

**Guide de préparation aux entretiens clients**

1. **Préparation technique de base**
2. **Se renseigner sur les design patterns:**

**Design Pattern Créationnel**

* **C’est qui un Singleton, comment le créer ?**
* Le **singleton** est un [patron de conception](https://fr.wikipedia.org/wiki/Patron_de_conception) *(design pattern)* dont l'objectif est de garantir qu'une unique instance d'une classe donné sera créée et ainsi offrir un point d'accès universel à cette instance. Il existe plusieurs techniques pour implémenter ce design pattern.

Citons :

LazyLoading (voir GIT repo) : l'instance du Singleton est automatiquement créée au chargement de la classe par son classloader.

Eager Loading (voir GIT repo) : L'initialisation de l'instance est retardée jusqu'au premier appel de "getInstance()"

* **Définition de Factory**

Le design pattern Factory, ou **Fabrique** est un design pattern permettant de séparer la création d'objets dérivant d'une classe mère de leur utilisation. De ce fait, on a alors la possibilité de créer plusieurs objets issue d'une même classe mère. (Voir GIT Repo)

Factory vs Abstract Factory :

Prenons le cas de la construction d’un ordinateur de type (PC ou serveur) :

Dans le cas de design pattern Factory, vous allez donner les spécifications de l’ordinateur (RAM, GPU, CPU, type, …) à la fabrique et c’est à la fabrique de créer le PC demandé.

Dans le cas de design pattern Abstract Factory, vous allez faire recours à une fabrique spécialisé pour construire l’ordinateur.

**Design Pattern Comportemental**

* **Définition de Proxy**

Un proxy est une classe se substituant à une autre classe. Par convention et simplicité, le proxy implémente la même interface que la classe à laquelle il se substitue[1](https://fr.wikipedia.org/wiki/Proxy_(patron_de_conception)#cite_note-gof-1). L'utilisation de ce proxy ajoute une indirection à l'utilisation de la classe à substituer (voir GIT repo)

* **Définition de Observer**

 Il est utilisé pour envoyer un signal à des modules qui jouent le rôle d'observateurs

* **Définition de Commande**

Il encapsule la notion d'*invocation*. Il permet de séparer complètement le code initiateur de l'action, du code de l'action elle-même**.**

* **Définition L’inversion de contrôle**

-(*inversion of control*, IoC) est un patron d'[architecture](https://fr.wikipedia.org/wiki/Architecture_logicielle) commun à tous les [frameworks](https://fr.wikipedia.org/wiki/Framework) (ou cadre de développement et d'exécution). Il fonctionne selon le principe que le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du framework ou de la couche logicielle sous-jacente.

-L’inversion de contrôle est un terme générique. Selon le problème, il existe différentes formes, ou représentation d'IoC, le plus connu étant l'[injection de dépendances](https://fr.wikipedia.org/wiki/Injection_de_d%C3%A9pendances) (*dependency injection*) qui est un [patron de conception](https://fr.wikipedia.org/wiki/Patron_de_conception) permettant, en [programmation orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet), de découpler les dépendances entre objets.

* **Définition de l’Injection de dépendance**

est un mécanisme qui permet d'implémenter le principe de l'[inversion de contrôle](https://fr.wikipedia.org/wiki/Inversion_de_contr%C3%B4le).

Il consiste à créer dynamiquement (injecter) les dépendances entre les différents objets en s'appuyant sur une description (fichier de configuration ou métadonnées) ou de manière programmatique. Ainsi les dépendances entre composants logiciels ne sont plus exprimées dans le code de manière statique mais déterminées dynamiquement à l'exécution.

* **Définition du MVC pour le monde J2EE**

**2- Java standard :**

* Qu’est-ce que Java ? Comment le définiriez-vous ?

Java est un langage de programmation moderne développé par **Sun Microsystems** (aujourd'hui racheté par **Oracle**)

* Quel est votre niveau sur Java 8 ? Quelles sont les nouveautés de Java 8 ?

Mon niveau en java 8 est moyen.

-Méthodes par défaut et statique dans l’interface.

Interface **default** methods:

* Elle aide à l’extension d’une interface spécifique par l’ajout d’une méthode sans se soucier d’une potentielle erreur dans les classes qui implémentent cette interface.

