Chương 4: Thiết kế và cài đặt

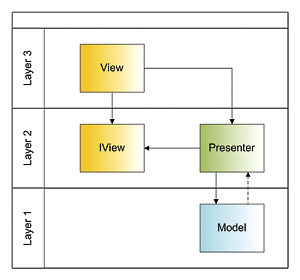
* 1. Nền tảng sử dụng
     1. **Giới thiệu mô hình xây dựng website theo kiến trúc MVP**

Mô hình MVP gồm ba thần phần chính View, Model và Presenter.

Model chứa dữ liệu và các tính toán xử lý logic để giải quyết vấn đề mà Website hướng tới.

View là thành phần đảm nhận việc thể hiện những dữ liệu của Model và là tổng hợp của các form, control được sử dụng.

Presenter là thành phần đảm nhận các xử lý thể hiện cũng như tương tác đến dữ liệu bên dưới và có thể tương tác để thay đổi View trong quá trình xử lý.



Trong mô hình MVP, các lớp View sẽ được sử dụng thông qua một Interface được định nghĩa trong .NET. Các lớp Presenter tương ứng sẽ sử dụng Interface này để đọc và ghi dữ liệu lên trên các View. Trong đa số các cách implement, một View sẽ có một Presenter tương ứng của nó. View sẽ khởi tạo Presenter cho nó và truyền cho Presenter này tham chiếu đến chính nó. Khi một event nào trên view được kích hoạt chẳng hạn như button\_clicked, text\_changed,.. bản thân lớp View sẽ không làm gì cả mà sẽ để cho lớp Presenter xử lý những sự kiện đó. Presenter sẽ đọc dữ liệu từ View (vì nó giữ một instance của View như là một member trong class) thông qua View Interface, thực hiện những xử lý ứng với Event được kích hoạt và set những thay đổi từ dữ liệu Model lên trên View thông qua View Interface.

Trong môi trường .NET, cùng một Presenter có thể được sử dụng cho View trên web như các trang ASP.NET hoặc được sử dụng cho các Form trong Windows Form Application. Các Presenter đọc và ghi dữ liệu thông qua một Interface của .NET nên nó hoàn toàn độc lập với layer View. Chính nhờ cách làm này mà ta có thể áp dụng Unit Test cho các lớp xử lý Presenter rất dễ dàng và nó cũng chính là một trong những lợi ích lớn nhất của MVP cho khả năng tái sử dụng.

**Các nhược điểm của MVP**

Có nhiều lớp, nhiều layer phải thực thi; có nghĩa độ phức tạp của ứng dụng sẽ tăng

Chúng ta phải nghĩ ra một cách để kết hợp giữa các Views và các Presenters. Trong ASP.NET thì các lớp Code-Behind sẽ implement một interface IView nào đó và sử dụng một member Presenter.

Mô hình Model không biết gì về Presenter. Vì thế nếu dữ liệu Model bị thay đổi từ một chương trình khác thay vì bởi Presenter thì phải nghĩ ra một cách để Presenter biết. Các bạn có thể implement chức năng này nếu cần bằng cách sử dụng các events, hoặc tham khảo thiết kế Observer Pattern.

* + 1. **Công cụ StructureMap**

StructureMap là công cụ thường được dùng cho các hệ thống lớn. Nhằm mục đích nâng cao tính linh hoạt, tính mở của hệ thống, khả năng tái sử dụng mã. StructureMap sẽ tham chiếu đến Interface của một lớp thay vì sử dụng chính lớp đó. Do vậy khi thay đổi một đối tượng ta chỉ cần tham chiếu đến interface của nó trong StructureMap. Chúng ta có thể sử dụng StructureMap khi hệ thống cần những yêu cầu như: khả năng mở rộng hệ thống, có một công cụ cấu hình chung, hệ thống sử dụng Test-Driven Development hoặc muốn test ứng dụng tự động…

* 1. **Quá trình thiết kế**
     1. Thiết kế database
        1. Mô hình đồ ER
* Account



