# Fashion Ade Code Review Session 1

Renombrar la carpeta BuildYourCloser por BuildYourCloset.

Renombar BuildYourClosetController.cs por FlavorSelectController.cs

Mover el Quiz dentro de esta carpeta y renombrar a FlavorQuizController.cs

Cambiar el BuildYourClosetBaseController.cs por un atributo de NotAuthorizeLoggedUser. Ver: http://schotime.net/blog/index.php/2009/02/17/custom-authorization-with-aspnet-mvc/

Todos los controlladores deben heredar de BaseController y aprovechar las propiedades disponibles en el mismo.

**Eliminar el uso de SESSION en todo el sitio, consultar en caso de que crean que es IMPOSIBLE hacerlo sin SESSION. El orden debe ser siempre INPUT HIDDEN (imposible si es algo genérico para todo el site), COOKIE (imposible si es mucha data o tiene información personal/critica) y SESSION (solo se usa si las dos anteriores no alcanzaron y si no tenemos la posibilidad de saber que usuario es, sino usar DB).**

**Simplificar el uso de VIEWDATA. Siempre que sea posible usar vistas tipeadas, si hace falta usar el ViewData, tratemos de usar objetos de comunicación, no hardcodeemos tanto!!!**

Quitar todo uso de FormCollection values en los parámetros SIEMPRE que sea posible, genera poca claridad en el código, leer sobre ModelBinder si hace falta ir mas alla de los defaults provistos.

No me parece optimo la forma de manejar la información entre los pasos, quitar session. Usar hidden fields de Build Your Closet.

No me parece claro que la misma acción sea llamada de dos botones. Por ej: FlavorWeightController.SetWeight se llama desde Next y Previous, me parece mas lógico dos Acciones que sean Next y Previous; y diferenciar la lógica por la acción no por un nombre que puede cambiar y esta hardcodeado. Siempre hacer 2 acciones para estos casos.

Porque guardan estas cosas en session: Session["previousUrl"] = "FlavorWeight";???

La lógica del Quiz esta en el controlador de Fashion Flavor Select, pasar al Quiz.

**No poder TEXTO escrito en el código. Usar resources, investigar como es la mejor forma, pero no se puede poner texto escrito por todos lados, debería ser tan simple como buscar en el resource donde esta este texto y poder cambiarlo. Lo mismo con los mails, NO poner el HTML en el código, usar un resource o un archivo html (si hace falta).**

No dejar METODOS ni código comentado que NO SE USA mas, hay que limpiar cuando se cambian o corrigen cosas.

**Comentar TODOS los métodos públicos o privados, incluyendo sus parámetros y excepciones.**

Separar Login, Logout y Forgot Password en 3 controladores en la subcarpeta Account.  
AccountController debe ser ProfileController dentro de esta carpeta. Separar los Friends del AccountController y todo el tema de Contactos.

RatingController tiene 4 metodos que empiezan con Index()? Index no es nada, no es una acción, no es funcionalidad. SER CLARO, como puede otra persona entender esto?

Ver como conviene separar MyOutfitsController.cs.  
  
Cambiar RatingController.cs por InvitationResponseController.cs o similar.

Borrar StyleAlertsController.cs

El método LoadSeasons parece usar de varios lados, mover a Common.

No hardcodear PATHS, por ej: "/res/Garments/".

**Separar claramente en carpetas las funcionalidades y clases relacionadas. No poner las clases todas en MVCInteraction porque si, no se entiende quien las usa y ninguna debería ser publica me parece, quizás protected. Borrar carpeta MVCInteraction. Por ej.**

**Namespace FashionAde.Web.Controllers.AddGarment /Carpeta AddGarment**

**UploadGarmentController.cs**

**Coords.cs**

**Namespace FashionAde.Web.Controllers.Outfits /Carpeta Outfits**

**MyOutfitsController.cs**

**OutfitNotate.cs**

**OutfitSearch.cs**

RESPETAR LA NOMENCLATURA Y COMENTAR METODOS, VOY A SER ESTRICTO CON ESTE TEMA.  
HACE MUCHO TIEMPO QUE VENIMOS TRABAJANDO EN ESTO Y NO SE RESPETA.