Interface **static** methods:

* Elle permet de protéger une méthode d’être re-implémentée/redéfinie dans une classe fils. (voir code GIT). Elles permettent de supprimer les classes utilitaires par l’ajout directe de la méthode utilitaire à l’interface correspondente

-Références de méthodes :

Les références de méthodes permettent d'offrir une syntaxe simplifiée pour invoquer une méthode comme une expression lambda : elles offrent un raccourci syntaxique pour créer une expression lambda dont le but est d'invoquer une méthode ou un constructeur. (Voir GIT)

-Expression lambda.

-Annotation de type.

L’apparition de nouveaux API :

-Date and Time API (package java.time au lieu de java.util.date et java.util.calendar)

Limitation de java.util.date :

-Elle ne supporte pas la notion de timezone.

-Elle est mutable (risque d’erreur et thread insafety).

-Les dates antérieurs à 1900 ne sont pas supportés.

Solution :

L’utilisation de java.time : cette classe est immutable donc threadSafe.

Date(LocalDate), time, date/time(LocalDateTime), time zones, instants, duration (difference between two instants), Period (difference between two dates).

-Interface fonctionnelle :

Ce concept, introduit depuis Java 8, permet de définir une interface n'ayant qu'une et une seule méthode abstraite : c'est grâce à cette restriction qu'il sera possible d'utiliser les lambdas car, lors de l'exécution, Java pourra automatiquement déterminer quelle est la signature de la méthode que la lambda remplace et tout sera automatique.

On pourra annoter cette interface avec @FunctionalInterface.

Il y a plusieurs interfaces fonctionnelles citons par exemple l’interface Runnable.

-Améliorations de la gestion de concurrence.

-Le moteur Javascript Nashorn.

* Quelles sont les différences avec Java 7 ? Quelles sont les nouveautés sur Java 9 ?
* Qu’est-ce qu’une expression lambda ?
* Concernant les « LAMBDA » ; où peut-on les utiliser et à quoi servent telles

?

* Qu’est-ce qu’un Stream ?
* C’est quoi la différence entre Stream et List
* Dans un stream, si vous appelez uniquement des méthodes intermédiaires sans appeler des méthodes terminales, quand on lance le programme, que ce passe-t-il ?
* Utilisation des « stream», imaginez que vous faîtes une collecte dessus ? Pouvez-vous le réutiliser ?
* Utilisation du « Classpath », imaginez que vous avez une application, sur le

« classpath » de cette application vous avez 2 versions d’une même librairie

(avec les mêmes classes mais pas les mêmes implémentations), peut-on savoir si c’est les classes de la librairie 1 ou 2 qui sont utilisées ?

* Différence entre type primitif et objet ? Différence entre Int et Integer ?
* Avez-vous utilisé des « moc » (bouchons) ? Exemple j’ai un service qui dépend d’un DAO, comment faire pour monter un service sans attaquer la

BDD ?

* Comment définiriez-vous le type optional ?
* Classe abstraite : définition et utilité
* Différence entre Set, List, Map, Tree, Queue
* Connaitre les Array, Hashtable, Hashmap, Hashset, Treemap, Collection...
* Différence entre LinkedList et ArrayList
* Différence entre Hashtable, Hashmap, TreeMap



1/6



* Différence entre ArrayList et Vector
* Comment implémenter une Hashtable
* Comment parcourir les éléments d’une Map
* Peut-on supprimer un élément d’une Collection dans une boucle for ?
* Comment remédier à ce problème (Supprimer un élément dans une boucle)

?

* Signification de synchronized, volatile, transient
* Création des threads, thread safe method
* Différents classes d’exception/error/runtime
* Méthodes hashCode() et equals() (définition de chaque une et pour quelle utilisation)
* Deux objets égaux ont-ils forcément le même hashcode ?
* Deux objets avec le même hashcode sont-ils forcement égaux ?
* Définition du Polymorphisme, les différentes formes du polymorphisme
* Surcharge et redéfinition en java
* Les 4 niveaux d'accessibilité
* Définition d’une classe anonyme
* Définition des classes internes
* Les génériques : dans quel cas les uitiliser
* La réflexion (reflection) : définition, utilité
* Comment fonctionne le garbage collector ? Connaissez-vous une méthode qui permette de déclencher un garbage collector ?
* Peut-on forcer le passage du garbage collector ?
* Multithreading en Java et comment créer un Thread
* Principe de fonctionnement Garbage Collector
* Paramétrage de la mémoire