* Friends



* Alert s, StatusUpdate, Blogs



* Coments, Tags, Ratings



* Group



* Messages



* Files, Folder



* + - 1. Mô hình quan hệ
* Account



* Friends



* Alert s, StatusUpdate, Blogs



* Coments, Tags, Ratings



* Group



* Messages



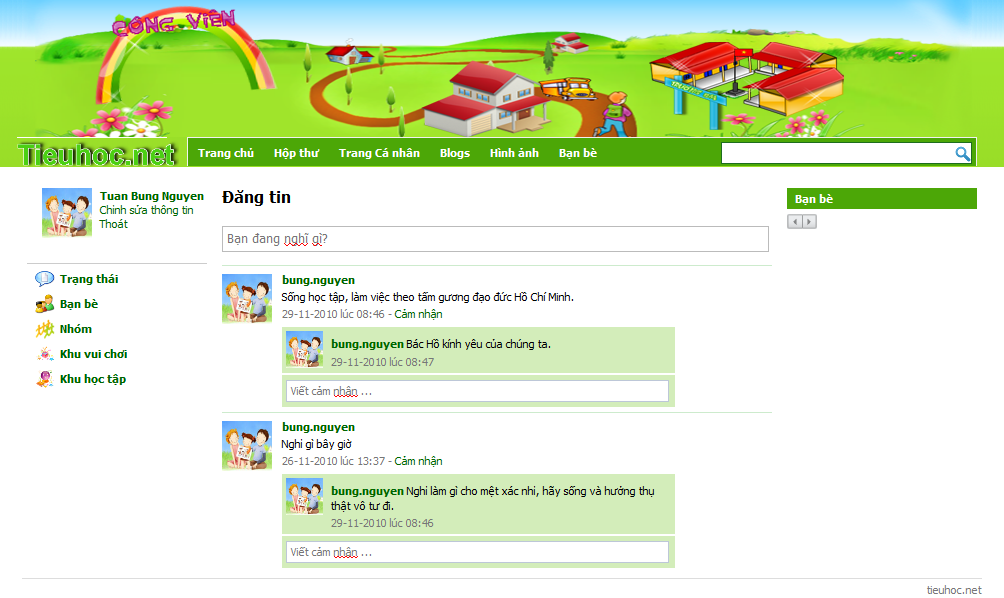
* Files, Folder



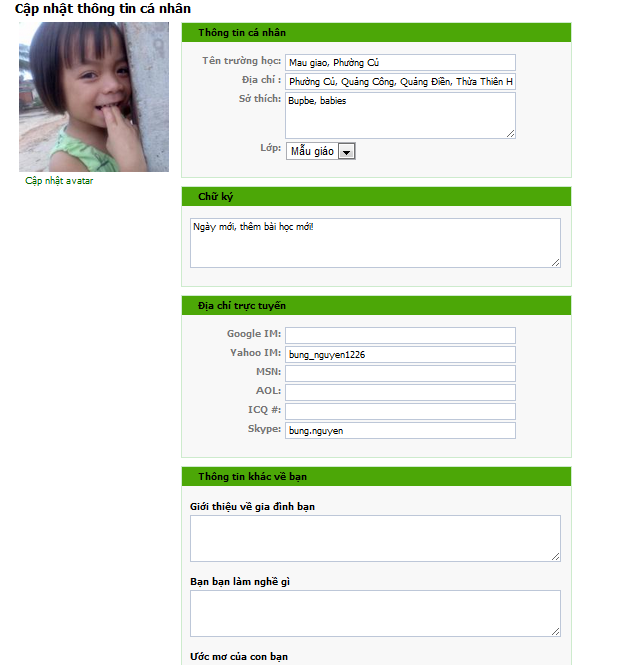
* + 1. **Thiết kế giao diện**
* Giao diện đăng ký, đăng nhập



