Questions d'entretien pour Ingénieur FullStack Java/Angular

Table des matières

Programmation Orienté Objet

Encapsulation

L'encapsulation fournit aux objets la possibilité de masquer leurs caractéristiques internes et leur comportement. Chaque objet fournit un certain nombre de méthodes auxquelles d'autres objets peuvent accéder et qui modifient ses données internes. En Java, il y a trois modificateurs d'accès : public, privé et protected. Chaque modificateur impose différents droits d'accès à d'autres classes, dans le même paquet ou dans des paquetages externes. Certains des avantages de l'utilisation de l'encapsulation sont énumérés ci-dessous :

- L'état interne de chaque objet est protégé en masquant ses attributs.
- Cela augmente la facilité d'utilisation et la maintenance du code, car le comportement d'un objet peut être modifié ou étendu indépendamment.
- Il améliore la modularité en empêchant les objets d'interagir les uns avec les autres, d'une manière indésirable.

Polymorphisme

Le polymorphisme veut dire que le même service, aussi appelé opération ou méthode, peut avoir un comportement différent selon les situations.

Il consiste à fournir une interface unique à des entités pouvant avoir différents types

Héritage

L'héritage fournit un objet avec la capacité d'acquérir les champs et les méthodes d'une autre classe, appelée classe de base. L'héritage fournit la réutilisation du code et peut être utilisé pour ajouter des fonctionnalités supplémentaires à une classe existante, sans la modifier.

Abstraction

L'abstraction est un processus consistant à masquer les détails de la l'implémentation à l'utilisateur. Seulement la fonctionnalité sera fournie à l'utilisateur.

L'abstraction est le processus qui consiste à séparer des idées d'instances spécifiques et à développer ainsi des classes en fonction de leurs propres fonctionnalités, au lieu de leurs détails d'implémentation. Java prend en charge la création et l'existence de classes abstraites exposant des interfaces, sans inclure l'implémentation réelle de toutes les méthodes. La technique d'abstraction vise à séparer les détails d'implémentation d'une classe de son comportement.

Différences entre l'abstraction et l'encapsulation

L'abstraction et l'encapsulation sont des concepts complémentaires. D'une part, l'abstraction se concentre sur le comportement d'un objet. D'autre part, l'encapsulation se concentre sur l'implémentation du comportement d'un objet. L'encapsulation est généralement réalisée en cachant des informations sur l'état interne d'un objet et peut donc être considérée comme une stratégie utilisée pour fournir l'abstraction.

Design Pattern

Un patron de conception est un concept de génie logiciel destiné à résoudre des problèmes récurrents suivant le paradigme objet.

Fabrique (Factory)	Créer un objet dont le type dépend du contexte
Fabrique abstraite (abstract Factory)	Fournir une interface unique pour instancier des objets d'une même famille sans avoir à connaître les classes à instancier
Monteur (Builder)	
Prototype (Prototype)	Création d'objet à partir d'un prototype
Singleton	Classe qui ne pourra avoir qu'une seule

(Singleton)	instance

Quelle est la différence entre une classe Interface et une classe abstraite ?

interface: pour implémenter un contrat avec plusieurs objets indépendants
Classe abstraite: Pour implémenter le même comportement ou un comportement différent parmi plusieurs objets associés

Envisagez d'utiliser des classes abstraites si:

1. Vous souhaitez partager du code entre plusieurs classes étroitement liées.
2. Vous vous attendez à ce que les classes qui étendent votre classe abstraite aient de nombreuses méthodes ou champs communs, ou nécessitent des modificateurs d'accès autres que publics (tels que protected et private).
3. Vous voulez déclarer des champs non statiques ou non finaux.

Envisagez d'utiliser des interfaces si :

1. Vous vous attendez à ce que des classes indépendantes implémenter votre interface. Par exemple, de nombreux objets indépendants peuvent implémenter l'interface `Serializable`.
2. Vous souhaitez spécifier le comportement d'un type de données particulier, mais ne vous inquiétez pas de savoir qui implémente son comportement.
3. Vous voulez profiter de l'héritage multiple de type.

classe abstraite établit "est une" relation avec des classes concrètes. L'interface fournit une capacité "a" pour les classes.