**3- Spring :**

* Qu’est-ce que Spring ? A quoi sert-il ? Quels sont les avantages de l’utilisation de Spring ? En quoi cela peut vous faire gagner du temps ?
* A quoi sert l’utilisation des annotations dans spring ?
* Scope des beans
* IOC (L’inversion de contrôle), fonctionnement et but de ce concept.
* Différence entre Spring et les EJBs. Conteneur léger vs lourd.
* Connaitre les différents modules et principe de fonctionnement.
* Qu’est-ce que l’injection des dépendances ?
* Les annotations REST de Spring MVC et leur différence avec Jersey
* Quel est le grand concept de Spring ? Le concept qui permet d’utiliser des

@(injection des dépendances)

* C’est quoi le Spring boot ?
* A-t-il une façon de gérer Spring MVC au niveau des contrôleurs ?
* Avec spring mvc, je peux renvoyer de l’XML et comment ?
* Comment configurer Spring ?



2/6



* Que sont les transactions ? Comment utilisez-vous les transactions ? Si on lance une exception, que ce passe-t-il ? Ceci est-il pareil pour tous les cas d’exception ou seulement pour un cas particulier d’exception ?
* Qu’est-ce qu’un attribut transactionnel (ex : tx required) ?
* Comment et quand annuler une transaction ?
* Que sont les niveaux d'isolation des transactions ?
* Qu’est-ce que la notion de profil dans Spring ?
* Lorsque vous annotez votre méthode avec @transactionnal, qu’est-ce que spring fait ? Quel est le mécanisme derrière ?
* Terme utilisé pour citer ce type de manipulation ? (proxy)
* @Autowired vs @Inject vs @Resource
* L’annotation @Qualifier
* Les différents modes de auto wiring
* C’est quoi un inner bean
* Peut-on injecter null et String vide dans Spring ?
* AOP fait-il partie de Spring core ?
* Quelle classe on peut utiliser pour exécuter des requêtes SQL dans Spring ?
* C’est quoi Thread scope
* Le rôle de DispatcherServlet dans Spring MVC
* Spring Data par rapport à Hibernate

**4- Hibernate :**

* Avantages d’un ORM sur JDBC
* Connaitre les composants/Objets : Configuration, SessionFactory, Session, Transaction, Query et Criteria
* SessionFactory est-il thread safe ?
* Session est-il thread safe ?
* Différences entre save() et persist()
* Différences entre get() et load()
* La signification des concepts clés : ORM, cardinalité, proxy, gestion des modifications, identity, les maping one to many…
* Lazyloading
* Les 3 états d’une instance d’une entité
* Différence avec JPA
* Niveaux caches hibernate ? Niveau 1 vs niveau 2 ? Comment configurer le niveau 2 de cache ? différentes stratégies utilisées
* Comment fonctionne le cache de requêtes

**5- Webservices :**

* Définition WSDL : contrat webservices…
* Les implémentations de moteurs SOAP avec Java
* Comment consommer une réponse SOAP ?
* Connaitre quelques outils de génération des classes webservices
* Connaitre le principe des webservices Rest
* RestFul : lister les méthodes
* Microservices : définition



3/6



**6- Angular :**

* Angular JS : directives
* Angular JS vs Angular 2/4/5
* Comment boucler: itérer sur une liste avec typescript: For vs FOR OF vs FOR IN vs FOREACH
* Différence entre JavaScript vs ECMAScript 6 (ES6) vs TypeScript
* L’utilité des annotations @Input et @Output sur Angular
* Comment partager les données entre les composants
* C’est quoi un Model, un Service, un Composant, un Module sur Angular
* Angular CLI : définition + exemples de commandes possibles
* Reactive programming avec RxJS/ngRx et les Observables

