Prepare development environment

- IIS7.0 + required
- Visual Studio 2010(SP1)
- MSSQL 2008 Express Client Only
- .Net framework 4.5 installed

Get all the source code located at: svn://svn2.gammatrix.com/cms2012

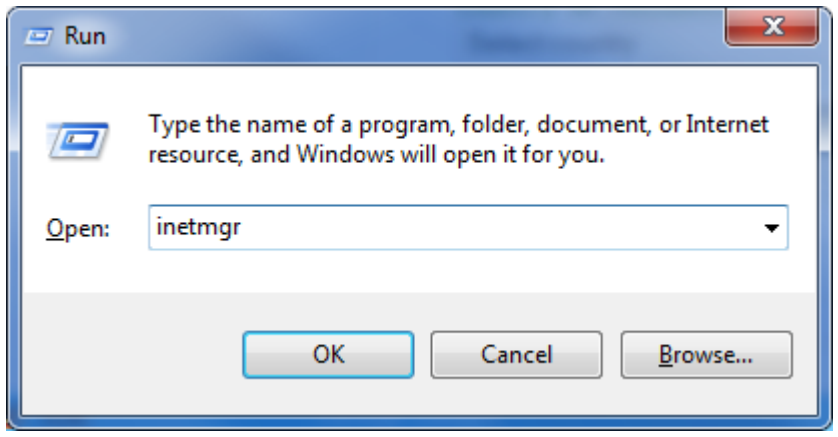
DEV environment
http://dev.gammatrix.com:2012/
http://demo1.gammatrix.com:2012/en/Register
http://demo1.gammatrix.com:2012/he/Register

