TP .Net

Bug tracking like

Lemettre Arnaud

Version 1.0

9 Pages

02/04/2009

EPITA-MTI2014-NET-TP-bug-track

Propriétés du document

Auteur	**Lemettre Arnaud**
Version	1.0
Nombre de pages	9
Références	EPITA-MTI2014-NET-TP-bug-track

Historique du document

Date de vision	Version	Auteur	Changements
27/02/2009	0.1	Lemettre	création

Site de référence

description	url
Site MTI	
Blog MTI	

Sommaire

Introduction

Le but de cette série de TP sera la réalisation d'un système de bug tracking. Ce tp se décompose en plusieurs parties. Toutes les semaines vous aurez une partie à rendre. Cette partie sera évaluée. Toute fois si pour une raison quelconque vous n'avez pas réussi à faire convenablement cette partie une correction sera postée une journée avant le début du prochain TP. Vous aurez à charge de l'intégrer dans votre code si celui ne produit pas le fonctionnement attendu.

Ce travail est à faire individuellement, tout code similaire sur deux personnes sera considéré comme un travail non rendu et non négociable.

Durant cette série de TP les notions suivantes seront abordées :

- modélisation BDD
- création de script SQL (Base, tables, données)
- création procédures stockées
- Entities Framework
- Linq To SQL
- Manipulation Open XML
- Génération de Flux XML
- Dynamic Data Entities
- Membership provider / Role provider
- utilisation des pages aspx et composants
- WCF (Webservice et sécurisation)

Chaque partie à rendre ne demande pas plus de 3h de travail par semaine, Bonne chance ;)

Contexte

Un système de suivi de problèmes (de l'anglais bug tracking system ou système de suivi de bogues) est une application informatique qui permet d'aider développeurs et utilisateurs à améliorer la qualité d'un logiciel. Les utilisateurs soumettent les bogues (défauts de fonctionnement) rencontrés dans le logiciel. Les développeurs sont alors toujours au fait des problèmes rencontrés.

L'ensemble de la solution devra utiliser SQL Server 2012 pour la base de données, et le Framework 4.5 de .Net.

Le rendu de chaque lot, correspond au rendu de chaque partie.

Partie 5

Nous avons maintenant un système exploitable. Cependant nous pouvons encore rajouter certaines fonctionnalités.

Nous allons donc rajouter un flux Rss sur le site, qui montrera l'apparition de nouveaux bugs sur les projets.

De plus l'utilisation d'un site peut être compliquée pour certaine personnes, nous allons donc également développer une petite application console qui viendra se connecter directement sur le site pour récupérer certaines informations et en changer d'autre.

Renseignement

Le flux RSS devra afficher le nom du projet et le titre du bug ainsi que la date de modification. Il faudra peut être rajouté des tables en base de données. En ce qui concerne les web services vous devez utilisez WCF et créer un objet qui permettra de stocker la liste des bugs, ainsi que la personne affecté aux bugs.

Une fonction des web services sera de pouvoir affecter une personne à un bug à partir de l'application console.

Travail à faire

Vous devez réaliser le projet (nommé SyndicationServiceBugTrack) permettant d'avoir des flux Rss. Dans un but de simplicité re-développer une couche de dataAccess, pour cela utiliser soit du Linq to Sql ou Entity Framework.

Développer une classe nommée Bug.cs qui vous permettra d'effectuer les requêtes sur la table dans la couche dataAccess.

```
List<T_Bug> GetLastBug(int max)
```

Cette méthode permettra de retourner les derniers bugs triés par ordre croissant en limitant le nombre selon le paramètre max. Afin de remonter les informations jusqu'à l'interface vous devez créer une classe dans la couche BusinessManagement nommée de la même manière Bug.cs. Cette classe contiendra une méthode :

```
public static List<DBO.Bug> GetLastBug(int max)
```

Cette méthode permettra de remonter les bugs de la dataAccess et de les transformer en DBO.

Le flux RSS devra être accessible à travers la classe Feed1 qui implémentera l'interface IFeed1 qui contient la méthode `SyndicationFeedFormatter CreateFeed();` La méthode devra remonter les 10 derniers bugs.

