Architecture Applicative

Différences entre Webforms et MVC

TP

El hadji Ibrahima DIAGO

Master 1 2016 - 2017

ISI
INSTITUT SUPERIEUR D'INFORMATIQUE
L'institut de référence dans les TIC

I. Introduction

1. Définition de l'architecture Logicielle

L'architecture logicielle décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. Contrairement aux spécifications produites par l'analyse fonctionnelle, le modèle d'architecture, produit lors de la phase de conception, ne décrit pas ce que doit réaliser un système informatique mais plutôt comment il doit être conçu de manière à répondre aux spécifications. L'analyse décrit le « quoi faire » alors que l'architecture décrit le « comment le faire ».

- Elle se consacre à structurer et à concevoir une application à partir de ses spécifications fonctionnelles
- Elle structure et décompose de façon logique chaque application en couches
- Elle introduit les notions et concepts de découpage en couches, modules, composants, design patterns et Framework

2. Définition d'un framework

Un framework (cadre de travail) correspond à un ensemble d'outils du marché, de bibliothèques spécifiques et de méthodologies qui visent à faciliter, cadrer et accélérer les développements du projet.

- il définit le cadre de travail et les engagements de chacune des parties, fonctionnelles et techniques, en spécifiant le processus de développement
- il permet de mieux corréler les aspects techniques de l'implémentation des fonctionnalités
- il contraint et standardise les développements
- il diminue les risques liés au projet

3. Le rôle de l'architecte

Le rôle de l'architecte est de :

1. Définir le modèle d'architecture en couches et en tiers à mettre en œuvre pour chacun des blocs applicatifs
2. Préconiser des motifs de conception à mettre en œuvre pour les couches
3. Préconiser les librairies, composants, Framework et outils à utiliser pour :
   - L'implémentation des couches logicielles (présentation / coordination / services / domaine)
   - La fabrication de l'application (conception, développement, intégration, packaging)
   - La mise en production et le suivi (déploiement, configuration, surveillance, suivi de la qualité de service, suivi des erreurs)

4. Guider les phases de conception et de développement en s'assurant que les concepteurs et les développeurs ont bien compris l'architecture

II. Structuration des systèmes en vues

La structuration du système peut être vue sous différents angles, selon que l'on considère le découpage logique (vues en couches) ou le découpage physique (vues en niveaux).

1. La vue en couche

La structuration des applications en couches permet :

- de maîtriser la complexité des applications (développement, échanges entre les applications, interactions entre objets)
- d'optimiser les temps de développement, en factorisant certaines briques applicatives
- d'isoler les problématiques d'enchaînements de processus en définissant et en délimitant le périmètre du contrôle de l'intégrité transactionnelle (locale et distribuée)
- de favoriser la communication :
  a. à l'intérieur d'une application, en structurant les échanges entre les différentes couches
  b. entre les applications en précisant les principes de communication liée aux couches de diverses applications

Le passage d'une couche vers une autre doit impérativement se faire via des interfaces qui représentent chacune un service d'accès (introduction de la notion de contrats de services)

La cohérence entre l'urbanisme (qui conçoit les plans des SI dans une perspective de fonctionnement rationnel et d'évolutivité) et l'architecture d'un projet (qui en construit les blocs applicatifs), est d'autant plus aisée à assurer si l'on parvient à découper les blocs applicatifs selon les couches du modèle de l'architecture applicative logique

La structuration des applications se traduit par une décomposition logique de chaque application en 5 couches : **Présentation**, **Contrôleur**, **Services**, **Domaine** et **Persistance**

- Chaque couche a ses propres responsabilités et utilise la couche située en dessous d'elle
- En fonction du projet, les architectes enrichissent et élaguent le modèle. La structuration est alors guidée par les contraintes exprimées et existantes

2. La vue en niveau

La vue en niveaux (la tier view) donne une vision plus « physique » de la structuration de l'application. Les niveaux (ou tiers) peuvent être répartis physiquement sur différents composants matériels.

On identifie un changement de « niveau » dès qu'un module logiciel doit passer par un intermédiaire de communication (middleware) pour en invoquer un autre. Si l'utilisation du middleware est en général transparente pour les développeurs, elle n'est pas sans impact sur l'architecture. L'architecte doit donc maîtriser les caractéristiques (client/serveur, publication/abonnement, sécurité, support du transactionnel, …) et en justifier l'usage.