Setup your local server

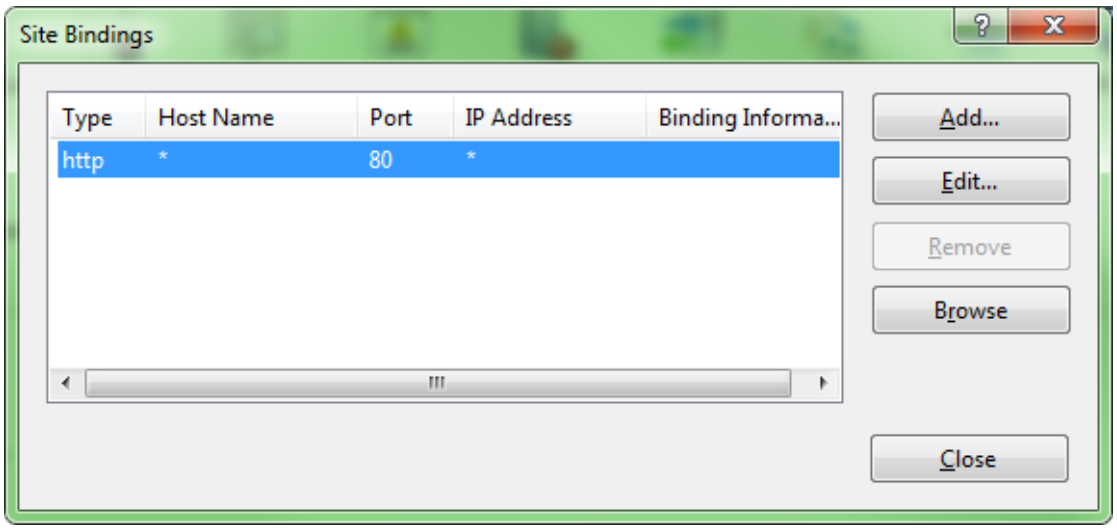
Create the Event Log Entry

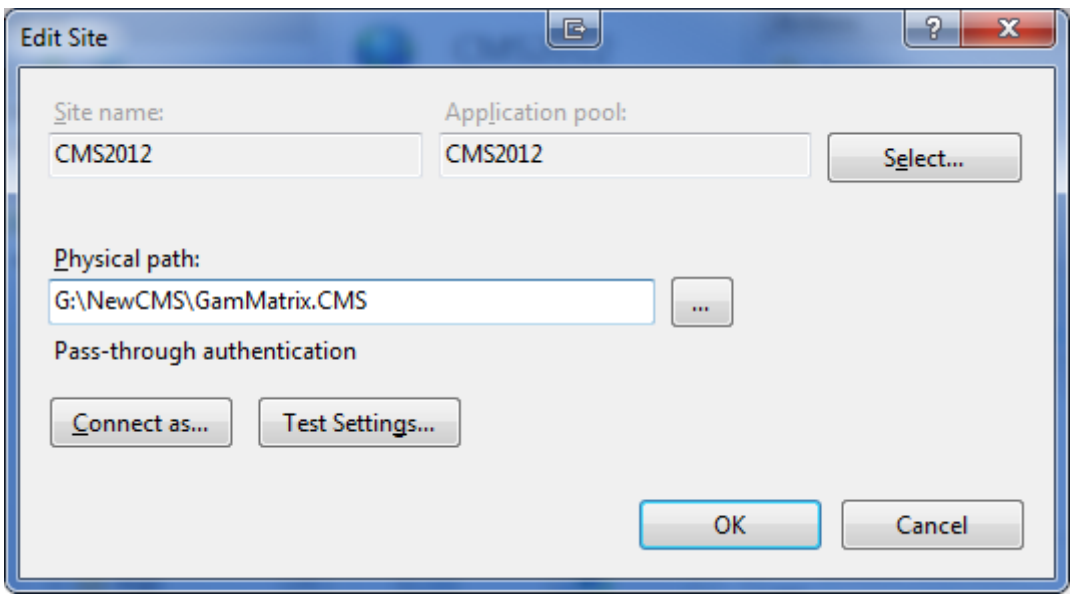
```
eventcreate /ID 1 /L APPLICATION /T INFORMATION /SO CMS2012  /D "CMS2012"
eventcreate /ID 1 /L APPLICATION /T INFORMATION /SO CasinoEngine  /D "CasinoEngine"
```