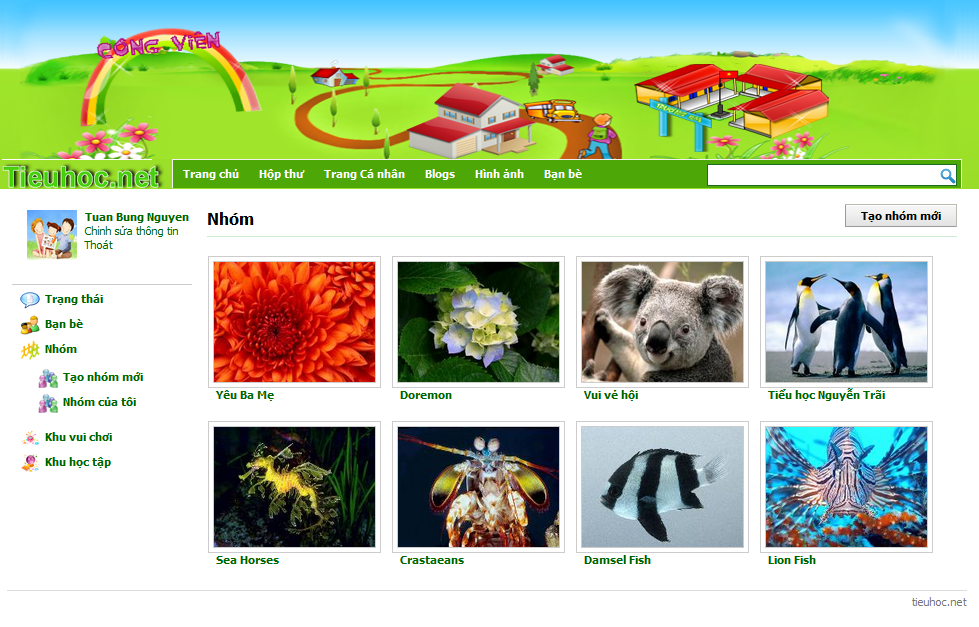
* Giao diện trang chủ



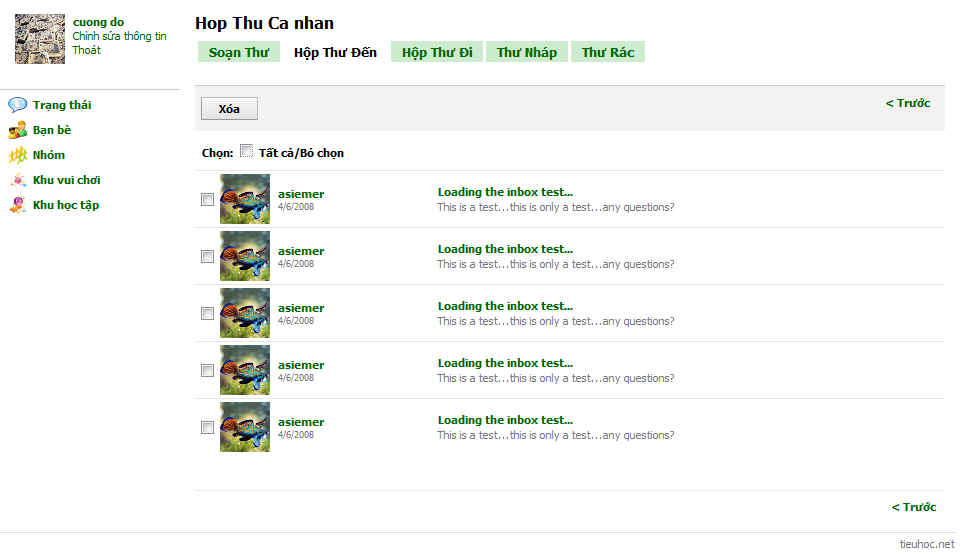
* Chỉnh sữa thông tin cá nhân



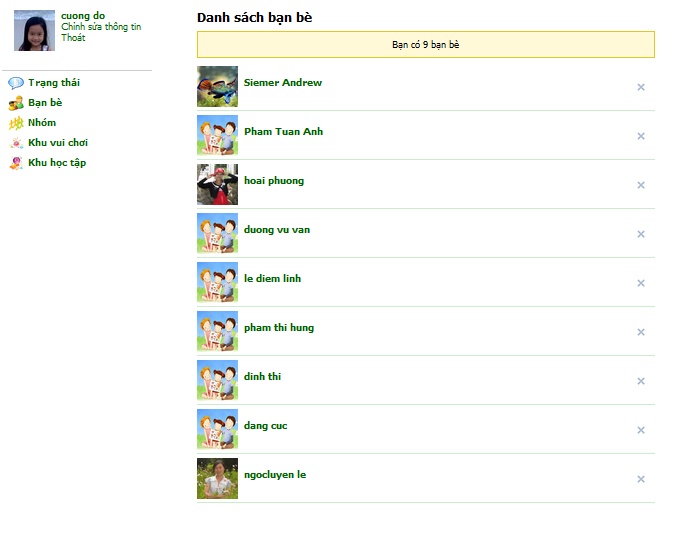
* Danh sách nhóm



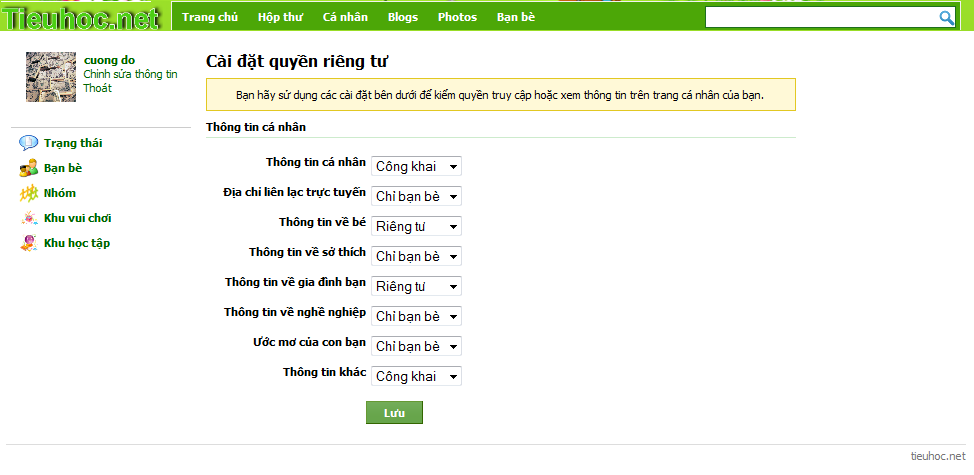
* Hộp thư cá nhân



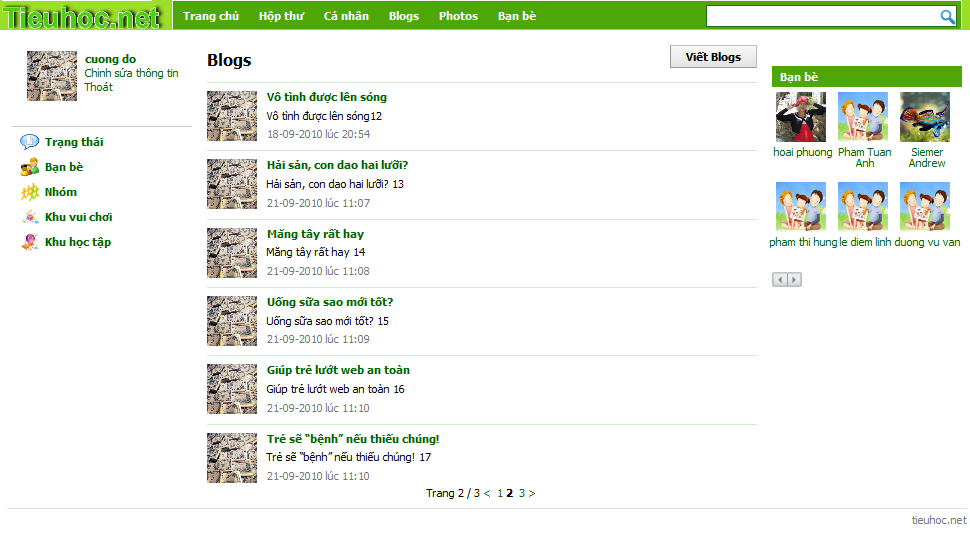
* Danh sách bạn bè



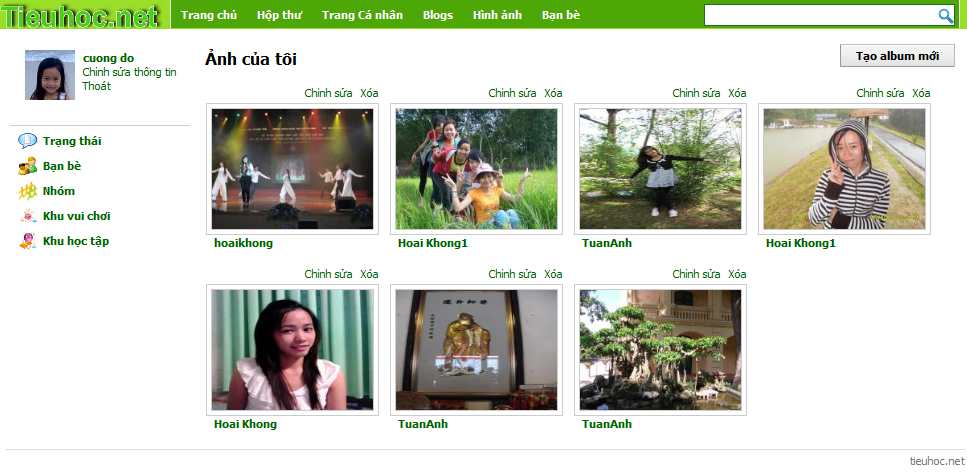
* Quyền riêng tư



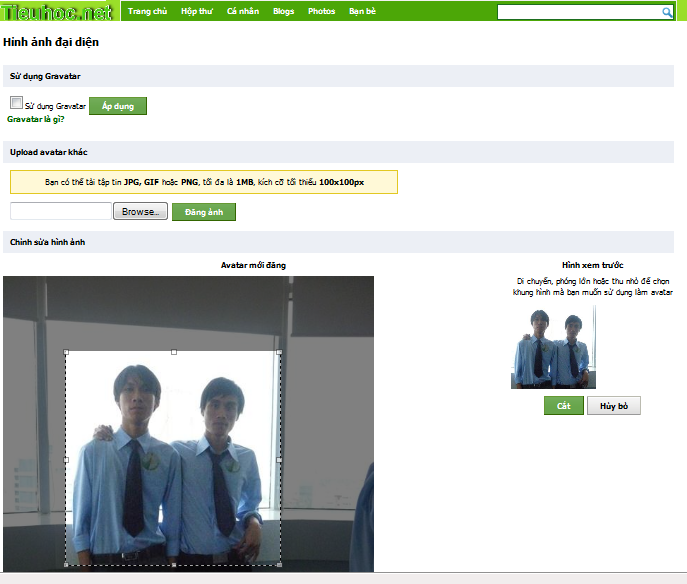
* Blogs cá nhân



* Hình ảnh cá nhân



* Chức năng cập nhật avatar



* + 1. **Thiết kế xử lý**
       1. Kiến trúc hệ thống

Data Access Layer

Business Layer

Presentation Layer

Service Layer

Database