Java fournit et prend en charge la création de classes et d'interfaces abstraites . Les deux implémentations partagent certaines caractéristiques communes, mais elles diffèrent par les caractéristiques suivantes :

- Toutes les méthodes d'une interface sont implicitement abstraites. D'un autre côté, une classe abstraite peut contenir à la fois des méthodes abstraites et non abstraites.
- Une classe peut implémenter un certain nombre d'Interfaces, mais ne peut étendre qu'une seule classe abstraite.
- Pour qu'une classe puisse implémenter une interface, elle doit implémenter toutes ses méthodes déclarées. Cependant, une classe peut ne pas implémenter toutes les méthodes déclarées d'une classe abstraite. Bien que, dans ce cas, la sous-classe doit aussi être déclarée comme abstraite.
- Les classes abstraites peuvent implémenter des interfaces sans même prévoir l'implémentation de méthodes d'interface.
- Les variables déclarées dans une interface Java sont par défaut finales. Une classe abstraite peut contenir des variables non finales.
- Les membres d'une interface Java sont publics par défaut. Un membre d'une classe abstraite peut être privé, protégé ou public.

- Une interface est absolument abstraite et ne peut pas être instanciée. Une classe abstraite ne peut pas non plus être instanciée, mais peut être invoquée si elle contient une méthode principale.

Questions générales sur Java

Qu'est-ce que JVM ? Pourquoi Java est appelé "Platform Independent Programming Language » ?

Une machine virtuelle Java (JVM) est une machine virtuelle de processus pouvant exécuter le bytecode Java . Chaque fichier source Java est compilé dans un fichier bytecode, qui est exécuté par la JVM. Java a été conçu pour permettre la création de programmes d'application pouvant être exécutés sur n'importe quelle plate-forme, sans avoir à être réécrits ou recompilés par le programmeur pour chaque plate-forme séparée. Une machine virtuelle Java rend cela possible, car elle est consciente des longueurs d'instructions spécifiques et des autres particularités de la plate-forme matérielle sous-jacente.

Quelle est la différence entre JDK et JRE ?

Java Runtime Environment (JRE) est essentiellement la machine virtuelle Java (JVM) sur laquelle vos programmes Java sont exécutés. Il inclut également des plugins de navigateur pour l'exécution de l'applet. Java Development Kit (JDK) est le kit de développement logiciel complet pour Java, y compris JRE, les compilateurs et les outils (JavaDoc et Java Debugger), permettant à un utilisateur de développer, compiler et exécuter des applications Java.

Que signifie le mot-clé « static » ? Pouvez-vous remplacer la méthode privée ou statique en Java ?

Le mot-clé static indique qu'une variable ou une méthode membre peut être accédée, sans nécessiter une instanciation de la classe à laquelle elle appartient. Un utilisateur ne peut pas remplacer les méthodes statiques dans Java , car la substitution de méthode est basée sur la liaison dynamique au moment de l'exécution et les méthodes statiques sont liées statiquement au moment de la compilation. Une méthode statique n'est associée à aucune instance d'une classe, de sorte que le concept n'est pas applicable.

Pouvez-vous accéder à une variable non statique dans un contexte static ?

Une variable statique en Java appartient à sa classe et sa valeur reste la même pour toutes ses instances. Une variable statique est initialisée lorsque la classe

est chargée par la JVM. Si votre code tente d'accéder à une variable non statique, sans aucune instance, le compilateur se plaindra, car ces variables ne sont pas encore créées et ne sont associées à aucune instance.

Quels sont les types de données supportés par Java ? Qu'est-ce que Autoboxing et Unboxing ?

Les huit types de données primitifs pris en charge par le langage de programmation Java sont les suivants :

- Octet
- Court
- Int
- Long
- Float
- Double
- Boolean
- Char

Autoboxing est la conversion automatique faite par le compilateur Java entre les types primitifs et leurs classes wrapper d'objets correspondantes. Par exemple, le compilateur convertit un Int en Integer , un double en Double , et ainsi de suite. Si la conversion se passe dans l'autre sens, cette opération s'appelle unboxing.

