()  {      // Add settings      \_settingDictionaryService.Insert(new BeYourMarket.Model.Models.SettingDictionary()      {          Name = SettingTrackingID,          Value = string.Empty,          Created = DateTime.Now,          LastUpdated = DateTime.Now,          ObjectState = Repository.Pattern.Infrastructure.ObjectState.Added,          SettingID = CacheHelper.Settings.ID      });      \_unitOfWorkAsync.SaveChanges();      base.Install();  }  /// <summary>  /// Uninstall plugin  /// </summary>  public override void Uninstall()  {      // Remove settings      var settings = \_settingDictionaryService.Query(x => x.Name == SettingTrackingID).Select();      foreach (var setting in settings)      {          \_settingDictionaryService.Delete(setting);      }      \_unitOfWorkAsync.SaveChanges();      base.Uninstall();  } |

**Saving Plugin Data to the Database**

To get/save data to the database from the Plugin, the easiest way is to use ISettingDictionaryService, which can be injected via dependency injection. It could be used to save/get settings to the table Settings.

And in other cases where new tables are required, it would need to define Model and Migrations in order to get the tables created during installation. It would require a few classes and steps to do it. Example for reference is Plugin.Payment.Stripe, which is part of the BeYourMarket solution.

[**Google Analytics plugin**](https://beyourmarket.atlassian.net/wiki/display/BYM/Google+Analytics+plugin)

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Google+Analytics+plugin#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1540112&selectedPageVersions=2&selectedPageVersions=3)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Google+Analytics+plugin#page-metadata-start)

Google Analytics Plugin (Plugin.Widget.GoogleAnalytics) inserts tracking code on each page to track the website traffic.

[**Stripe Plugin**](https://beyourmarket.atlassian.net/wiki/display/BYM/Stripe+Plugin)

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Stripe+Plugin#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1540108&selectedPageVersions=1&selectedPageVersions=2)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Stripe+Plugin#page-metadata-start)

Stripe Plugin (Plugin.Payment.Stripe) provides BeYourMarket application to accept and pay via [Stripe Connect API](https://stripe.com/docs/connect).

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Translation#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1540098&selectedPageVersions=2&selectedPageVersions=3)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Translation#page-metadata-start)

The platform is by designed with support of multiple language with i18n easily. Smart internationalization for ASP.NET based on GetText / PO ecosystem is used. The only thing you would need is to translate into your own language.

To localize text in your application, surround your strings with [[[ and ]]] markup characters to mark them as translatable.

Here's an example of localizing text "Create an account" and "Log in" in a Razor view:

|  |
| --- |
| <ul class="nav navbar-nav">      <li class="dropdown messages-menu hidden-xs">          @Html.ActionLink("[[[Create an account]]]", "Register", "Account", new { area = string.Empty }, htmlAttributes: new { id = "registerLink" })      </li>      <li class="dropdown messages-menu hidden-xs">          @Html.ActionLink("[[[Log in]]]", "Login", "Account", new { area = string.Empty }, htmlAttributes: new { id = "loginLink" })      </li>  </ul> |

The template file is located at folder locale/messages.pot after the solution is built. The file can be translated with any POEditor.

When language specific po file is translate, it can be placed in your locale folder relative to the provided language, e.g. locale/fr. If you change a PO file on the fly, i18n will update accordingly; you do not need to restart your application.

The locale-specific file must be named messages.po. For example, your locale folder structure will be similar to (three languages, fr, es, and es-MX are defined):

locale/fr/messages.po

locale/es/messages.po

locale/es-MX/messages.po

If you would like to contribute as a translator, please contact us at [hello@beyourmarket.com](mailto:hello@beyourmarket.com)

[**Designer Guide**](https://beyourmarket.atlassian.net/wiki/display/BYM/Designer+Guide)

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Designer+Guide#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1179658&selectedPageVersions=2&selectedPageVersions=3)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Designer+Guide#page-metadata-start)

In many cases, you might want to customize the look and feel (Layout and css) of the BeYourMarket application. And you might even want to make a Theme that can be reusable.

You could achieve this with BeYourMarket Theme.

There is a folder called Themes under BeYourMarket.Web, and there is a folder Default which is the default theme that BeYourMarket is used. You could customize the less in order to change the color.

To add a new Theme, you could create a new folder with the theme name and define the same structure as Default.

[**Designer Guide**](https://beyourmarket.atlassian.net/wiki/display/BYM/Designer+Guide)

[Skip to end of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Designer+Guide#page-metadata-end)

* Created by [Ken](https://beyourmarket.atlassian.net/wiki/display/%7Eadmin), last modified on [Jul 11, 2015](https://beyourmarket.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=1179658&selectedPageVersions=2&selectedPageVersions=3)

[Go to start of metadata](https://beyourmarket.atlassian.net/wiki/display/BYM/Designer+Guide#page-metadata-start)

In many cases, you might want to customize the look and feel (Layout and css) of the BeYourMarket application. And you might even want to make a Theme that can be reusable.

