Test de recrutement classique en tant qu'Ingénieur ou Architecte .Net avec quelques notions Java

Merci 38

INTRODUCTION

Ce tutoriel a pour but de vous préparer à un test technique sur les technologies .NET et un peu en général sur les notions de programmation objet que nécessite chaque acteur dans un projet informatique (chef de projet, directeur de projet, architecte, ingénieur d'étude et dev) Perso, je voulais garder ce test pour le mettre à disposition dans ma structure mais je vais être généreux et partager mon expérience avec vous. Je n'avais jamais réussi à trouver une telle chose sur la toile, on dirais que tous les informaticiens sont des rageurs égoïstes. Ce n'est pas mon cas, je partage. Dernière chose, si vous tombez sur moi en test technique de recrutement fuyez :D

C'EST QUOI LES PRINCIPAUX POINTS DANS LA PROGRAMMATION OBJET ?

- Encapsulation : Les attributs (ou plus exactement leur représentation informatique) et les méthodes sont cachés ; ils forment une boîte noire. C'est le principe d'encapsulation. Son avantage principal réside dans la capacité à pouvoir modifier la structure interne des objets ou les méthodes associées aux messages sans impact sur les utilisateurs des objets. - Le typage et le polymorphisme : Un objet peut appartenir à plus d'un type. C'est ce que l'on appelle le polymorphisme. Ceci permet d'utiliser des objets de types différents là où est attendue une valeur d'un type précis, dès que ceux-ci satisfont le type requis. Une façon de réaliser le polymorphisme est le sous-typage (appelé aussi héritage de type) : mécanisme par lequel est raffiné un type père en un autre type, le sous-type ; c'est un mécanisme de restrictions des espaces de valeurs du type. Les objets de ce sous-type sont conformes aussi au type père, ces objets sont donc d'au moins deux types. —> typage dynamique, typage statique, typage par inférence - La redéfinition - La programmation objet permet à un sous-type de raffiner la mise en oeuvre d'un message défini dans le type père, autrement dit de redéfinir la méthode associée au message : c'est le principe de redéfinition des messages (ou overriding en anglais). - Classe et prototype

CONNAISSEZ VOUS LES DESIGN PATTERN ? C'EST QUOI LEURS CATÉGORIES ? DONNER QUELQUES EXEMPLES PERTINENTS !

Un patron de conception (design pattern en anglais) est un concept de génie logiciel destiné à résoudre les problèmes récurrents suivant le paradigme objet. En français on utilise aussi le terme motif de conception qui est une traduction alternative de design pattern, perçue comme incorrecte par certains. Les patrons de conception décrivent des solutions standard pour répondre à des problèmes d'architecture et de conception des logiciels. À la différence d'un algorithme qui s'attache à décrire d'une manière formelle comment résoudre un problème

particulier, les patrons de conception décrivent des procédés de conception généraux. On peut considérer un patron de conception comme une formalisation de bonnes pratiques, ce qui signifie qu'on privilégie les solutions éprouvées. IL EXISTE 3 TYPES DE DESIGN PATTERN : 1) de construction : ils définissent comment faire l'instanciation et la configuration des classes et des objets.

- Fabrique abstraite (Abstract Factory)
- Monteur (Builder)
- Fabrique (Factory Method)
- Prototype (Prototype)
- Singleton (Singleton)

vous devez connaitre la définition de chaque design 2) structuraux : ils définissent comment organiser les classes d'un programme dans une structure plus large (séparant l'interface de l'implémentation).

- Adaptateur (Adapter)
- Pont (Bridge)
- Objet composite (Composite)
- Décorateur (Decorator)
- Façade (Facade)
- Poids-mouche ou poids-plume (Flyweight)
- Proxy (Proxy)

3) comportementaux : ils définissent comment organiser les objets pour que ceux-ci collaborent (distribution des responsabilités) et expliquent le fonctionnement des algorithmes impliqués.