A modo de ejemplo:

public RedirectToRouteResult PasswordRecovery(string UserName, string Answer)

{

MembershipUser mu = Membership.GetUser(UserName);

try

{

string newpassword = mu.ResetPassword(Answer);

SmtpMailing mail = new SmtpMailing();

string body = newpassword + "<br /><br /><h2 style='font-family:Century Gothic; color:#4D679B; font-weight:bold; font-size:14px; margin:0 0 0 30px; padding-bottom:2px;'>Please use it to login and change it in PROFILE page.</h2>";

mail.SendMail(mu.Email, "Password Reset", "Your new password is", body, false);

Session["Errors"] = "Your new password has been sent by email, please check your inbox.";

return RedirectToAction("Index");

}

catch (Exception e)

{

Session["Errors"] = e.Message;

}

return RedirectToAction("ForgotPassword");

}

* Un método es una acción. Debería ser RecoverPassword.
* Las parámetros empiezan con minúsculas.
* Se pone HTML en el código como si nada. Todo el texto esta harcodeado.
* **Se usa Session para mostrar un ERRROR!!!**
* Lo que es peor se catchea cualquier excepción y se la muestra al usuario con una SESSION ¡!!!! O sea… si da error de mail le vas a mostrar la excepción del sistema al usuario? ES POCO SERIO.
* No se valida que pueda ser NULL el resultado de Membership.
* No esta comentando en lo absoluto.

public ActionResult Index()

{

LoadSeasons();

IList<Category> categories = categoryRepository.GetAll();

foreach (Category category in categories)

ViewData["Category" + category.Description] = category.Id;

List<WebClosetGarment> webClosetGarments = closetRepository.GetWebClosetGarments(this.ProxyLoggedUser);

ViewData["pants\_jeans"] = webClosetGarments.FindAll(delegate(WebClosetGarment record)

{

if (record.CatId == (int)ViewData["CategoryPants"] || record.CatId == (int)ViewData["CategoryJeans"])

{

return true;

}

return false;

});

ViewData["skirts\_shorts"] = webClosetGarments.FindAll(delegate(WebClosetGarment record)

{

if (record.CatId == (int)ViewData["CategoryShorts"] || record.CatId == (int)ViewData["CategorySkirts"])

{

return true;

}

return false;

});

ViewData["dresses"] = webClosetGarments.FindAll(delegate(WebClosetGarment record)

{

if (record.CatId == (int)ViewData["CategoryDresses"])

{

return true;

}

return false;

});

ViewData["jackets"] = webClosetGarments.FindAll(delegate(WebClosetGarment record)

{

if (record.CatId == (int)ViewData["CategoryJackets"])

{

return true;

}

return false;

});

ViewData["tops"] = webClosetGarments.FindAll(delegate(WebClosetGarment record)

{

if (record.CatId == (int)ViewData["CategoryCoats"] || record.CatId == (int)ViewData["CategoryCardigan"] || record.CatId == (int)ViewData["CategoryShirts"])

{

return true;

}

return false;

});

ViewData["accesories"] = webClosetGarments.FindAll(delegate(WebClosetGarment record)

{

if (record.CatId == (int)ViewData["CategoryJewelry"] || record.CatId == (int)ViewData["CategoryBelts"] || record.CatId == (int)ViewData["CategoryShoes"] || record.CatId == (int)ViewData["CategoryBags"])

{

return true;

}

return false;

});

return View();

}

* Les parece que esta es la mejor forma de resolver el problema? No veo que se halla pensando un poco mas alla. Y si cambiamos la categorización? Se podría mejorar esto? Les parece optimo esto?
* Les parece que teniendo un sistema que compila y podemos detectar errores en tiempo de compilación, dependamos tanto de ViewData[“”] con strings. El 95% del código debería usar clases de comunicación.