**7- Question sur Javascript :**

* Comment pouvez-vous décrire Javascript ? Est-il orienté objet ?
* Côté Front-End, sur angular.js, quels est la différence entre un contrôleur et un service ?
* Quels sont les composants les plus utilisés sur Angular.js ? Comment fonctionne les web-services ? Les contrôleurs ?
* Qu’est-ce que $Scope ? Peut-on se passer de $Scope pour afficher les données d’un contrôleur dans une vue ?
* Qu’est-ce que les routes dans le contrôleur ? Qu’est-ce que UI-Router ?
* Pourriez-vous me citer des designs patterns que vous connaissez ? Comment fonctionne-t-il ?
* Quand un service est un singleton, quand est-il du contrôleur ? Est-ce également un singleton ?
* En javascript natif, qu’est-ce qu’est la fonction apply ?
* Qu’est-ce que la fonction bind ?
* Maitrisez-vous le développement CSS/HTLM ?
* Pourriez-vous me cité des designs patterns que vous connaissez ? Comment fonctionne-t-il ?
* Avez-vous fait la version 2 Angular ?
* Donnez-moi la différence entre NG model et NG bye ?
* Quelle est la dernière version d’AngularJS que vous avez utilisé ?
* Parlez-moi des directives ?
* Comment avez-vous fait pour isoler ses composants ? comment définir l’implémentation ? Scope de directive ?
* La clé scope : **plusieurs possibilités de valeurs et par défaut une scope à quelle** **valeur.** (scope deux points « accolable » ) ça vous parle ?
* Connaissez-vous ES2015 ? Avez-vous utilisé les modules javascript et pas angular ?
* Connaissez-vous vous le spring Template d’angular ?
* La différence entre double égale et triple égale ?
* Différence double côte / simple côte ? sommes-nous interdit d’en utiliser certaines en javascript ?



4/6



**8- SQL-PL/SQL:**

* Tuning des requêtes : ajout d’index…
* C’est quoi un Curseur dans PL/SQL
* C’est quoi une exception ? Comment la créer ?
* C’est quoi les hints ? ça sert à quoi ? Donner un exemple ou 2.
* Fonctions d’agrégation
* C’est quoi les jointures ? C’est quoi une jointure interne et une jointure externe ? Donner un exemple.

**9- Serveur d’application :**

1. Service JNDI
2. Notion datasources

**10- Outils :**

* Jenkins (rôle) ? job Jenkins ?
* SonarQube (rôle, majeur vs critique vs mineur - règles)
* Dette technique : Définition
* Marathon : rôle
* Comment configure-t-on un job jenkins ?
* Pouvez-vous me citer les principales failles de sécurité ? Pourriez-vous me les expliquer ? Comment s’en protège-t-on ?
* Avez-vous déjà utilisé GIT ? En ligne de commande ou à travers une interface graphique ?
* Quelle commande vous feriez pour créer une branche ?
* Avez-vous travailler avec GIT et SVN ? c’est quoi la différence ?
* Connaissez-vous l’agile ? Le Scrum ? Comment cela fonctionnait dans vos anciennes xp ?

**11- Projets :**

* Connaissez-vous les tests unitaires ? Quels outils avez-vous utilisés ?
* Quel est le principe TDD et BDD ? A quoi correspondent-ils ?
* A quoi correspondent les tests d’intégration ?
* Avez-vous déjà codé des tests par exemple avec MGMock qui tourne dans Karma ?
* Comment vous allez tester une classe qui implémente le design paterne singleton pour vérifier avec un test unitaire que la classe respecte bien le design paterne singleton ?
* Différence entre tests unitaires et tests d’intégration
* Mokito : rôle. Différence entre objet Mock et objet Spy
* Tests Postman
* Tests fonctionnelles automatisés (avec les outils : FITNESSE, CUCUMBER, …)

(Définitions)

* Fonctionnement méthodologie Agile/scrum (Cycle ?)
* TDD (Test-Driven Development): définition, cycle



5/6



* + Maven : définition
  + Maven : comment configurer/ automatiser les tests
  + Quel est votre niveau sur Maven ?
  + Quel est le lifecycle de Maven ?
  + Que fait le plugin assembly de Maven ?
  + Pour définir des profils de build dans Maven ?
  + Quelle est la différence entre la commande mvn install et mvn install : install ?
  + Principe d’Intégration continue : Jenkins, repos livrable, tests unitaires, Sonar…
  + Connaitre quelques outils de bug tracking
  + Git vs SVN
  + Comment invoquer un pluging ?
  + Connaissez-vous l’acronyme SOLID ?

**II.Présentation en entretien (Prioritaire)**

Avant toute chose, le consultant doit maitriser la présentation du CV lors des entretiens et connaitre par cœur le contexte de chaque mission dans l’ordre chronologique.