Qu'est-ce que la fonction Overriding et Overloading dans Java ?

La surcharge de méthode en Java se produit lorsque deux ou plusieurs méthodes dans la même classe ont exactement le même nom, mais des paramètres différents. D'un autre côté, le remplacement de méthode est défini comme le cas où une classe enfant redéfinit la même méthode qu'une classe parente. Les méthodes surchargées doivent avoir le même nom, la même liste d'arguments et le même type de retour. La méthode prioritaire ne peut pas limiter l'accès de la méthode qu'elle remplace.

Qu'est-ce qu'un constructeur, une surchargeuse de constructeur en Java et un constructeur de copie?

Un constructeur est appelé lorsqu'un nouvel objet est créé. Chaque classe a un constructeur . Dans le cas où le programmeur ne fournit pas un constructeur pour une classe, le compilateur Java (Javac) crée un constructeur par défaut pour cette classe. La surcharge du constructeur est similaire à la surcharge de méthode en Java. Différents constructeurs peuvent être créés pour une seule

classe. Chaque constructeur doit avoir sa propre liste de paramètres unique. Enfin, Java prend en charge les constructeurs de copie comme C ++, mais la différence réside dans le fait que Java ne crée pas de constructeur de copie par défaut si vous n'écrivez pas le vôtre.

Java prend-il en charge l'héritage multiple ?

Non, Java ne prend pas en charge l'héritage multiple. Chaque classe est capable de s'étendre uniquement sur une classe, mais est capable d'implémenter plus d'une interface.

Qu'est-ce qui passe par référence et passe par valeur ?

Lorsqu'un objet est passé par valeur, cela signifie qu'une copie de l'objet est transmise. Ainsi, même si des modifications sont apportées à cet objet, cela n'affecte pas la valeur d'origine. Quand un objet est passé par référence, cela signifie que l'objet réel n'est pas passé, plutôt qu'une référence de l'objet est passée. Ainsi, toutes les modifications apportées par la méthode externe sont également reflétées dans tous les endroits.

Quelle est la différence entre les processus et les threads ?

Un processus est une exécution d'un programme, alors qu'un Thread est une seule séquence d'exécution dans un processus. Un processus peut contenir plusieurs threads. Un fil de discussion est parfois appelé un processus léger.

Quelles sont les interfaces de base de Java Collections Framework ?

Java Collections Framework fournit un ensemble bien conçu d'interfaces et de classes qui prennent en charge les opérations sur une collection d'objets. Les interfaces les plus basiques qui résident dans le Java Collections Framework sont :

- Collection , qui représente un groupe d'objets connus comme ses éléments.
- Set , qui est une collection qui ne peut pas contenir d'éléments en double.
- List , qui est une collection ordonnée et peut contenir des éléments en double.
- Map , qui est un objet qui mappe les clés sur des valeurs et ne peut pas contenir de clés en double.

Comment HashMap fonctionne en Java ?

Un HashMap dans Java stocke les paires clé-valeur . Le HashMap nécessite une fonction de hachage et utilise des méthodes hashCode et égales, afin de mettre et de récupérer des éléments de et vers la collection respectivement. Lorsque la méthode put est appelée, HashMap calcule la valeur de hachage de la clé et stocke la paire dans l'index approprié dans la collection. Si la clé existe, sa valeur est mise à jour avec la nouvelle valeur. Certaines caractéristiques importantes d'un HashMap sont sa capacité, son facteur de charge et le redimensionnement du seuil.

Quelles différences existent entre HashMap et Hashtable ?

Les deux classes HashMap et Hashtable implémentent l'interface Map et ont donc des caractéristiques très similaires. Cependant, ils diffèrent dans les caractéristiques suivantes :

- Un HashMap permet l'existence de clés et de valeurs NULL, alors qu'un Hashtable n'autorise pas les clés NULL, ni les valeurs nulles.
- Une Hashtable est synchronisée, alors qu'une HashMap ne l'est pas. Ainsi, HashMap est préféré dans les environnements à un seul thread, tandis qu'un Hashtable est adapté aux environnements multithreads.
- Un HashMap fournit son ensemble de clés et une application Java peut itérer sur eux. Ainsi, un HashMap est fail-fast. D'un autre côté, un Hashtable fournit une énumération de ses clés.
- La classe Hashtable est considérée comme une classe héritée.