* Tầng Presentation

Tầng này làm nhiệm vụ giao tiếp với người dùng cuối để thu thập dữ liệu và hiển thị kết quả hoặc dữ liệu thông qua các thành phần trong giao diện người dùng. Tầng này sử dụng các dịch vụ do tầng Business hoặc Service cung cấp.

* Tầng service

Tầng này chịu trách nhiệm liên kết các đối tượng các thành phần trong tầng Business với nhau. Tạo ra các dịch vụ để các thành phần trên tầng Presentation có thể sử dụng và hiển thị cho người dùng.

* Tầng Business

Tầng này thực hiện các nghiệp vụ chính của hệ thống, sử dụng các dịch vụ do tầng DataAcess cung cấp, và cung cấp các dịch vụ cho tầng Presentation và service. Tầng này cũng có thể sử dụng các dịch vụ của nhà cung cấp khác để thực hiện công việc của mình. Trong tầng này có các thành phần chính như là Business Entities và Business Component. Có thể xem tầng Service phía trên là một phần của tầng business.

* Tầng Data Access

Tầng này thực hiện các nghiệp vụ liên quan đến lưu trữ và truy xuất dữ liệu của ứng dụng, thường lớp này sẽ sử dụng các dịch vụ của các hệ quản trị cơ sở dữ liệu như SQL Server, Oracle, để thực hiện nhiệm vụ của mình.

* Cơ sở dữ liệu

Mọi dữ liệu mọi thông tin của người dùng sẽ được lưu xuống cơ sở dữ liệu. Trong chương trình sẽ sử dụng hệ quản trị cơ sở dữ liệu SQL Server của Microsoft.