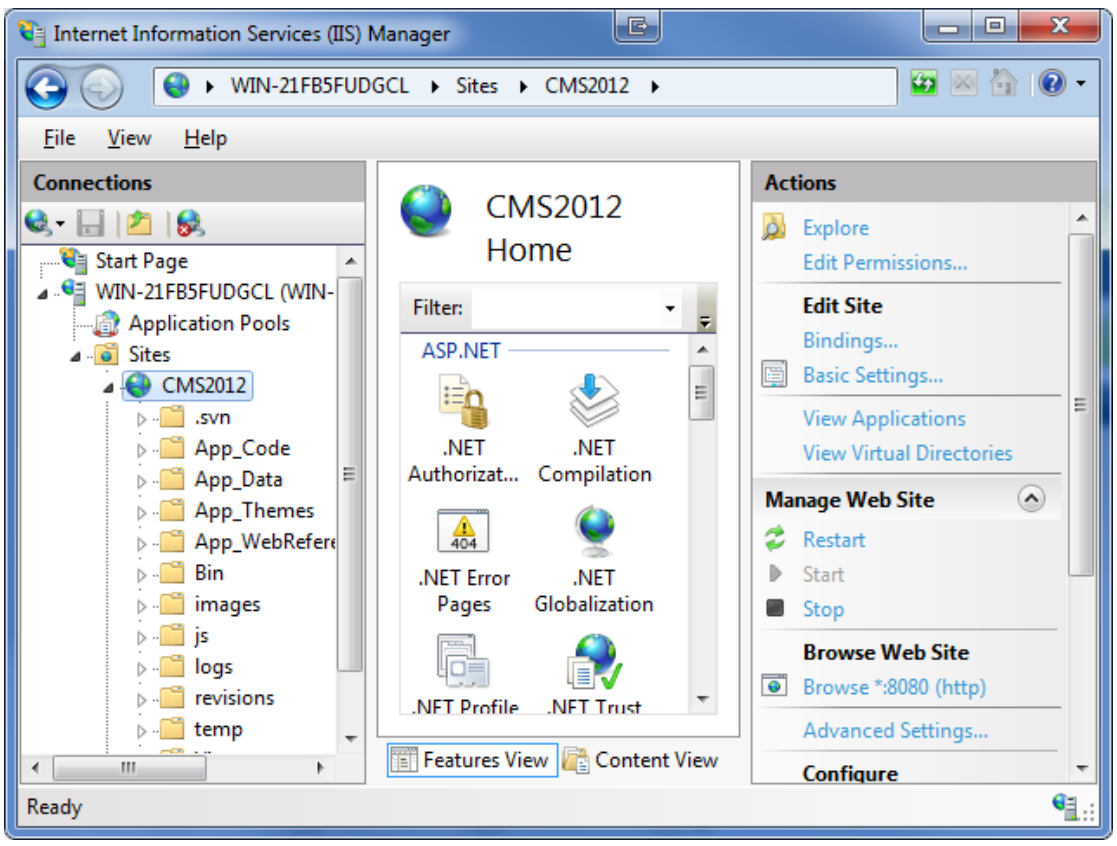
IIS configuration



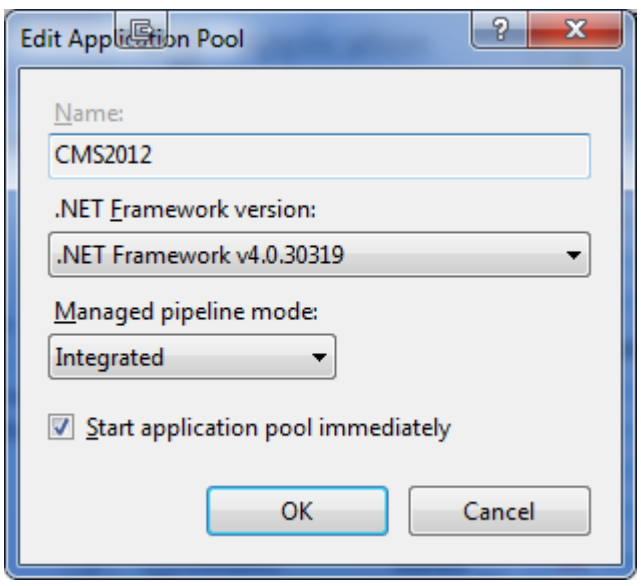
Create the site





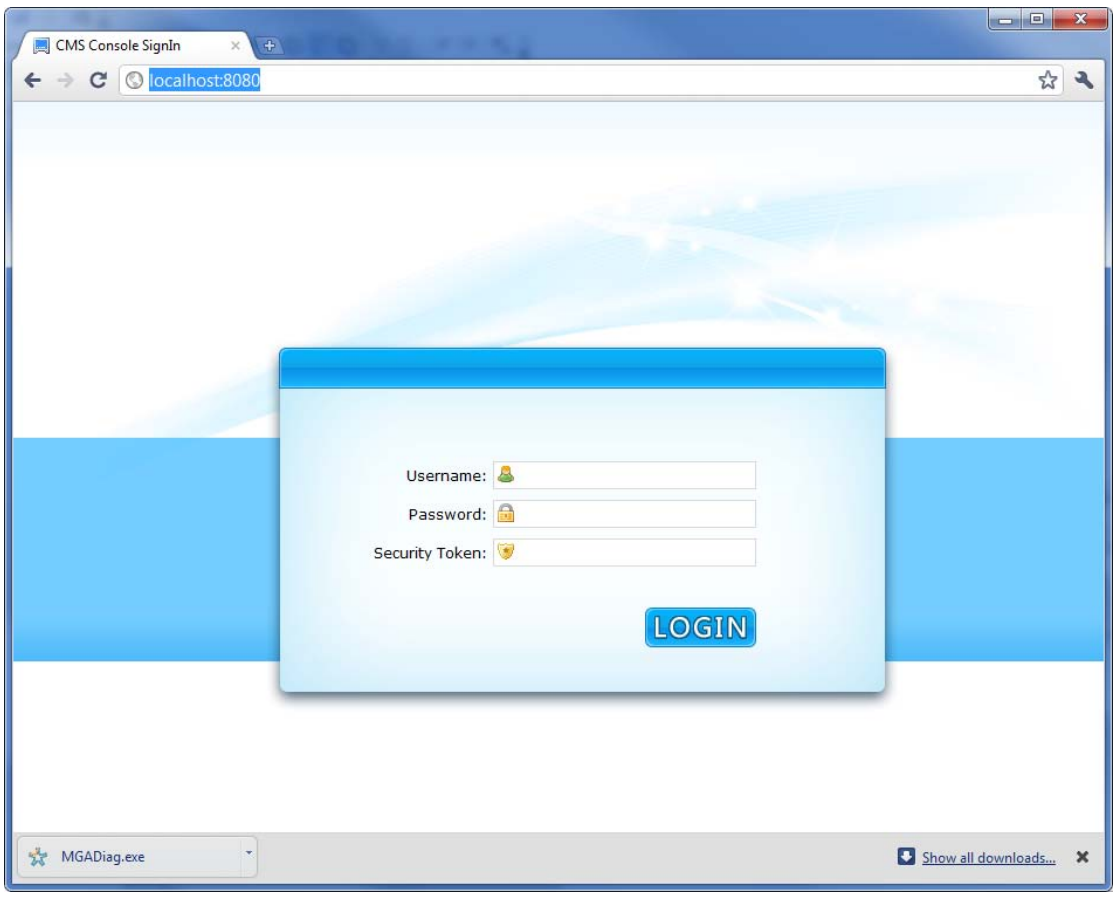


Please ensure ASP.Net 4.0 is enabled for the application pool