Quelle est la différence entre Array et ArrayList? Quand utiliserez-vous Array sur ArrayList ?

Les classes Array et ArrayList diffèrent sur les fonctionnalités suivantes :

- Les tableaux peuvent contenir primitive ou des objets, alors qu'une ArrayList peut contenir que des objets.
- Les tableaux ont une taille fixe, tandis qu'un ArrayList est dynamique.
- Un ArrayList fournit plus de méthodes et de fonctionnalités, telles que addAll, removeAll, itérateur, etc.
- Pour une liste de types de données primitifs, les collections utilisent l'autoboxing pour réduire l'effort de codage. Toutefois, cette approche les ralentit lorsque vous travaillez sur des types de données primitifs de taille fixe.

Quelle est la différence entre ArrayList et LinkedList?

Les classes ArrayList et LinkedList implémentent l'interface List, mais elles diffèrent sur les fonctionnalités suivantes:

- Une ArrayList est une structure de données basée sur un index soutenue par un tableau . Il fournit un accès aléatoire à ses éléments avec une performance égale à O (1). D'un autre côté, une LinkedList stocke ses données sous forme de liste d'éléments et chaque élément est lié à son élément précédent et suivant. Dans ce cas, l'opération de recherche d'un élément a un temps d'exécution égal à O (n).
- Les opérations d'insertion, d'ajout et de suppression d'un élément sont plus rapides dans une LinkedList que dans une ArrayList , car il n'est pas nécessaire de redimensionner un tableau ou de mettre à jour l'index lorsqu'un élément est ajouté dans une position arbitraire.
- Une LinkedList consomme plus de mémoire qu'une ArrayList , car chaque nœud d'une LinkedList stocke deux références, une pour son élément précédent et une pour son élément suivant. Consultez aussi notre article ArrayList vs. LinkedList .

Quelle est la différence entre HashSet et TreeSet?

Le HashSet est implémenté en utilisant une table de hachage et ainsi, ses éléments ne sont pas commandés. Les méthodes add, remove et contains d'un HashSet ont une complexité de temps constante O (1). D'un autre côté, un TreeSet est implémenté en utilisant une structure arborescente. Les éléments d'un TreeSet sont triés, et ainsi, les méthodes add, remove et contains ont une complexité temporelle de O (logn).

Quels sont les deux types d'exceptions en Java ? Quelles sont les différences entre eux ?

Java a deux types d’exceptions : les exceptions vérifiées et les exceptions non contrôlées. Les exceptions non vérifiées n'ont pas besoin d'être déclarées dans une méthode ou dans la clause throws d'un constructeur, si elles peuvent être lancées par l'exécution de la méthode ou du constructeur, et propagées en dehors de la méthode ou de la frontière du constructeur. D'un autre côté, les exceptions vérifiées doivent être déclarées dans une méthode ou dans la clause throws d'un constructeur. Voir ici pour des conseils sur la gestion des exceptions Java .

Quelle est la différence entre Exception et Error dans Java ?

Les classes Exception et Error sont les deux sous-classes de la classe Throwable . La classe Exception est utilisée pour des conditions exceptionnelles que le programme d'un utilisateur doit attraper. La classe Error définit les exceptions qui ne sont pas censées être interceptées par le programme utilisateur.

• Questions sur Java 8

Quelles nouvelles fonctionnalités ont été ajoutées dans Java 8 ?

Java 8 est livré avec plusieurs nouvelles fonctionnalités, mais les plus importantes sont les suivantes :

- **Lambda Expressions** - une nouvelle fonctionnalité de langage permettant de traiter des actions en tant qu'objets
- **Références de méthode** - permettent de définir des expressions lambda en se référant directement aux méthodes en utilisant leurs noms
- ***Optionnal*** - classe d'encapsulation spéciale utilisée pour exprimer l'optionalité
- **Interface fonctionnelle** - une interface avec une méthode abstraite maximale, l'implémentation peut être fournie en utilisant une expression Lambda
- **Méthodes par défaut** - nous permettent d'ajouter des implémentations complètes dans les interfaces en plus des méthodes abstraites
- **Nashorn, JavaScript Engine** - **Moteur** Java pour exécuter et évaluer le code JavaScript
- ***Stream* API** - une classe d'itération spéciale qui permet de traiter les collections d'objets de manière fonctionnelle
- **Date API** - une **API de date** JodaTime améliorée et immuable

Parallèlement à ces nouvelles fonctionnalités, de nombreuses améliorations sont apportées sous le capot, tant au niveau du compilateur que de la JVM.