- Chaîne de responsabilité (Chain of responsibility)
- Commande (Command)
- Interpréteur (Interpreter)
- Itérateur (Iterator)
- Médiateur (Mediator)
- Mémento (Memento)
- Observateur (Observer)
- État (State)
- Stratégie (Strategy)
- Patron de méthode (Template Method)
- Visiteur (Visitor)

C'EST QUOI LE MODÈLE MVC ?

L'organisation globale d'une interface graphique est souvent délicate. L'architecture MVC ne résout pas tous les problèmes. Elle fournit souvent une première approche qui peut ensuite être adaptée. Elle offre aussi un cadre pour structurer une application. Ce modèle d'architecture impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.

C'EST QUOI LA DIFFÉRENCE ENTRE MVC ET COUCHES 3 TIERS ?

L'architecture trois tiers est un modèle en couches, c'est à dire, que chaque couche communique seulement avec ses couches adjacentes (supérieures et inférieures) et le flux de contrôle traverse le système de haut en bas; les couches supérieures contrôlent les couches inférieures, c'est-à-dire, que les couches supérieures sont toujours sources d'interaction (clients) alors que les couches inférieures ne font que répondre à des requêtes (serveurs). Dans le modèle MVC, il est généralement admis que la vue puisse consulter directement le modèle (lecture) sans passer par le contrôleur. Par contre, elle doit nécessairement passer par le contrôleur pour effectuer une modification (écriture). Ici, le flux de contrôle est inversé par rapport au modèle en couche, le contrôleur peut alors envoyer des requêtes à toutes les vues de manière à ce qu'elles se mettent

à jour. Dans l'architecture 3-tiers, si une vue modifie les données, toutes les vues concernées par la modification doivent être mises à jour, d'où l'utilité de l'utilisation du MVC au niveau de la couche de présentation. La couche de présentation permet donc d'établir des règles du type "mettre à jour les vues concernant X si Y ou Z sont modifiés". Mais ces règles deviennent rapidement trop nombreuses et ingérables si les relations logiques sont trop élevées. Dans ce cas, un simple rafraîchissement des vues à intervalle régulier permet de surmonter aisément ce problème. Il s'agit d'ailleurs de la solution la plus répandue en architecture 3-tiers, l'utilisation du MVC étant très moderne et encore marginale.

DONNER UN EXEMPLE D'IMPLÉMENTATION DE MODÈLE MVC ?

L'utilisation du MVC est native en Swing. La plupart des composants Swing (hormis les conteneurs) utilisent une classe spécifique pour contenir leurs données.

- Les boutons (JButton) utilisent un objet de classe ButtonModel pour les données.
- Les curseurs (JSlider) utilisent un objet de classe BoundedRangeModel pour les données.
- Les listes (JList) utilisent un objet de classe ListModel pour les données et un objet de classe ListSelectionModel pour gérer les sélections.
- Les arbres (JTree) utilisent un objet de classe TreeModel pour les données et un objet de classe TreeSelectionModel pour gérer les sélections.
- Les tables (JTable) utilisent des objets de classe TableModel et TableColumnModel pour les données et un objet de classe ListSelectionModel pour gérer les sélections.
- Les composants de texte (JTextComponent et les classes dérivées JEditorPane, JTextArea et JTextField) utilisent un objet de classe Document pour gérer le texte.

C'EST QUOI LA DIFFÉRENCE ENTRE LE MOT CLÉ VIRTUAL ET OVERIDE EN C#.NET ?

Le mot virtual permet de définir une méthode dans une classe mère qui peut être changé dans la classe fille à l'aide du mot clé override.

A QUOI SERT LE MOT CLÉ DELEGUE EN C#.NET ?

Un délégué est un type qui référence une méthode. Une fois qu'une méthode est assignée à un délégué, ce dernier se comporte exactement comme cette méthode. La méthode du délégué peut être utilisée comme n'importe quelle autre méthode, avec des paramètres et une valeur de retour On peut l'assimiler aux pointeurs fonction C++, mais sont de type sécurisé.