You could achieve this with BeYourMarket Theme.

There is a folder called Themes under BeYourMarket.Web, and there is a folder Default which is the default theme that BeYourMarket is used. You could customize the less in order to change the color.

To add a new Theme, you could create a new folder with the theme name and define the same structure as Default.

# Asp.net open source marketplace framework - BeYourMarket



[kenyeungdk](http://www.codeproject.com/script/Membership/View.aspx?mid=3463569), 21 Sep 2015 [CPOL](http://www.codeproject.com/info/cpol10.aspx)

|  |  |
| --- | --- |
| https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-fill-md.png  https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-empty-md.png | 4.97 (35 votes) |
|  | |  |  |  | | --- | --- | --- | | Rate: | [vote 1vote 2vote 3vote 4vote 5](http://www.codeproject.com/Articles/1001019/Asp-net-open-source-marketplace-framework-BeYourMa) |  | |

Build your own marketplace with BeYourMarket

http://www.codeproject.com/images/warning48.png**Is your email address OK?** You are signed up for our newsletters but your email address is either unconfirmed, or has not been reconfirmed in a long time. Please **click** [**here**](http://www.codeproject.com/script/Membership/SendConfirmRequest.aspx?rp=%2fArticles%2f1001019%2fAsp-net-open-source-marketplace-framework-BeYourMa) **to have a confirmation email sent** so we can confirm your email address and start sending you newsletters again. Alternatively, you can [update your subscriptions](http://www.codeproject.com/script/Membership/Subscribe.aspx?rp=%2fArticles%2f1001019%2fAsp-net-open-source-marketplace-framework-BeYourMa).