Qu'est-ce qu'une expression Lambda et dans quel cas est-elle utilisée ?

En termes très simples, une expression lambda est une fonction qui peut être référencée et transmise en tant qu'objet.

Les expressions lambda introduisent un traitement de style fonctionnel en Java et facilitent l'écriture de code compact et facile à lire.

Pour cette raison, les expressions lambda remplacent naturellement les classes anonymes en tant qu'arguments de méthode. L'une de leurs principales

utilisations est de définir des implémentations en ligne d'interfaces fonctionnelles.

Qu'est-ce qu'un Stream ? En quoi diffère-t-il d'une collection ?

C'est une séquence d'éléments sur laquelle on peut effectuer un groupe d'opérations de manière séquentielle ou parallèle.

En termes simples, un flux est un itérateur dont le rôle est d'accepter un ensemble d'actions à appliquer sur chacun des éléments qu'il contient.

Le flux représente une séquence d'objets provenant d'une source telle qu'une collection, qui prend en charge les opérations d'agrégation. Ils ont été conçus pour rendre le traitement des collections simple et concis. Contrairement aux collections, la logique d'itération est implémentée dans le flux, donc nous pouvons utiliser des méthodes comme *map* et *flatMap* pour effectuer un traitement déclaratif.

Une autre différence est que l'API *Stream* est fluide et permet la mise en pipeline :

```
1   int sum = Arrays.stream(new int[]{1, 2, 3})
2     .filter(i -> i >= 2)
3     .map(i -> i * 3)
4     .sum();
```

Et encore une autre distinction importante des collections est que les flux sont intrinsèquement chargés et traités paresseusement.

Parlez-nous de la nouvelle API Date and Time dans Java 8

Un problème de longue date pour les développeurs Java a été le support inadéquat des manipulations de date et d'heure requises par les développeurs ordinaires.

Les classes existantes telles que *java.util.Date* et *SimpleDateFormatter* ne sont pas thread-safe, ce qui entraîne des problèmes de concurrence potentiels pour les utilisateurs.

La mauvaise conception de l'API est également une réalité dans l'ancienne API Java Data. Voici un exemple rapide - les années
dans *java.util.Date* commencent à 1900, les mois commencent à 1, et les jours commencent à 0 ce qui n'est pas très intuitif.

Ces problèmes et plusieurs autres ont conduit à la popularité des bibliothèques de date et d'heure tierces, telles que Joda-Time.

Afin de résoudre ces problèmes et d'offrir une meilleure prise en charge dans JDK, une nouvelle API de date et d'heure, libre de ces problèmes, a été conçue pour Java SE 8 sous le package *java.time* .

- ## Spring

Qu'est-ce que Spring Framework?

Spring est le framework le plus largement utilisé pour le développement d'applications Java Enterprise Edition. Les fonctionnalités principales de Spring peuvent être utilisées dans le développement de toute application Java.

Nous pouvons utiliser ses extensions pour construire diverses applications Web au-dessus de la plate-forme Java EE, ou nous pouvons simplement utiliser ses dispositions d'injection de dépendance dans des applications autonomes simples.

Qu'est-ce que Spring Boot ?

Spring Boot est un projet qui fournit un ensemble préconfiguré de frameworks pour réduire la configuration standard afin que vous puissiez avoir une application Spring opérationnelle avec la plus petite quantité de code.

Quels sont les avantages de l'utilisation de Spring?