* + - 1. Một số thuật toán trong chương trình

Cắt hình đại diện

Khi upload một hình lên trên server hệ thống cho phép người dùng cắt hình ảnh của mình theo nhưng chi tiết nhỏ hơn. Ở đây chúng ta sẽ sử dụng thư viện javascript để tìm vị trí cắt ảnh:

<script type="text/javascript" src="/js/cropper/lib/prototype.js"

language="javascript"></script>

<script type="text/javascript"

src="/js/cropper/lib/scriptaculous.js?

load=builder,dragdrop" language="javascript"></script>

<script type="text/javascript" src="/js/cropper/cropper.js"

language="javascript"></script>

Đoạn scritp chọn vị trí cắt ảnh ở client:

<script type="text/javascript" language="javascript">

Event.observe( window, 'load', function() {

new Cropper.ImgWithPreview(

'ctl00\_Content\_imgCropImage',

{

previewWrap: 'previewWrap',

minWidth: 100,

minHeight: 100,

ratioDim: {x: 100,y: 100},

displayOnInit: true,

onEndCrop: onEndCrop

}

);

} );

</script>

Chúng ta sẽ tìm dùng các trường ẩn để lưu lại các vị trí cần cắt của ảnh

<asp:HiddenField ID="hidX1" runat="server" />

<asp:HiddenField ID="hidY1" runat="server" />

<asp:HiddenField ID="hidX2" runat="server" />

<asp:HiddenField ID="hidY2" runat="server" />

<asp:HiddenField ID="hidWidth" runat="server" />

<asp:HiddenField ID="hidHeight" runat="server" />

Hàm javascript dưới đây sẽ tự động cập nhật vị trí của các điểm ảnh sẽ bị cắt khi người dùng thay đổi vùng chọn

<script type="text/javascript">

function onEndCrop( coords, dimensions )

{

$( 'ctl00\_Content\_hidX1' ).value = coords.x1;

$( 'ctl00\_Content\_hidY1' ).value = coords.y1;

$( 'ctl00\_Content\_hidX2' ).value = coords.x2;

$( 'ctl00\_Content\_hidY2' ).value = coords.y2;

$( 'ctl00\_Content\_hidWidth' ).value = dimensions.width;

$( 'ctl00\_Content\_hidHeight' ).value = dimensions.height;

}

</script>

Ảnh của người dùng chỉ được cắt khi lên server. Để thực hiện được điều này thì dưới client sẽ gửi lên server vị trí của ảnh cần cắt. Phương thức thực hiện việc này:

public void CropFile(Int32 X, Int32 Y, Int32 Width, Int32 Height)

{

byte[] imageBytes = profile.Avatar.ToArray();

using (MemoryStream ms = new MemoryStream(imageBytes, 0,

imageBytes.Length))

{

ms.Write(imageBytes, 0, imageBytes.Length);

System.Drawing.Image img =

System.Drawing.Image.FromStream(ms, true);

Bitmap bmpCropped = new Bitmap(200, 200);

Graphics g = Graphics.FromImage(bmpCropped);

Rectangle rectDestination = new Rectangle(0, 0,

bmpCropped.Width, bmpCropped.Height);

Rectangle rectCropArea = new Rectangle(X,Y,Width,Height);

g.DrawImage(img, rectDestination, rectCropArea,

GraphicsUnit.Pixel);

g.Dispose();

MemoryStream stream = new MemoryStream();

bmpCropped.Save(stream,

System.Drawing.Imaging.ImageFormat.Jpeg);

Byte[] bytes = stream.ToArray();

profile.Avatar = bytes;

\_profileRepository.SaveProfile(profile);

}

\_view.ShowApprovePanel();

}

Ghi đè liên kết ảo

Nhằm mục đích tạo cho người dùng có môt địa chỉ trên trên website ví dụ như <http://tieuhoc.net/tenguoidung> hoặc muốn tạo ra một liên kết có ý nghĩa với người dùng, hoặc muốn giấu liên kết. Để thực hiện điều này chúng ta phải ghi đè liên kết.