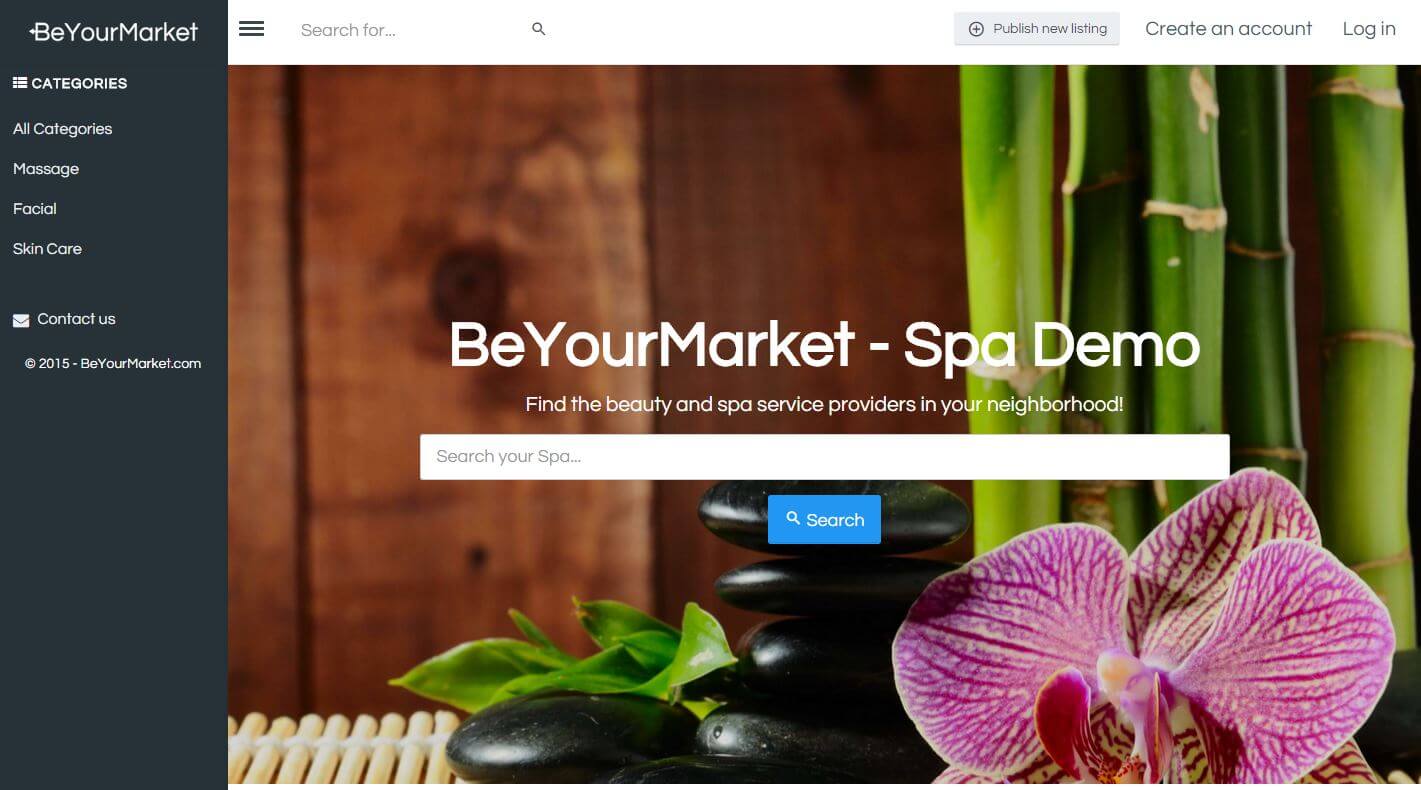
## Introduction

Many are excited about the new collaborative/sharing economy, where people leverage technologies to get what they need from each other. This has created disruptions for some industries.

What if you want to create a AirBnb for X or TaskRabbit for Y? There is not much framework/tools in .net, where you could build a marketplace easily. This article is to describe an open source framework [BeYourMarket](http://beyourmarket.com) to build and customize your own marketplace in minutes!

At the end of this article, you'll have a fully functional marketplace to find the beauty and spa service providers in your neighborhood!

Online demo  
<http://demo.beyourmarket.com>



This framework is ASP.NET MVC 5 based, so you need MS Visual Studio 2013. And, SQL Compact Server/SQL Server 2014 database is required. Web Essentials 2013 and Entity Framework Power Tools are recommended to be installed.

## Run Demo with Web Platform

You can start a marketplace in 5 minutes using the Web Platform, BeYourMarket is listed in Web App Gallery <http://www.microsoft.com/web/gallery/beyourmarket.aspx>

To install BeYourMarket, you just need to click on the Install button on the link above and follow the instruction.

## Using the code

The code can be download from github release <https://github.com/beyourmarket/beyourmarket/releases>

Or else the code can be cloned the solution from github

In command line,

Hide   Copy Code

git clone https://github.com/beyourmarket/beyourmarket.git

The solution can be opened with visual studio 2013 with support of .net 4.5

Compile the solution and run the project **BeYourMarket.Web.** It will launch an installation wizard first time, where you could specify admin username/password and database (either SQL CE compact or MS SQL server).

Remember to check "Install Sample Data", it would create a sample marketplace - beauty and spa service marketplace!

## Terminology

Marketplace is a 2-sided market. Some users would provide supply, and some would demand in certain categories and locations. In this sample, it's beauty and spa services in your local area.

In addition, the marketplace would provide a way for the service receivers to pay as well as for the service providers to get paid.

The marketplace itself needs a community manager to promote and maintain the community. It ensures the service quality and resolve the dispute if any. It would normally charge a booking fee when a service is booked. For example, AirBnb charges a booking fee when someone books a room. The more the transaction, the more the profits.