Cibles printanières pour faciliter le développement de Java EE. Voici les avantages de l'utiliser:

- **Léger:** il y a un léger surcoût d'utilisation du framework en développement
- **Inversion de contrôle (IoC): le** conteneur Spring prend en charge les dépendances de câblage de divers objets, au lieu de créer ou de rechercher des objets dépendants
- **Programmation Orientée Aspect (AOP):** Spring prend en charge AOP pour séparer la logique métier des services système
- **Conteneur IoC:** gère le cycle de vie du bean Spring et les configurations spécifiques au projet
- **Framework MVC:** utilisé pour créer des applications Web ou des services Web RESTful, capables de renvoyer des réponses XML / JSON

- **Gestion des transactions:** réduit la quantité de code de plaque de chaudière dans les opérations JDBC, le téléchargement de fichiers, etc., soit à l'aide d'annotations Java, soit à l'aide du fichier de configuration XML Spring Bean.
- **Gestion des exceptions:** Spring fournit une API pratique pour traduire les exceptions spécifiques à la technologie en exceptions non contrôlées

Quels sous-projets de printemps connaissez-vous? Décrivez-les brièvement.

- **Core** - un module clé qui fournit des parties fondamentales du cadre, comme IoC ou DI
- **JDBC** - ce module active une couche d'abstraction JDBC qui supprime le besoin de codage JDBC pour des bases de données de fournisseurs spécifiques
- **Intégration ORM** - fournit des couches d'intégration pour les API de mappage objet-relationnelles populaires, telles que JPA, JDO et Hibernate
- **Web** - un module d'intégration orienté Web, fournissant un téléchargement de fichiers en plusieurs parties, des écouteurs Servlet et des fonctionnalités de contexte d'application orientées Web
- **Framework MVC** - un module Web implémentant le modèle de conception Model View Controller
- **Module AOP** - implémentation de programmation orientée aspect permettant la définition d'intercepteurs de méthodes et de pointcoins propres

Qu'est-ce que Dependency Injection?

L'injection de dépendances, un aspect d'Inversion de contrôle (IoC), est un concept général qui indique que vous ne créez pas vos objets manuellement mais que vous décrivez comment ils doivent être créés. Un conteneur IoC instanciera les classes requises si nécessaire.

Comment pouvons-nous injecter des beans au spring?

Quelques options différentes existent:

- Setter Injection
- Constructeur Injection
- Injection de champ

La configuration peut être effectuée à l'aide de fichiers XML ou d'annotations.

Quelle est la meilleure façon d'injecter des beans et pourquoi?

L'approche recommandée consiste à utiliser les arguments constructor pour les dépendances obligatoires et les setters pour les dépendances facultatives. L'injection de constructeur permet d'injecter des valeurs dans des champs immuables et facilite les tests.

Quelle est la différence entre *BeanFactory* et *ApplicationContext* ?

BeanFactory est une interface représentant un conteneur qui fournit et gère les instances du bean. L'implémentation par défaut instancie paresseusement les beans quand *getBean ()* est appelé.

ApplicationContext est une interface représentant un conteneur contenant toutes les informations, les métadonnées et les beans dans l'application. Il étend également l' interface *BeanFactory* mais l'implémentation par défaut instancie les beans avec impatience au démarrage de l'application. Ce comportement peut être remplacé pour les beans individuels.

Pour toutes les différences, veuillez vous référer à la référence .

Qu'est-ce qu'un bean de Spring?

Les beans de spring sont des objets Java qui sont initialisés par le conteneur Spring IoC.

Quelle est la portée du bean par défaut dans Spring framework?

Par défaut, un Spring Bean est initialisé en tant que *singleton* .

Comment définir la portée d'un bean?

Pour définir la portée de Spring Bean, nous pouvons utiliser l' annotation *@Scope* ou l'attribut "scope" dans les fichiers de configuration XML. Il existe cinq portées prises en charge:

- **singleton**
- **prototype**
- **demande**
- **session**
- **session globale**

Les Beans singleton sont-elles sûres?

Non, les beans singleton ne sont pas adaptés aux threads, car la sécurité des threads concerne l'exécution, tandis que le singleton est un modèle de

conception centré sur la création. La sécurité des threads dépend uniquement de l'implémentation du bean elle-même.

À quoi ressemble le cycle de vie du bean de Spring?

