PROJET .NET POUR LE WEB

Benjamin de Vaublanc
Dorian Boussac
Wiston Cherry
GROUPE EB1

Les Models

Post attributs

- int PostId
- int TagId
- int AuthorId
- string Title
- DateTime Date
- string Content
- Tag Tag
- Author Author
- bool Online

Page attributs

- int PageId
- int AuthorId
- string Title
- DateTime Date
- string Content
- Author Author
- bool Online

Tag attributs

- Int TagId
- String Name
- virtual ICollection<Post> Posts

Auhor attributs

- int AuthorId
- string Name
- List<Post> Posts

Message

- int MessageId
- string Name
- string Site
- string Title
- string content
- bool statement

AccountModels

Class ChangePassword

- string OldPassword
- string NewPassword
- string ConfirmPassword

Class LogOnModel

- string UserName
- string Password
- bool RememberMe

Class RegisterModel

- string UserName
- string Email
- string Password
- string ConfirmPAssword

CmsEntities : DbContext

- DbSet<Key> Keys
- DbSet<Post> Posts
- DbSet<Page> Pages
- DbSet<Message> Messages

SampleData

Pour pouvoir remplir la base de donnée avec des données fictives, lorsque la base de donnée se réinitialise, Cf

```
protected void Application_Start()
        {
            System.Data.Entity.Database.SetInitializer(newMvcCms.Models.SampleData());
```

Les Controllers

Pour chaque entité nous avons décidé de créer deux Controller

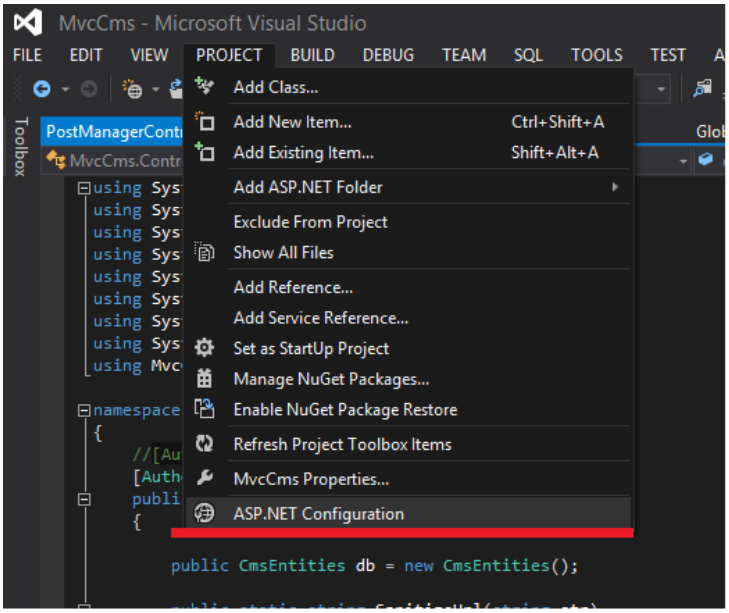
Un controller détenant les actions publiques à tous les visiteurs et un autre destiné aux actions privées pour les utilisateurs et administrateur, par convention ce dernier comportera la mention Manager

EX : Model Post

PostController
(contenant les action publique du site)
PostManagerController
(contenant les action réservé aux membres et administrateur)

Certaine classe et actions seront précédé de la mention **[Authorize]** afin d'être accessible seulement par des membres du site. D'autres requiert des droits Administrateur, ces dernières seront précédé.
par un **[Authorize(Roles = « Administrator »)]**.

Rappel : les rôles on été géré à l'aide de l'interface ASP.NET Configuration.
Pour accéder aux actions de l'administrateur. Il faut créer un compte admin avec un rôle Administrator



PostController

Actions

- Index return view()
  Permet de lister tous les posts
- Abstract(int limit) return view()
  Permet de récupérer les derniers posts par ordre anté-chronologique en fonction de la limit paramétré
- Navigation (int tag, string title) return view()
  Permet de récupérer les posts ou tag = tagId, le deuxième paramètre title nous servira plus tard pour le formatage de la route.
- Details(int id) return view(post)
  Permet d'accéder au detail d'un post

PostManagerController [Authorize]

Actions

- Index return view()
  Permet de lister les posts
- Details(int id) return view()
  Permet d'accéder au detail d'un posts
- Create(Post) return view(Post)
  Permet de créer un posts
- Edit (Post) return view(Post)
  Permet d'éditer un post
- Delete(Post) return view (Post)
  Permet de supprimer un post
- DeleteConfirmed(int id) return RedirectToAction("Index")
  Permet de confirmer la supression de l'index avant de rediriger à la list des posts