1. Service provider - users who provide the services/products; they will get paid when someone use/buy their services.

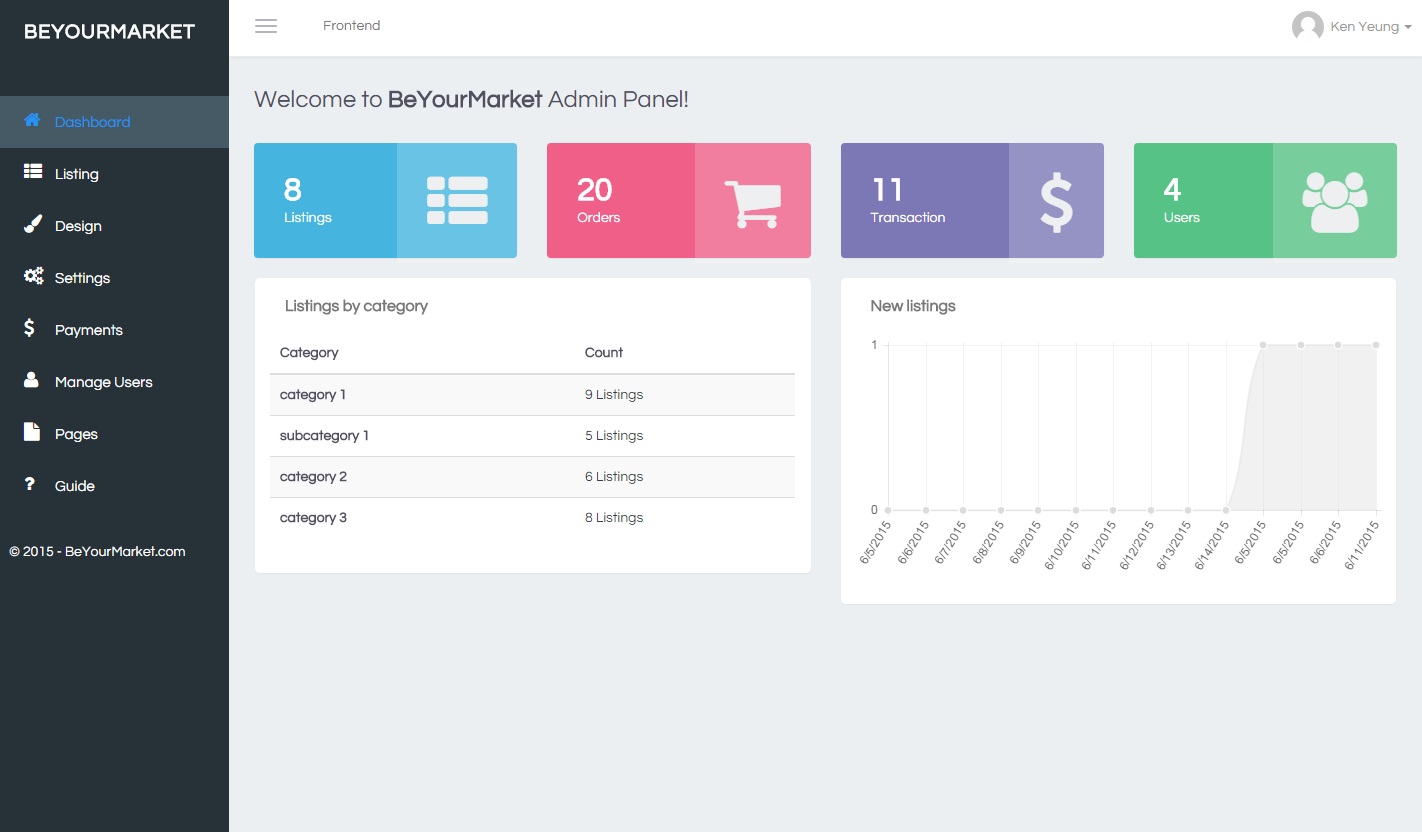
2. Service receiver - users who consume the services/products; they will pay for the services/products

3. Community Manager - administrator who creates and manages the marketplace, including listings/orders/transactions/users. He/she also resolves the issues between service providers and receivers.

Create a Marketplace

To create a marketplace, configure the settings in the admin panel.

BeYourMarket provides an admin panel where the community manager would mange the users/orders/transaction.



## Database structure

The database structure is quite simple and straightforward. It uses Entity Framework (EF) with code-first, but database-first is also supported. Models files are structured in the project **BeYourMarket.Model**

URF - Unit of Work & (extensible/generic) Repositories Framework (see reference) is used for easily extending domains (e.g. new tables/column) mapping between the database and the application.

## Create Listings (Supply)

Once users sign up, they could create a listing to their service/product. In this sample, it's the beauty and spa service.

Generally, a Listing contains 4 types of information

1. Item info (Name/Description/Price...)  
2. Location info (Latitude/Longitude)  
3. Photos  
4. Custom/Extra info

\*The format of custom info can be defined in the admin panel.

In the ListingController.cs, it's responsible for get/update/delete listing.

In the ListingUpdate method, it handles new or updates listing request.

Hide   Shrink http://www.codeproject.com/images/arrow-up-16.png  Copy Code

[HttpPost]

public async Task<ActionResult> ListingUpdate(Item item, FormCollection form, IEnumerable<HttpPostedFileBase> files)

{

bool updateCount = false;

int nextPictureOrderId = 0;

// Add new listing

if (item.ID == 0)

{

item.ObjectState = Repository.Pattern.Infrastructure.ObjectState.Added;

item.IP = Request.GetVisitorIP();

item.Expiration = DateTime.MaxValue.AddDays(-1);

item.UserID = User.Identity.GetUserId();

updateCount = true;

\_itemService.Insert(item);