Chúng ta sẽ phải tạo ra một đối tượng UrlRewrite kế thừa đối tượng HttpModule. Module HTTP sẽ thực thi xử lý HTTP. Module này có thay đổi được các yêu cầu được gửi đến từ người dùng.

Đăng ký xử lý ghi đè liên kết vào trong phương thức init của HttpModule

public void Init(HttpApplication application)

{

//let's register our event handler

application.PostResolveRequestCache +=

(new EventHandler(this.Application\_OnAfterProcess));

}

Trong phương thức xử lý ghi đè liên kết AppApplication\_OnAfterProcess

Phương thức này sẽ request liên kết từ client gửi đến nếu nó không phải là đường dẫn tồn tại thì phương thức này sẽ dựa vào thông tin trên liên kết đó để có thể xử lý ghi đè đến liên kết thật trên thực tế.

Nếu liên kết có dạng <http://tieuho.net/blogs/tieu-de-bai-viet.aspx> thì nó sẽ được xử lý như sau từ thông tin trang trên liên kết ta sẽ lấy được nội dung blog theo thông tin “tieu-de-bai-viet”. Từ đó hệ thống thực thi liên kết thực sự của nó là <http://tieuho.net/blogs/ViewPost.aspx?BlogID=1> giả sử như ID của blog bên trên là 1. Cách thức xử lý như vậy sẽ tương tự cho Tags, Group, Profile

if (!System.IO.File.Exists(application.Request.PhysicalPath))

{

#region BLOGS

if (application.Request.PhysicalPath.ToLower().Contains("blogs"))

{

string[] arr = application.Request.PhysicalPath.ToLower().Split('\\');

string blogPageName = arr[arr.Length - 1];

string blogUserName = arr[arr.Length - 2];

blogPageName = blogPageName.Replace(".aspx", "");

if (blogPageName.ToLower() != "profileimage" && blogUserName.ToLower() != "profileavatar")

{

if (blogPageName == "default")return;

Account acc = Account.GetAccountByUsername(blogUserName);

if (acc == null)return;

Blog blog = Blog.GetBlogByPageName(blogPageName, acc.AccountID);

context.RewritePath("~/blogs/ViewPost.aspx?BlogID=" + blog.BlogID.ToString());

}

else {return;}

}

#endregion

#region GROUPS

else if (application.Request.PhysicalPath.ToLower().Contains("groups") && \_WebContext.GroupID == 0)

{

string[] arr = application.Request.PhysicalPath.ToLower().Split('\\');

string groupPageName = arr[arr.Length - 1];

groupPageName = groupPageName.Replace(".aspx", "");

Group group = Group.GetGroupByPageName(groupPageName);

context.RewritePath("~/groups/viewgroup.aspx?GroupID=" + group.GroupID.ToString());

}

#endregion

#region TAGS

else if (application.Request.PhysicalPath.ToLower().Contains("tags"))

{

Tag tag = null;

int tagsPosition = 0;

string tagName;

string[] arr = application.Request.PhysicalPath.ToLowerInvariant().Split('\\');

for (int i = 0; i < arr.Length; i++)

{

if (arr[i].ToLower() == "tags")

{

tagsPosition = i;

}

if (tagsPosition > 0)

{

tagName = arr[i + 1];

tag = Tag.GetTagByName(tagName.Replace("-", " "));

break;

}

}

if (tag != null)

{

context.RewritePath("~/tags/tags.aspx?TagID=" + tag.TagID);

}

}

#endregion

#region PROFILES

else if (CoreSupport.StringCount(application.Request.Path, '/') <= 1)

{

string username = application.Request.Path.Replace("/", "");

Account account = Account.GetAccountByUsername(username);

if (account != null)

{

string UserURL = "~/Profiles/profile.aspx?AccountID=" + account.AccountID.ToString();

context.Response.Redirect(UserURL);

}

else

{

context.Response.Redirect("~/PageNotFound.aspx");

}

}

else

{ }

#endregion

Thông báo thông tin mới nhất của mình cho bạn bè

Khi người dùng đăng ảnh, viết blogs hoặc đăng trang thái lên thì những thông tin này sẽ được tự động thông báo đến bạn bè của họ.

public void AddNewBlogPostAlert(Blog blog)