PageController

Actions

- GetMenuPages() return View
  Permet de pouvoir lister les pages avec le statut online afin de pouvoir créer le menu dans le header
- Details(int id) return view(page)
  Permet d'accéder au detail d'une page

PageManagerController [Authorize]

Actions

- Index return view()
  Permet de lister les pages
- Details(int id) return view()
  Permet d'accéder au detail d'une page
- Create(Page) return view(Page)
  Permet de créer une page
- Edit (Page) return view(Page)
  Permet d'éditer une page
- Delete(Page) return view (Page)
  Permet de supprimer une page
- DeleteConfirmed(int id) return RedirectToAction("Index")
  Permet de confirmer la suppression d'une page

TagManagerController [Authorize]

Actions

- Index return view()
  Permet de lister les tag
- Create(Post) return view(Post)
  Permet de créer un nouveau tag
- Edit (Post) return view(Post)
  Permet d'éditer une tag
- Delete(Post) return view (Post)
  Permet de supprimer un tag
- DeleteConfirmed(int id) return RedirectToAction("Index")
  Permet de confirmer la suppression d'un tag

MessageController

Actions

- Index(Message) return RedirectToAction(Success)
  Permet de laisser un message aux members du site
- Success(int id) return view(recap)
  Confirmation de l'envoi du message, avec un récapitulatif de ce dernier

MessageManagerController [Authorize]

Actions

- Index return view()
  Permet de lister tous les message envoyé par les visiteurs
- Details(int id) return view()
  Permet d'accéder au detail du message
- CounterMessage retrun int
  Permet de compter le nombre de message dans la boite de reception
- Delete(Post) return view (Message)
  Pemet de supprimer un message (dans notre application cette action est exécuté en ajax à l'aide du helper razor)
- Update(int id)
  Permet de changer le statu du message, pour qu'il soit marqué comme lu

AccountController

Actions

- LogOn(LogOnModel model, string returnUrl)return view(model)
  Permet de se connecter et de rediriger sur la page que l'utilisateur avait appelé avant d'être redirigé sur le formulaire d'authentification.
- LogOff() return RedirectToAction("index","hom")
  Permet à l'utilisateur de se déconnecter, puis il est redirigé sur la page d'acceuil
- Register(RegisterModel model) return view(model)
  Permet au visiteur de créer un compte en vérifiant que les identifiant et email ne sont pas déjà utilisé par un autre compte
- ChangePassword(ChangePasswordModel model) return View(model)
  Permet à l'utilisateur de changer de mot de passe, en cas d'oubli
  ChangePasswordSuccess() return view()
  Permet de confirmer que le password à bien été changé
- getUsers() return view(users) **[Authorize(Roles = « Administrator »)]**.
  Permet de lister tous les utilisateurs

BDD

Tables

- Applications
- Author
- Tags
- PostsInTags
- Posts
- Profiles
- Users
- UsersInRoles
- Roles
- Memberships
- Messages

Routing

- "/" : root, page d'accueil du site affichant les derniers posts
- "/blog/{id}/{titre}" : page du blog identifiée par {id}
- "/blog/tags/{tag}" : liste de posts identifiés par l'étiquette {tag}
- "/blog/archives/{annee}/{mois}" : liste de posts correspondant à {annee} et {mois}
- "/pages/{id}/{titre}" : page du site indépendante du blog définie par son identifiant et titre
- "/contact/message": affiche le formulaire pour laisser le message ainsi que la validation
- "/contact/confirm": affiche la page de confirmation du message
- "/Logon" : page d'autentification et d'affichage d'une erreur de login. En cas de succès
- rediriger vers la page d'origine

Design

Utilisation du boostrap de twitter afind d'obtenir rapidement un rendu agréable visuellement et intuitif pour la navigation.
http://twitter.github.com/bootstrap/

Librairies et framework Javascript

jQuery UI
Utilisation de Jquery UI pour l'interface utilisateur afin d'obtenir des plugins tel que des date Picker et autre élements utiles à l'interface
http://jqueryui.com/

jQuery
Utilisation de jQuery pour la simplification et l'exécution de code javascript

TinyMce
Utilisation de ce plugin de wysiwyg afin de pouvoir mettre en forme les actualités et les pages directement par le backoffice. Le WYSIWYG devra être configurer de manière restrictive afin que l'utilisateur ne puisse pas entraver le Design du site

http://www.tinymce.com/

Hébergement

Le Projet sera visionné sur un git. Et sera accessible via GitHub à l'url suivante

https://github.com/mewmew92/MvcCms