Le flux Rss devra avoir pour titre : « *Bug Track* » et pour description : « *Flux rss du bug track* ». Les items devront avoir la forme suivante :

Nom du projet : Titre du bug « \ n» détails du bug

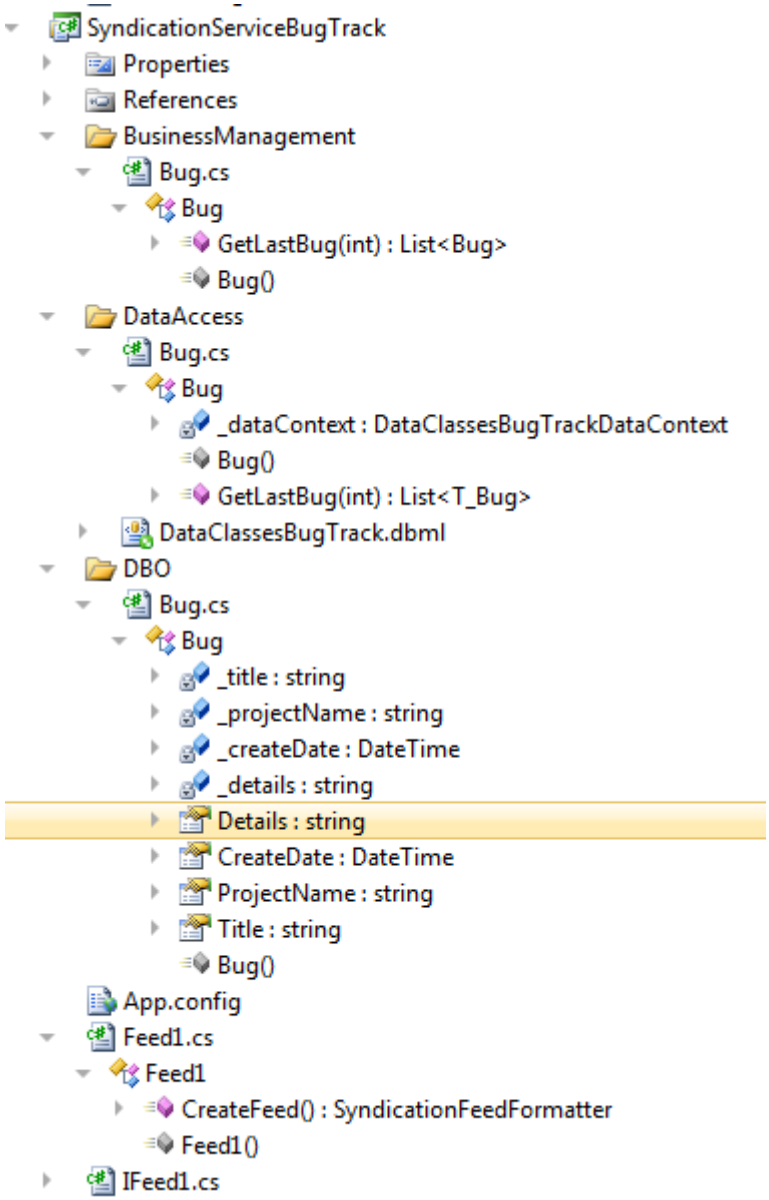

Figure 1 - Architecture du projet

La deuxième étape du TP est de créer un web service dans la partie Web. Pour cela ajouter un service WCF dans la partie web nommé :

ServiceClientBugTrack.svc

Ce service devra implémenter l'interface IServiceClientBugTrack.cs avec pour méthode :

```
List<ServiceReferenceBugTrack.T_Bug> GetListBugWithEF(int max)
```

Cette méthode devra retourner la liste des bugs selon la limite passée en paramètre.

De plus vous devez créer un projet console (ClientBugTrack) permettant de se connecter sur les web services et de récupérer les informations.

La référence au service devra se nommer :

ServiceReferenceBugTrack

Il vous faudra implémenter également une architecture en couche afin de remonter les informations jusqu'à la sortie de la console.