[AcceptVerbs(HttpVerbs.Get)]

public ActionResult Outfits()

{

GetOutfitsInfo();

RegisteredUser user = this.ProxyLoggedUser;

int totalCount;

int season = 4;

ViewData["Outfits"] = closetOutfitViewRepository.Search(out totalCount, 10, 1, 1, user.Id, string.Empty, "1", user.Closet);

ViewData["TotalOutfits"] = totalCount;

ViewData["Pages"] = Common.Paging(totalCount, 1, 10, 10);

ViewData["PrivacyLevel"] = user.Closet.PrivacyLevel.ToString();

if (DateTime.Today >= new DateTime(DateTime.Today.Year, 3, 21))

season = 1;

if (DateTime.Today >= new DateTime(DateTime.Today.Year, 6, 21))

season = 2;

if (DateTime.Today >= new DateTime(DateTime.Today.Year, 9, 21))

season = 3;

if (DateTime.Today >= new DateTime(DateTime.Today.Year, 12, 21))

season = 4;

ViewData["Season"] = season;

return View(user.Closet);

}

* Porque tengo que descifrar el código de que hace con el season? No se piensa en la otra persona que tenga que arreglar un problema en este método?
* Season = 1? No hay una enumeración?
* Si hay tantos ViewData[] no conviene crear un objeto de comunicación?

try

{

using (Image OriginalImage = Image.FromFile(Server.MapPath(coords.ImageUri)))

{

using (Bitmap bmp = new Bitmap(Width, Height))

{

bmp.SetResolution(OriginalImage.HorizontalResolution, OriginalImage.VerticalResolution);

using (Graphics Graphic = Graphics.FromImage(bmp))

{

Graphic.SmoothingMode = SmoothingMode.AntiAlias;

Graphic.InterpolationMode = InterpolationMode.HighQualityBicubic;

Graphic.PixelOffsetMode = PixelOffsetMode.HighQuality;

Graphic.DrawImage(OriginalImage, new Rectangle(0, 0, Width, Height), X, Y, Width, Height, GraphicsUnit.Pixel);

MemoryStream ms = new MemoryStream();

bmp.Save(ms, OriginalImage.RawFormat);

byte[] CropImage = ms.GetBuffer();

using (MemoryStream mstream = new MemoryStream(CropImage, 0, CropImage.Length))

{

mstream.Write(CropImage, 0, CropImage.Length);

using (Image CroppedImage = Image.FromStream(mstream, true))

{

string folderPath = Server.MapPath("/res/Garments/");

string originalPath = Server.MapPath(coords.ImageUri);

string smallPath = Path.Combine(folderPath, "small\_" + Path.GetFileName(coords.ImageUri));

string bigPath = Path.Combine(folderPath, "big\_" + Path.GetFileName(coords.ImageUri));

OriginalImage.Dispose();

CroppedImage.Save(originalPath, CroppedImage.RawFormat);

ResizeImage(originalPath, bigPath, 135, 100, true); //Imagenes Grandes

ResizeImage(originalPath, smallPath, 65, 65, true); //Imagenes Grandes

}

}

}

}

}

return Json(new { Success = true });

}

catch

{

return Json(new { Success = false });

}

* Try/Catch por cualquier cosa? Nunca usar catch sin el tipo de excepción que se espera!
* Si fuera el caso que sea asi cualquier cosa, peor aun ya que no se logeara nunca asi que jamás sabremos que paso…. Y solo nos entederamos cuando el usuario putee….
* Que alguien me explique que hace todo esto mirándolo rápidamente…

Al ver CIERTOS códigos (como estos, no todos) no veo DEDICACION, no veo PENSAR, no veo PROFESIONALISMO. Deben mejorar en este aspecto, no todo es HACER más en menos tiempo, y normalmente lo primero que sale puede no ser lo mejor. piense en factorizar y revisar el codigo que generan, como mejorarlo. hagan las pruebas mientras crean codigo, ayudense a mejorar su codigo.

## Documento de Interes

Leer este articulo: <http://codeclimber.net.nz/archive/2009/04/08/13-asp.net-mvc-extensibility-points-you-have-to-know.aspx>