}

else

{

// Update listing

var itemExisting = await \_itemService.FindAsync(item.ID);

itemExisting.Title = item.Title;

itemExisting.Description = item.Description;

itemExisting.CategoryID = item.CategoryID;

itemExisting.Enabled = item.Enabled;

itemExisting.Active = item.Active;

itemExisting.Premium = item.Premium;

itemExisting.ContactEmail = item.ContactEmail;

itemExisting.ContactName = item.ContactName;

itemExisting.ContactPhone = item.ContactPhone;

itemExisting.Latitude = item.Latitude;

itemExisting.Longitude = item.Longitude;

itemExisting.Location = item.Location;

itemExisting.ShowPhone = item.ShowPhone;

itemExisting.ShowEmail = item.ShowEmail;

itemExisting.UserID = item.UserID;

itemExisting.Price = item.Price;

itemExisting.Currency = item.Currency;

itemExisting.ObjectState = Repository.Pattern.Infrastructure.ObjectState.Modified;

\_itemService.Update(itemExisting);

}

// Delete existing fields on item

var customFieldItemQuery = await \_customFieldItemService.Query(x => x.ItemID == item.ID).SelectAsync();

var customFieldIds = customFieldItemQuery.Select(x => x.ID).ToList();

foreach (var customFieldId in customFieldIds)

{

await \_customFieldItemService.DeleteAsync(customFieldId);

}

// Get custom fields

var customFieldCategoryQuery = await \_customFieldCategoryService.Query(x => x.CategoryID == item.CategoryID).Include(x => x.MetaField.ItemMetas).SelectAsync();

var customFieldCategories = customFieldCategoryQuery.ToList();

// Update custom fields

foreach (var metaCategory in customFieldCategories)

{

var field = metaCategory.MetaField;

var controlType = (BeYourMarket.Model.Enum.Enum\_MetaFieldControlType)field.ControlTypeID;

string controlId = string.Format("customfield\_{0}\_{1}\_{2}", metaCategory.ID, metaCategory.CategoryID, metaCategory.FieldID);

var formValue = form[controlId];

if (string.IsNullOrEmpty(formValue))

continue;

formValue = formValue.ToString();

var itemMeta = new ItemMeta()

{

ItemID = item.ID,

Value = formValue,

FieldID = field.ID,

ObjectState = Repository.Pattern.Infrastructure.ObjectState.Added

};

\_customFieldItemService.Insert(itemMeta);

}

await \_unitOfWorkAsync.SaveChangesAsync();

// Update photos

if (Request.Files.Count > 0)

{

var itemPictureQuery = \_itemPictureService.Queryable().Where(x => x.ItemID == item.ID);

if (itemPictureQuery.Count() > 0)

nextPictureOrderId = itemPictureQuery.Max(x => x.Ordering);

}

foreach (HttpPostedFileBase file in files)

{

if ((file != null) && (file.ContentLength > 0) && !string.IsNullOrEmpty(file.FileName))

{

// Picture picture and get id

var picture = new Picture();

picture.MimeType = "image/jpeg";

\_pictureService.Insert(picture);

await \_unitOfWorkAsync.SaveChangesAsync();

// Format is automatically detected though can be changed.

ISupportedImageFormat format = new JpegFormat { Quality = 90 };

Size size = new Size(500, 0);

//https://naimhamadi.wordpress.com/2014/06/25/processing-images-in-c-easily-using-imageprocessor/

// Initialize the ImageFactory using the overload to preserve EXIF metadata.

using (ImageFactory imageFactory = new ImageFactory(preserveExifData: true))

{

var path = Path.Combine(Server.MapPath("~/images/item"), string.Format("{0}.{1}", picture.ID.ToString("00000000"), "jpg"));

// Load, resize, set the format and quality and save an image.

imageFactory.Load(file.InputStream)

.Resize(size)

.Format(format)

.Save(path);

}

var itemPicture = new ItemPicture();

itemPicture.ItemID = item.ID;

itemPicture.PictureID = picture.ID;

itemPicture.Ordering = nextPictureOrderId;

\_itemPictureService.Insert(itemPicture);

nextPictureOrderId++;

}

}

await \_unitOfWorkAsync.SaveChangesAsync();

// Update statistics count

if (updateCount)

{

\_sqlDbService.UpdateCategoryItemCount(item.CategoryID);

\_dataCacheService.RemoveCachedItem(CacheKeys.Statistics);

}

return RedirectToAction("Listings");

}

## Service Booking (Demand)

Once there are some listings (services/products) which users can book and pay, we need to create orders for these request.

PaymentController.cs handles all the order and payment requests.

In the Order method, it creates order with what users requested; when it succeeds, it would redirect the user to the payment page afterwards.