Des modèles standards de répartition de niveaux ont été définis dans les projets par l'industrie au fur et à mesure de l'évolution des capacités matérielles et des besoins

- **Modèle à 1 tiers**
  a. Le modèle à 1 niveau (ou tiers) correspond à un binaire dans lequel s'exécutent toutes les couches, de la présentation à la persistance.
  b. C'est l'exemple de l'application utilisée en monoposte ou sur un réseau de serveurs de fichiers, ainsi que de l'application sur système central.
  c. Les données sont stockées sur un fichier local ou partagées sur un serveur de fichiers

- **Le modèle à 2 tiers**
  a. Le modèle à 2 niveaux (ou tiers), encore appelé « client/serveur première génération », repose sur l'utilisation de moteurs de bases de données relationnelles. Ces moteurs permettent de distribuer la gestion de la persistance sur un serveur
  b. Ce modèle a typiquement permis de mieux répondre au besoin d'accès concurrents et de supporter d'importants volumes
  c. C'est avec ce type d'architectures que l'on a pu gagner en flexibilité et se passer des onéreux systèmes centraux
  d. L'application d'entreprise peut ainsi être accédée depuis un ordinateur personnel avec des standards de présentation moderne

- **Modèle n-tiers**
  a. La littérature parle du modèle générique « N-tiers » (ou N-niveaux)
  b. Le modèle N-tiers est celui mis en œuvre dans le cadre des projets web
  c. Exemple : tiers impliqués dans le modèle d'architecture J2EE

III. Différence entre Webform et MVC

1. Rappel su principe MVC

Le patron de conception Modèle Vue Contrôleur introduit la séparation et la limitation des responsabilités de 3 éléments.

– Le modèle représente les données.

– La vue les affiche et permet à l'utilisateur de lancer des actions sur le Controller.

– Le contrôleur lorsque l'utilisateur appelle une de ses actions : interroge ou met à jour le modèle, choisit la vue et l'affiche.

Jusqu'à la sortie du framework Microsoft ASP.net MVC en Mars 2009. Tous les sites web développés en ASP.net sans framework autre que celui du framework .net utilisent la technologie « WebForm ». Cette technologie permet au développeur de coder comme pour une application en client lourd. La page se constitue de contrôles, les valeurs de contrôles sont sauvegardées entre les postback d'une même page grâce au ViewState, un designer est fourni sous Visual Studio et la programmation est évènementielle.

En pratique, ASP.net est une technologie difficile à maitriser. D'abord l'utilisation de contrôles rend le code HTML souvent lourd et non maitrisé. Le cycle de vie de la page est complexe. Grâce au modèle de programmation évènementiel on a vite fait générer des allers-retours serveur à tout bout de champ. La gestion du javascript est fastidieuse et obscure.

Enfin, par défaut tous les contrôles enregistrent leur état dans le ViewState ce qui l'alourdit et fait d'ASP.net une technologie qui parait poussive parfois même en réseau local, à cause du ViewState renvoyés à chaque appel du client.

ASP.net MVC propose beaucoup moins de fonctionnalités « out of the box » mais il donne accès direct à la génération du code HTML. Sa prise en main est rapide pour un développeur web venant de n'importe quelle autre technologie utilisant HTML et Javascript. Les pages sont plus légères mais il n'a pas de ViewState ce qui suppose un effort du développeur pour coder des interfaces un peu complexe nécessitants des allers-retours serveur en Ajax.

ASP.net MVC, par son découplage des contrôleurs permet de coder des tests unitaires, ce qui est très loin d'être évident voire impossible avec ASP.net WebForms. Le tableau ci-dessous permet de voir les differences majeurs entre ces deux technologies.

ASP.Net MVC	**ASP.Net Web Forms**
Les views et la logique sont séparées	Pas de séparation
Introduction du concept du routing en se basant sur les Urls. (Ils sont déclarés dans Global.asax pour info)	Le routage se fait en se basant sur les pages elles-mêmes
Support de la syntaxe ainsi que celle du .aspx	Support de la syntaxe .aspx sans Razor
La gestion des états se fait via Tempdata, ViewBag, and View Data.	On utilise le View State pour stocker des informations ce qui alourdit la page
Utilisation des Partial Views	Utilisation des User Controls
Possibilité d'utilisation des HTML Helpers	Utilisation des contrôles serveurs
Plusieurs pages peuvent avoir le même contrôleur pour satisfaire leurs besoins. Un contrôleur peut comporter plusieurs actions	Chaque page a son propre code, en d'autres termes dépendance directe. Par exemple test.aspx dépend de test.aspx.cs (code behind).
Les contrôleurs sont de « simples » classes que vous pouvez **instancier sans contexte Http**. Elles sont donc plus faciles à tester avec des tests unitaires par exemple.	Les pages sont difficiles à instancier en dehors d'un fonctionnement normal. Ceci a pour effet de les rendre impossible à tester de manière automatique
Utilisation des layouts	Master pages

2. Quelques points en commun

- Sont fait dans le cadre ASP.Net
- Appartiennent au même espace de nom System.Web.
- Support des langages du Framework .Net
- Envoie la réponse au client sous forme de HTML.

NB : Le choix entre les pages WebForms et les pages MVC se fera en fonction de l'évaluation des critères suivants : nombre de pages Web dans le projet, exigence en matière d'architecture (la présence d'un architecte étant souhaitée), connaissances des développeurs en matière d'architecture, temps alloué pour développer le projet.