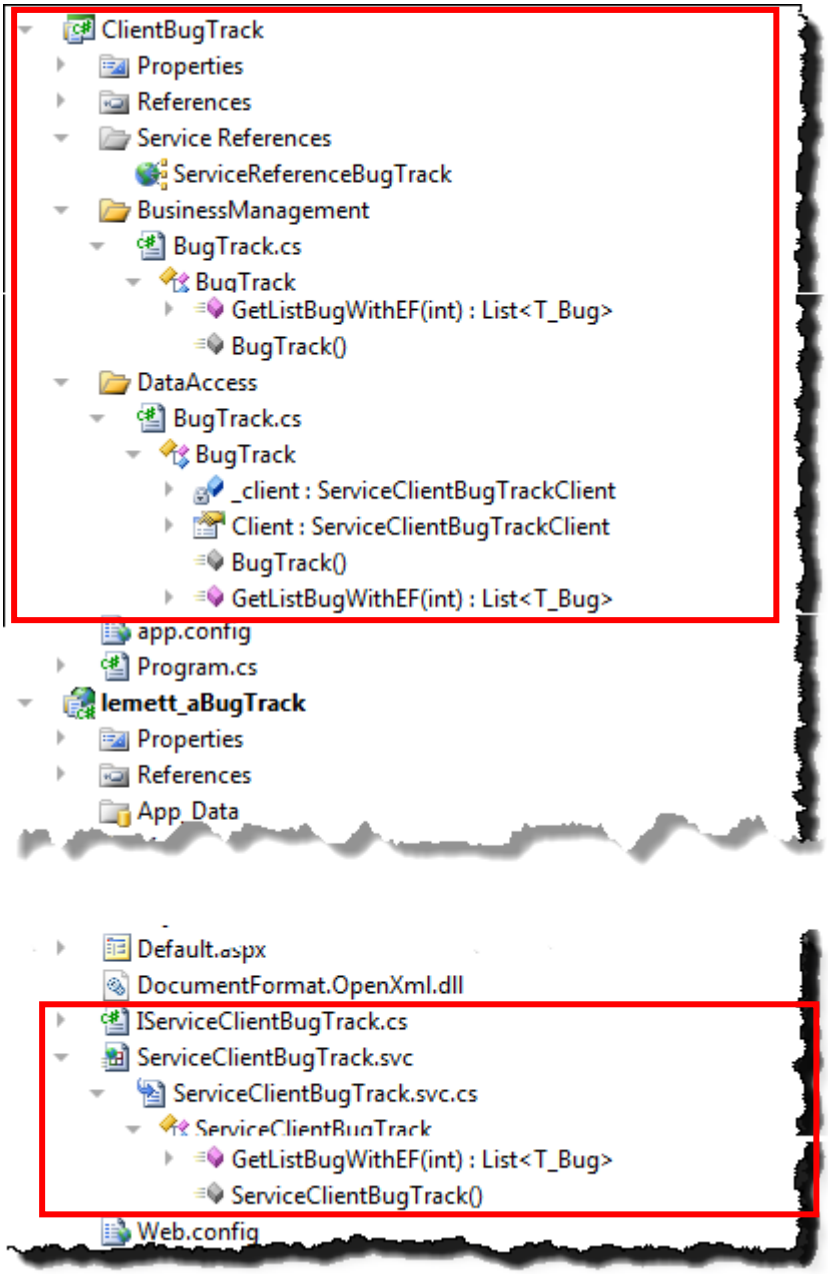

Figure 2 - Architecture serveur et client

Modalité de rendu

Les fichiers seront à rendre dans une tarball ayant pour nom :

login_l.zip

Cette tarball devra comprendre :

- un script de création de votre base ainsi que des tables.
  Nom : CreateBugTrack.sql
- Un script pour remplir votre base avec des données
  Nom : DataBugTrack.sql
- Un dossier contenant la solution Visual Studio qui devra compiler.
  Nom : login_lBugTrack

Le tout à envoyer sur l'adresse rendu.mti.dotnet@gmail.com avec les balises suivantes :

[MTI2014][NET][login_l][BugTrack] partie 5