Hide   Shrink http://www.codeproject.com/images/arrow-up-16.png  Copy Code

public async Task<ActionResult> Order(Order order)

{

var item = await \_itemService.FindAsync(order.ItemID);

if (item == null)

return new HttpNotFoundResult();

// Check if payment method is setup on user or the platform

var descriptors = \_pluginFinder.GetPluginDescriptors<IHookPlugin>(LoadPluginsMode.InstalledOnly, "Payment").Where(x => x.Enabled);

if (descriptors.Count() == 0)

{

TempData[TempDataKeys.UserMessageAlertState] = "bg-danger";

TempData[TempDataKeys.UserMessage] = "[[[The provider has not setup the payment option yet, please contact the provider.]]]";

return RedirectToAction("Listing", "Listing", new { id = order.ItemID });

}

foreach (var descriptor in descriptors)

{

var controllerType = descriptor.Instance<IHookPlugin>().GetControllerType();

var controller = ContainerManager.GetConfiguredContainer().Resolve(controllerType) as IPaymentController;

if (!controller.HasPaymentMethod(item.UserID))

{

TempData[TempDataKeys.UserMessageAlertState] = "bg-danger";

TempData[TempDataKeys.UserMessage] = string.Format("[[[The provider has not setup the payment option for {0} yet, please contact the provider.]]]", descriptor.FriendlyName);

return RedirectToAction("Listing", "Listing", new { id = order.ItemID });

}

}

if (order.ID == 0)

{

order.ObjectState = Repository.Pattern.Infrastructure.ObjectState.Added;

order.Created = DateTime.Now;

order.Modified = DateTime.Now;

order.Status = (int)Enum\_OrderStatus.Created;

order.UserProvider = item.UserID;

order.UserReceiver = User.Identity.GetUserId();

if (order.UserProvider == order.UserReceiver)

{

TempData[TempDataKeys.UserMessageAlertState] = "bg-danger";

TempData[TempDataKeys.UserMessage] = "[[[You cannot book the item from yourself!]]]";

return RedirectToAction("Listing", "Listing", new { id = order.ItemID });

}

if (order.ToDate.HasValue && order.FromDate.HasValue)

{

order.Description = string.Format("{0} #{1} ([[[From]]] {2} [[[To]]] {3})",

item.Title, item.ID,

order.FromDate.Value.ToShortDateString(), order.ToDate.Value.ToShortDateString());

order.Quantity = order.ToDate.Value.Date.AddDays(1).Subtract(order.FromDate.Value.Date).Days;

order.Price = order.Quantity \* item.Price;

}

else

{

order.Description = string.Format("{0} #{1}", item.Title, item.ID);

order.Quantity = 1;

order.Price = item.Price;

}

\_orderService.Insert(order);

}

await \_unitOfWorkAsync.SaveChangesAsync();

ClearCache();

return RedirectToAction("Payment", new { id = order.ID });

}

## Payment integration with Stripe

Services/Products in marketplaces need to get users paid. [Stripe](http://stripe.com) provides a protocol to accept payments. BeYourMarket is integated with Stripe Connect API, so the users can pay and get paid (even with BitCoin!) easily. If needed, it can also be integrated with other payment API like braintree/paypal checkout express by writing a plugin in BeYourMarket.

In Payment.cshtml, it embeds the stripe form with checkout. With the following html code, it would take care of building forms, validating input, and securing your users' card data.

The key thing to notice is the data-key attribute, which identifies your website when communicating with Stripe. The publishable API key can be configured in the Admin Panel.

Hide   Copy Code

<div class="form-group">

<form class="form" action="@Url.Action("Payment")" method="post" role="form">

<script src="https://checkout.stripe.com/checkout.js" class="stripe-button"

data-key="@CacheHelper.GetSettingDictionary(Plugin.Payment.Stripe.StripePlugin.SettingStripePublishableKey).Value"

data-image="https://stripe.com/img/documentation/checkout/marketplace.png"

data-name="@CacheHelper.Settings.Name"

data-description="@Model.Description"

data-currency="@CacheHelper.Settings.Currency"

data-amount="@Model.PriceInCents">

</script>

</form>

</div>

Once the user payed with their credit card, Stripe would generate and send back a secured token that could be used to charge them money. In the marketplace, the service provider usually would need to confirm their availability and accept the offer before the transaction is done. Therefore, we could save the token and create a transaction to be captured later on.

In PaymentController.cs, the Payment method handles the callback from Stripe after the user enter a valid card number and click pay. It contains a secured token as well as the orderid.