{

Init();

alert = new Alert();

alert.AccountID = account.AccountID;

alert.AlertTypeID = (int)AlertType.AlertTypes.NewBlogPost;

alertMessage =

GetProfileImage(account.AccountID)

+ "<div class='title'><h2>" + GetProfileUrl(\_userSession.CurrentUser.Username) + "</h2></div>"

+ " đã viết blog: <b><a href='[rootUrl]blogs/" + \_webContext.Username + "/" + blog.PageName + ".aspx'>"

+ blog.Title + "</a></b>";

alert.Message = alertMessage;

SaveAlert(alert);

SendAlertToFriends(alert);

}

Kết bạn

Khi người dùng muốn kết bạn họ sẽ phải biết được thông tin của người đó như Email, hoặc tên đăng nhập và nhấn vào tìm kiếm. nếu người này đã tồn tại trong hệ thống thì hệ thống sẽ hiện thị ra cho người dùng. Người dùng nhấp chuột vào liên kết bạn để gửi yêu cầu kết bạn đến người này.

Phương thức SendInvitations dưới đây sẽ tạo ra một thông điệp từ yêu cầu kết bạn của người dùng thông điệp này có kèm theo một khóa chứng thực. Thông điệp này sẽ được gửi vào hộp thư của người bạn muốn kết bạn và gửi đến địa chỉ email của họ.

public string SendInvitations(Account sender, string ToEmailArray, string Message)

{

string resultMessage = Message;

foreach (string s in ToEmailArray.Split(new char[] { ',', ';' }))

{

FriendInvitation friendInvitation = new FriendInvitation();

friendInvitation.AccountID = sender.AccountID;

friendInvitation.Email = s;

friendInvitation.GUID = Guid.NewGuid();

friendInvitation.BecameAccountID = 0;

FriendInvitation.SaveFriendInvitation(friendInvitation);

Account account = Account.GetAccountByEmail(s);

if (account != null)

{

\_alertService.AddFriendRequestAlert(\_userSession.CurrentUser, account, friendInvitation.GUID,

Message);

}

SendFriendInvitation(s, sender.FirstName, sender.LastName, friendInvitation.GUID.ToString(), Message);

Message = sender.FirstName + " " + sender.LastName +

"Muốn kết bạn với bạn!<HR><a href=\"" + \_configuration.RootURL +

"Friends/ConfirmFriendInSite.aspx?InvitationKey=" + friendInvitation.GUID.ToString() + "\">" + \_configuration.RootURL +

"Friends/ConfirmFriendInSite.aspx?InvitationKey=" + friendInvitation.GUID.ToString() + "</a><HR>" + Message;

Messages m = new Messages();

m.Body = Message;

m.Subject = "Thư mời kết bạn";

m.CreateDate = DateTime.Now;

m.MessageTypeID = (int)MessageTypes.FriendRequest;

m.SentByAccountID = \_userSession.CurrentUser.AccountID;

m.MessageID = 0;

m.Save();

Int64 messageID = m.MessageID;

MessageRecipient sendermr = new MessageRecipient();

sendermr.AccountID = \_userSession.CurrentUser.AccountID;

sendermr.MessageFolderID = (int)MessageFolders.Sent;

sendermr.MessageRecipientTypeID = (int)MessageRecipientTypes.TO;

sendermr.MessageID = messageID;

sendermr.MessageStatusTypeID = (int)MessageStatusTypes.Unread;

sendermr.MessageRecipientID = 0;

sendermr.Save();

Account toAccount = Account.GetAccountByEmail(s);

if (toAccount != null)

{

MessageRecipient mr = new MessageRecipient();

mr.AccountID = toAccount.AccountID;

mr.MessageFolderID = (int)MessageFolders.Inbox;

mr.MessageID = messageID;

mr.MessageRecipientTypeID = (int)MessageRecipientTypes.TO;

mr.MessageRecipientID = 0;

mr.MessageStatusTypeID = 1;

mr.Save();

}

resultMessage += "• " + s + "<BR>";

}

return resultMessage;

}

Khi người bạn của bạn đọc được thông điệp này và nhấn vào liên kết được gửi đến thì người đó sẽ vào danh sách bạn bè của bạn và sẽ được thông báo cho mọi người biết.

* 1. Quá trình cài đặt