Grant READ + WRITE Permission of the web site directory to IIS_IUSRS and IUSR users

Then, you can try to access the CMS console site now.
http://localhost



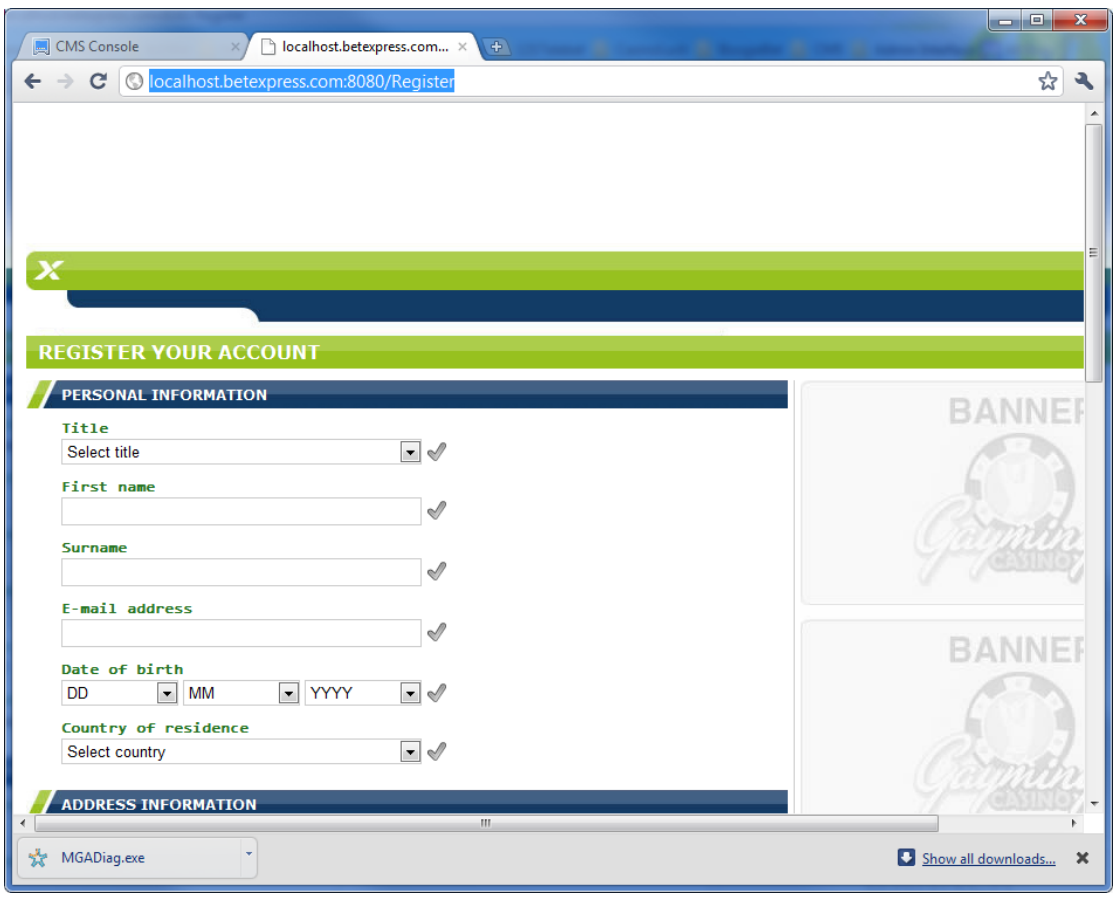
Try to login as “sa/asdfg”
If succeed, then configuration is correct for CMS Console site.