La présentation du CV doit commencer par la dernière mission réalisée:

1. Présenter le nom du projet et son contexte fonctionnel par rapport au client
2. Présenter le service rendu par l’application
3. Détailler votre contribution au sein du projet: conception, développement, support, résolution de bug, réalisation d’évolution, intégration…
4. Présenter l’architecture technique (3 Tiers, n-Tiers…) en commençant par la couche présentation et en terminant par les couches de persistance et/ou interconnexion avec d’autres modules via des connecteurs…
5. Au fur et à mesure de la présentation de l’architecture, vous citerez les

Frameworks et technos utilisés et qui sont détaillés dans la partie

* + environnement technique ».

**III.Préparation aux tests techniques**

Certains clients exigent le passage de test technique comme prérequis avant tout présentation en entretien.

En plus des éléments cités dans le premier paragraphe, vous pouvez vous connecter sur les sites internet suivants qui proposent des QCM avec réponse autour de différents technologies : JEE, Spring, hibernate, JSF, Angular JS,…

1. <http://www.tutorialspoint.com/questions_and_answers.htm>
2. <http://www.buggybread.com/2013/09/java-online-practice-tests.html>

Test Blanc SCJP : Très utile pour préparer le certificat SCJP

1. <http://javatests.hebergratuit.net/?i=2>



6/6



**IV.** **Réussir son entretien Téléphonique/Physique**

Selon la mission, il vous sera demandé de faire un point téléphonique ou de vous présenter pour passer un entretien physique chez le client final ou chez le partenaire.

Dans tous les cas, il est impératif de:

* Respecter l’heure du rendez-vous et de prévoir les aléas des transports en commun.
* Se présenter en costume/cravate (consultants) ou ensemble tailleur (consultantes).
* Avoir toujours sur soi, un cartable avec bloc note et un stylo pour prendre des notes lors de l’entretien.
  1. **ETRE RASSURANT**
* Au début, laisser parler en premier votre interlocuteur.
* Dès le début, votre interlocuteur doit sentir que vous êtes souriant et posé. Aussi, au téléphone le sourire s'entend au bout du fil. Ne surtout pas être stressé.
* Votre interlocuteur doit sentir qu'il est possible de vous faire confiance
* Montrer que vous êtes compétent, efficace avec un bon relationnel.
* Ne dites pas "je ne sais pas " ou " je ne connais pas" mais "j'ai eu l'occasion de le voir, je peux m'y remettre facilement" ou " je peux être opérationnelle rapidement car je suis à l'aise avec la technique et j'apprends vite" ou " je peux être opérationnelle rapidement car j'ai eu l'occasion de le voir" etc...
* Ne pas être timide ou complexé : vous parlez bien le français donc vous pouvez être à l'aise. Aussi, vous êtes rassurant donc il n'y a pas de raison de ne pas convaincre.
* Dire que vous êtes certifié et que votre certification est récente: que c'est frais dans votre tête que vous êtes à jour.
* Etre souple
  1. **ÉCOUTER**
* Bien écouter son interlocuteur, ne pas lui couper la parole.
* Ne pas hésiter à lui demander de reformuler la question si on n’a pas compris. Car c'est très mauvais de répondre à une question qu'on a mal comprise : c'est perdu d'avance!
  1. **SYNTHÉTIQUE:**
* Ne surtout pas parler de toutes les missions, aller à l'essentiel en parlant juste des missions les plus significatives.



7/6



* Parler très brièvement du projet et parler surtout de vos tâches : qu'est-ce que vous faites? Si on vous coupe la parole, n'oubliez pas de reprendre votre argumentaire là ou vous vous êtes arrêté.
* Parler des technologies, des framework, que vous utilisez et citer les versions
* Ne pas aller dans tous les sens : aller à l'essentiel en orientant votre discours vers un cadre professionnel et non pas privé.
* Ne pas parler des stages ou alors ne pas mettre en avant qu'il s'agit d'un stage mais d'une expérience professionnelle.



8/6



**d. MOTIVE:**

* Vous devez vous "vendre" de façon mesurée : lui donner envie de travailler avec vous
* Poser des questions : vos questions ne doivent pas se focaliser sur le client mais sur le poste. Vous devez montrer un intérêt pour le poste et la mission et non pas sur le "potentiel" du client.
* Conclure en vous montrant intéressé.



9/6