Hide   Shrink http://www.codeproject.com/images/arrow-up-16.png  Copy Code

[HttpPost]

public async Task<ActionResult> Payment(int id, string stripeToken, string stripeEmail)

{

var selectQuery = await \_orderService.Query(x => x.ID == id).Include(x => x.Item).SelectAsync();

// Check if order exists

var order = selectQuery.FirstOrDefault();

if (order == null)

return new HttpNotFoundResult();

var stripeConnectQuery = await \_stripConnectService.Query(x => x.UserID == order.UserProvider).SelectAsync();

var stripeConnect = stripeConnectQuery.FirstOrDefault();

if (stripeConnect == null)

return new HttpNotFoundResult();

//https://stripe.com/docs/checkout

var charge = new StripeChargeCreateOptions();

// always set these properties

charge.Amount = order.PriceInCents;

charge.Currency = CacheHelper.Settings.Currency;

charge.Source = new StripeSourceOptions()

{

TokenId = stripeToken

};

// set booking fee

var bookingFee = (int)Math.Round(CacheHelper.Settings.TransactionFeePercent \* order.PriceInCents);

if (bookingFee < CacheHelper.Settings.TransactionMinimumFee \* 100)

bookingFee = (int)(CacheHelper.Settings.TransactionMinimumFee \* 100);

charge.ApplicationFee = bookingFee;

charge.Capture = false;

charge.Description = order.Description;

charge.Destination = stripeConnect.stripe\_user\_id;

var chargeService = new StripeChargeService(CacheHelper.GetSettingDictionary("StripeApiKey").Value);

StripeCharge stripeCharge = chargeService.Create(charge);

// Update order status

order.Status = (int)Enum\_OrderStatus.Pending;

order.PaymentPlugin = StripePlugin.PluginName;

\_orderService.Update(order);

// Save transaction

var transaction = new StripeTransaction()

{

OrderID = id,

ChargeID = stripeCharge.Id,

StripeEmail = stripeEmail,

StripeToken = stripeToken,

Created = DateTime.Now,

LastUpdated = DateTime.Now,

FailureCode = stripeCharge.FailureCode,

FailureMessage = stripeCharge.FailureMessage,

ObjectState = Repository.Pattern.Infrastructure.ObjectState.Added

};

\_transactionService.Insert(transaction);

await \_unitOfWorkAsync.SaveChangesAsync();

await \_unitOfWorkAsyncStripe.SaveChangesAsync();

ClearCache();

// Payment succeeded

if (string.IsNullOrEmpty(stripeCharge.FailureCode))

{

TempData[TempDataKeys.UserMessage] = "[[[Thanks for your order! You payment will not be charged until the provider accepted your request.]]]";

return RedirectToAction("Orders", "Payment");

}

else

{

TempData[TempDataKeys.UserMessageAlertState] = "bg-danger";

TempData[TempDataKeys.UserMessage] = stripeCharge.FailureMessage;

return RedirectToAction("Payment");

}

}

## Accept payments and charge transaction fee

Once the service provider accepts what the user requested through the dashboard panel. The transaction would need to be captured. Money will get charged from the user's credit card, and service provider will get paid and provide the service, and the community manager will be awarded with a booking fee. Everyone is happy!

In the PaymentController.cs, the OrderAction handles the order action if service provider accept or decline the request. If he/she accepts, it would create a charge with Stripe API and capture the payment.

Hide   Shrink http://www.codeproject.com/images/arrow-up-16.png  Copy Code

[HttpPost]

public async Task<ActionResult> OrderAction(int id, int status)

{

var order = await \_orderService.FindAsync(id);

if (order == null)

return new HttpNotFoundResult();

var descriptor = \_pluginFinder.GetPluginDescriptorBySystemName<IHookPlugin>(order.PaymentPlugin);

if (descriptor == null)

return new HttpNotFoundResult("Not found");

var controllerType = descriptor.Instance<IHookPlugin>().GetControllerType();

var controller = ContainerManager.GetConfiguredContainer().Resolve(controllerType) as IPaymentController;

string message = string.Empty;

var orderResult = controller.OrderAction(id, status, out message);

var result = new

{

Success = orderResult,

Message = message

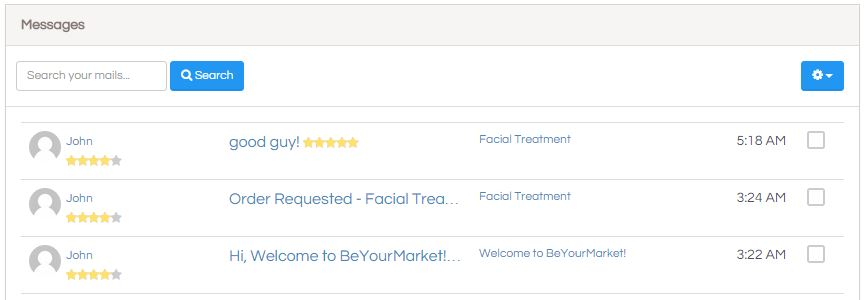
};

return Json(result, JsonRequestBehavior.AllowGet);

}

## Communication

BeYourMarket has an message box system which allows service providers and receivers to communciate with each other.