A sample of operator site

Add following entry to your %WinDir%\System32\drivers\etc\hosts file

127.0.0.1 localhost.betexpress.com

Then verify if it is accessible via
http://localhost.betexpress.com:8080/en/Register
http://localhost.betexpress.com:8080/he/Register



Development Notes

Standard HTML components

A set of standard HTML components
http://dev.gammatrix.com:2012/Tutorial/ShowView?viewName=StandardHTMLComponents

We need reuse the HTML components as much as possible
Also, we can add more the components if something we really needed.

You can find code sample and css sample there.

Validation

The new CMS extends the jQuery validation plugin for handy use.

You can find code samples at:

- /Views/Shared/Register/PersionalInformation.ascx
- /Views/Shared/Register/AccountInformation.ascx
- /Views/Shared/Register/AddressInformation.ascx
- /Views/Shared/Register/InputViewScript.ascx

Generally speaking, you need put a <ui:InputField> in the view.

```
<ui:InputField ID="fldAddress1" runat="server" ShowDefaultIndicator="true" BalloonArrowDirection="Left">
    <LabelPart>
        <%= this.GetMetadata(".Address1_Label").SafeHtmlEncode() %>
    </LabelPart>
    <ControlPart>
        <%: Html.TextBox( "address1", string.Empty, new
        {
           @id = "txtAddress1",
           @validator = ClientValidators.Create()
               .Required(this.GetMetadata(".Address1_Empty"))
               .MinLength( 2, this.GetMetadata(".Address_MinLength"))
        }
            ) %>
    </ControlPart>
</ui:InputField>
```

Validation rules are based on controls, you can add validation rules by adding the “**validator**” attribute to the control.
The rules are created by the **ClientValidators** class

After you added the rule, you need initialize the jQuery validation plugin in the client-side script.

```
$(document).ready(function () {
        // initialize the form validation
        $('#formRegister').initilizeForm();

        // bind an event to the submit button
        $('#btnRegisterUser').click(function (e) {
            e.preventDefault();
            // validate the form
            if (!$('#formRegister').valid())
                return;

            // if validation succeed, submit the form in AJAX
            // ……
        });
    });
```

The validation plugin supports complex validation as below.

```
<ui:InputField ID="fldUsername" runat="server" ShowDefaultIndicator="true" BalloonArrowDirection="Left">
    <LabelPart><%= this.GetMetadata(".Username_Label").SafeHtmlEncode() %></LabelPart>
    <ControlPart>
        <%: Html.TextBox( "username", string.Empty, new
        {
           @maxlength = 20,
            @id = "txtUsername",
           @validator = ClientValidators.Create()
               .Required(this.GetMetadata(".Username_Empty"))
               .MinLength(4, this.GetMetadata(".Username_Length"))
               .Custom("validateUsername")
               .Server(this.Url.RouteUrl("Register", new { @action = "VerifyUniqueUsername", @message =
this.GetMetadata(".Username_Exist") }))
        }
            ) %>
    </ControlPart>
</ui:InputField>
<script language="javascript" type="text/javascript">//<![CDATA[
   function validateUsername() {
       var value = this;
       var ret = /^\w+$/.exec(value);
       if (ret == null || ret.length == 0)
           return '<%= this.GetMetadata(".Username_Illegal").SafeJavascriptStringEncode() %>';
       return true;
   }//]]>
</script>
```