QUAND UTILISER DES DÉLÉGUÉS AU LIEU D'INTERFACES ?

les délégués et les interfaces permettent à un Concepteur de classes de séparer les déclarations de type et l'implémentation. Une interface donnée peut être héritée et implémentée par n'importe quelle classe ou struct. Un délégué peut être créé pour une méthode de n'importe quelle classe, tant que la méthode convient à la signature de méthode pour le délégué. Une référence d'interface ou un délégué peut être utilisé par un objet sans connaissance de la classe qui implémente l'interface ou la méthode déléguée. Étant donné ces ressemblances, quand un Concepteur de classes doit-il utiliser un délégué et quand doit-il utiliser une interface ? Utilisez un délégué quand :

- un modèle de design d'événement est utilisé.
- il est souhaitable d'encapsuler une méthode statique.
- l'appelant n'a aucun besoin d'accès aux autres propriétés, méthodes ou interfaces sur l'objet qui implémente la méthode.
- une facilité de composition est souhaitée.
- une classe peut avoir besoin de plusieurs implémentations de la méthode.

Utilisez une interface quand :

- Il existe un groupe de méthodes connexes pouvant être appelé.
- Une classe a besoin d'une seule implémentation de la méthode.
- La classe qui utilise l'interface souhaite effectuer un cast de cette interface vers d'autres types d'interfaces ou de classes.
- La méthode qui est implémentée est liée au type ou à l'identité de la classe : par exemple, les méthodes de comparaison.

C'EST QUOI UN THREAD EN C# ?

En programmation pure, on ne parle plus de tâches, mais de threads ou de processus. Il existe une différence entre ces définitions mais l'expliquer nous conduirait à nous éloigner de l'objectif de ce cours. Par la suite, je parlerai donc de threads. En C# et dans tous les autres langages du framework .NET, la classe Thread se trouve dans l'espace de noms System.Threading. Pour simplifier, un objet de la classe Thread symbolise une tâche. L'utilisation des threads avec .NET a été fortement simplifiée comparée aux méthodes natives Win32. Nous allons dans ce cours voir comment créer, utiliser et détruire des threads managés, ainsi que quelques mécanismes de protection de ressources critiques.

COMMENT UTILISER UN THREAD ?

PS : je conseille de voir ce tuto : http://emerica.developpez.com/csharp/threads/

C'EST QUOI UN REPEATER EN ASP.NET ?

Le contrôle Repeater va nous permettre à l'instar d'un DataGrid, par exemple, de faire un listing de données issu d'une requête SQL en suivant, cette fois ci, un template défini par nos propres soins plutôt qu'un tableau bête et méchant.

C'EST QUOI UN INDEX, UNE VUE, UN CURSEUR, UNE TRANSACTION, UN JOURNAL DE TRANSACTION ?

TRANSACTION ?

Vous devez revoir ces notions une par une, il n'est pas indispensable de les maîtriser mais juste de les connaitre.

QUEL MODELÉ RELATIONNEL UTILISEZ VOUS POUR INTERFÉRER AVEC UNE BASE DE DONNÉE ? DONNER UN EXEMPLE ET DÉCRIVEZ LES ÉTAPES ET OUTILS EMPLOYÉS.

Vous devez voir si vous avez déjà travailler avec Hybernate ou bien LINQ to SQL sinon une simple connaissance d'ADO.NET et des traitement en mode connecté et mode non connecté suffisent.

POUR ALLER PLUS LOIN

Il existe aussi d'autre exercices de programmation et de détection de bug dans un programme. Ils ne sont pas très compliqué mais je n'ai pas la liste sur moi. En plus, ils changent souvent mais pour la plupart des questions que je viens de citer en haut, ils ne sont pas spécifiques à une société, mais presque toutes les sociétés posent les mêmes parce que il n'y a pas dix mille tests techniques... Connaitre toutes ces réponses garanti le passage de l'étape de test technique