Message thread would be created whenever a user start a conversion first time. If there is a conversion between the users (MessageParticipanets) before, the same message thread will be used. Each message thread contains a list of messages. Each message would have a state if it's read or not as well.

Hide   Copy Code

public partial class MessageThread : Repository.Pattern.Ef6.Entity

{

public MessageThread()

{

this.Messages = new List<Message>();

this.MessageParticipants = new List<MessageParticipant>();

}

public int ID { get; set; }

public string Subject { get; set; }

public Nullable<int> ListingID { get; set; }

public System.DateTime Created { get; set; }

public System.DateTime LastUpdated { get; set; }

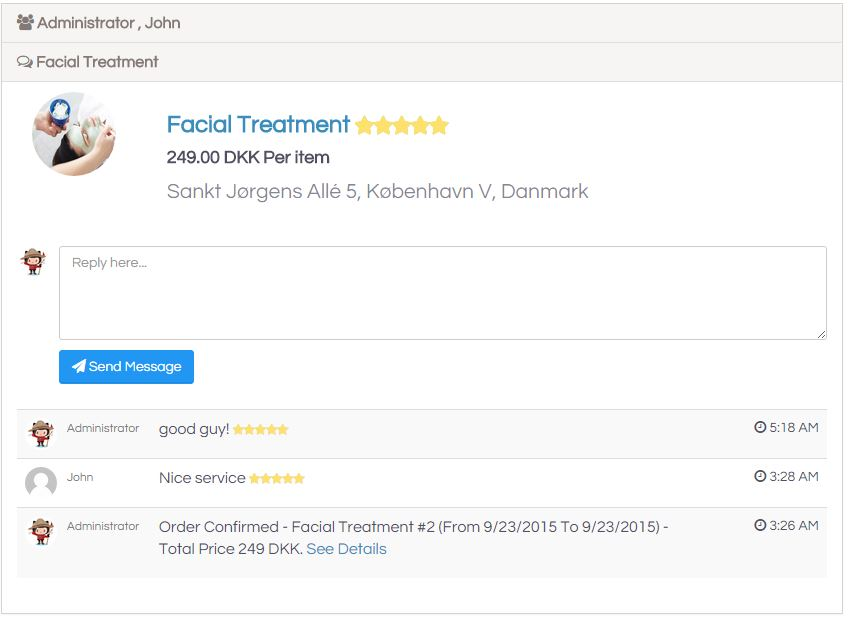
public virtual Listing Listing { get; set; }

public virtual ICollection<Message> Messages { get; set; }

public virtual ICollection<MessageParticipant> MessageParticipants { get; set; }

}

Each message thread is associated with each service provider and receiver can also associated on a specific listing (service or product), If a message is unread, a notification icon will be shown on the top navigation bar.



## Review and Rating

After the service, both server provider and receiver would have a chance to give feedback and ratings to each others. In this way, other users would be able to reviews about a service or product and whether it's good or not.



## Internalization

The platform is by designed with support of multiple language with i18n easily. Smart internationalization for ASP.NET based on GetText / PO ecosystem (see reference) is used. The only thing you would need is to translate into your own language.

To localize text in your application, surround your strings with [[[ and ]]] markup characters to mark them as translatable.

Here's an example of localizing text "Create an account" and "Log in" in a Razor view:

Hide   Copy Code

<ul class="nav navbar-nav">

<li class="dropdown messages-menu hidden-xs">

@Html.ActionLink("[[[Create an account]]]", "Register", "Account", new { area = string.Empty }, htmlAttributes: new { id = "registerLink" })

</li>

<li class="dropdown messages-menu hidden-xs">

@Html.ActionLink("[[[Log in]]]", "Login", "Account", new { area = string.Empty }, htmlAttributes: new { id = "loginLink" })

</li>

</ul>

The template file is located at locale/messages.pot after the solution is built.

## Conclusion

At the end, we would see how an beauty and spa marketplace is built with asp.net mvc. Collorative consumption is a trend and more tradtional business will be transformed into marketplace business.

Technology would need to be easily customized and scaled in order to improve the product time-to-market. BeYourMarket is started with an initiative, that an opensource framework where developers could help business build and customize marketplaces easily. It shall be also easy to extend with other features and components.

## License

BeYourMarket is licensed with MIT License.

## Reference

BeYourMarket - An opensource asp.net marketplace - <http://beyourmarket.com>

BeYourMarket Documentation - <https://beyourmarket.atlassian.net/wiki/display/BYM/BeYourMarket>

Smart internationalization for ASP.NET - <https://github.com/turquoiseowl/i18n>

URF - Unit of Work & (extensible/generic) Repositories Framework - <https://genericunitofworkandrepositories.codeplex.com/>