Custom: custom validation, indicates the client function name.
Server: server validation, here is the sample for server side:

```
        /// <summary>
        /// Verify the username is available
        /// </summary>
        /// <param name="username"></param>
        /// <param name="message"></param>
        /// <returns></returns>
        [HttpPost]
        public JsonResult VerifyUniqueUsername(string username, string message)
        {
            try
            {
                bool isExist = false;

                return this.Json(new
                {
                    @value = username,
                    @success = !isExist,
                    @error = isExist ? message : string.Empty
                });
            }
            catch (Exception ex)
            {
                Logger.Exception(ex);
                return this.Json(new
                {
                    @value = username,
                    @success = false,
                    @error = ex.Message
                });
            }
        }
```

Be aware of the differences and choose the correct method

ToLowerInvariant() ToUpperInvariant()	ToLower() ToUpper()
`string.Equals( a, b, StringComparison.InvariantCultureIgnoreCase)`	`string.Equals( a, b)`
`string.Format( CultureInfo.InvariantCulture, "{0}", arg1)`	`string.Format( "{0}", arg1)`
`StringBuilder sb = new StringBuilder();` `sb.AppendFormat(CultureInfo.InvariantCulture, "{0}", arg1);`	`StringBuilder sb = new StringBuilder();` `sb.AppendFormat( "{0}", arg1);`

`int a; string arg1;`
`int.Parse( arg1, CultureInfo.InvariantCulture)`
`int.TryParse( arg1, NumberStyles.Integer, CultureInfo.InvariantCulture, out a)`
(same for `decimal` / `double` / `long` / `DateTime` / etc)
	`int a; string arg1;`
	`int.Parse( arg1 )`
	`int.TryParse( arg1, out a)`

`decimal a = 103333.00M;`
`a.ToString(CultureInfo.InvariantCulture)`
(same for `int` / `double` / `long` / `DateTime` / etc)
	`decimal a = 103333.00M;`
	`a.ToString()`

`Regex.IsMatch(input, pattern, RegexOptions.IgnoreCase | RegexOptions.CultureInvariant)`	`Regex.IsMatch(input, pattern, RegexOptions.IgnoreCase)`
SafeHtmlEncode()	
SafeJavascriptStringEncode()	
HtmlEncodeSpecialCharacters()	

DIV+CSS layout basics

1. DIV+CSS layout must be compatible with IE7/8/9/10, Firefox, Chrome, Safari, Opera
2. When implementing frontend pages, keep native IE7(not simulated) opened along with Chrome for test
3. HTML must be Search-Engine-Friendly.
4. Avoid CSS hacker
5. CSS position knowledge
   a) `absolute` element is placed relative to the nearest parent with relative or absolute positioning.
   b) `relative` element is placed relative to the original position
   c) `fixed` element is placed relative to the web browser viewport.
   d) `float` element does not affect document flow.

DIV+CSS samples

Element with fixed width to be horizontal-center aligned within container

```
<div style="width:200px; margin:0 auto;">
</div>
```

Element with fixed width to be horizontal-center aligned within container (Another manner)

```
<div style="position:relative">
    <div style="width:200px; position:absolute; left:50%; margin-left:-100px;">
    </div>
</div>
```

Element with variable width to be horizontal-center aligned within container

```
<div style="width:100%;">
    <div style="float:left; left:50%; position:relative">
        <div style="float:left; left:-50%; position:relative">
            <!-- CONTENT START -->
            <img src="http://www.everymatrix.com/images/logo_em.png" />
            <!-- END -->
        </div>
    </div>
    <div style="clear:both"></div>
</div>
```

Element with variable height to be vertical-middle aligned within container

```
<div style="height:100%; #position: relative; display: table;">
    <div style="#position: absolute; #top:50%; display: table-cell; vertical-align: middle;">
        <div style="#position:relative; #top:-50%;">
            <!-- CONTENT START -->
            <img src="http://www.everymatrix.com/images/logo_em.png" />
            <!-- END -->
        </div>
    </div>
    <div style="clear:both"></div>
</div>
```