Tout d'abord, un bean Spring doit être instancié, en fonction de la définition du bean Java ou XML. Il peut également être nécessaire d'effectuer une initialisation pour l'amener dans un état utilisable. Après cela, lorsque le bean n'est plus requis, il sera retiré du conteneur IoC.

Le cycle complet avec toutes les méthodes d'initialisation est montré sur l'image (source):

Quelle est la configuration basée sur Java de Spring ?

C'est l'une des façons de configurer les applications basées sur Spring de manière sécurisée. C'est une alternative à la configuration basée sur XML.

Pouvons-nous avoir plusieurs fichiers de configuration Spring dans un projet?

Oui, dans les grands projets, plusieurs configurations de ressorts sont recommandées pour augmenter la maintenabilité et la modularité.

Vous pouvez charger plusieurs fichiers de configuration Java:

```
1  @Configuration
```

```
2  @Import({MainConfig.class, SchedulerConfig.class})
3  public class AppConfig {
```

Ou charger un fichier XML qui contiendra toutes les autres configurations:

```
1  ApplicationContext context = new ClassPathXmlApplicationContext("spring-all.xml");
```

Et à l'intérieur de ce fichier XML, vous aurez:

```
1  <import resource="main.xml"/>
2  <import resource="scheduler.xml"/>
```

Qu'est-ce que Spring Security?

Spring Security est un module distinct du framework Spring qui se concentre sur la fourniture de méthodes d'authentification et d'autorisation dans les applications Java. Il prend également en charge la plupart des vulnérabilités de sécurité courantes, telles que les attaques CSRF.

Pour utiliser Spring Security dans les applications Web, vous pouvez commencer par une simple annotation: *@EnableWebSecurity* .

Vous pouvez trouver toute la série d'articles liés à la sécurité sur Baeldung .

Nommez certains des modèles de conception utilisés dans le cadre de Spring ?

- **Singleton Pattern:** Haricots à l'échelle d'un singleton
- **Modèle d'usine:** classes d'usine de haricots
- **Patron de prototype:** Haricots à l'échelle du prototype
- **Modèle d'adaptateur:** Spring Web et Spring MVC
- **Modèle de proxy:** prise en charge de la programmation orientée aspect ressort
- **Modèle de modèle de modèle:** *JdbcTemplate* , *HibernateTemplate,* etc.
- **Contrôleur frontal:** Spring MVC *DispatcherServlet*
- **Objet d'accès aux données: prise en charge de** Spring DAO
- **Contrôleur de vue du modèle:** Spring MVC

Comment fonctionne le *scope Prototype* ?

Le *prototype* Scope signifie que chaque fois que vous appelez une instance du Bean, Spring crée une nouvelle instance et la renvoie. Cela diffère de la portée *singleton* par défaut , où une seule instance d'objet est instanciée une fois par conteneur IoC de Spring.

Quelle est la différence entre @Controller , @Component , @Repository et @Service Annotations au printemps?

Selon la documentation officielle de Spring, *@Component* est un stéréotype générique pour tout composant géré par Spring. *@Repository* , *@Service* et *@Controller* sont des spécialisations de *@Component* pour des cas d'utilisation plus spécifiques, par exemple, dans les couches de persistance, de service et de présentation, respectivement.

Jetons un coup d'œil aux cas d'utilisation spécifiques des trois derniers:

- ***@ Controller*** - indique que la classe joue le rôle de contrôleur et détecte les annotations *@RequestMapping* dans la classe
- ***@ Service*** - indique que la classe contient la logique métier et appelle les méthodes dans la couche du référentiel
- ***@ Repository*** - indique que la classe définit un référentiel de données; son travail consiste à intercepter les exceptions spécifiques à la plate-forme et à les relancer comme l'une des exceptions non vérifiées unifiées de Spring

Qu'est-ce que la programmation orientée aspect?

Les aspects permettent la modularisation des problèmes transversaux tels que la gestion des transactions qui couvrent plusieurs types et objets en ajoutant un comportement supplémentaire au code existant sans modifier les classes affectées.

Voici l'exemple de la consignation du temps d'exécution basée sur les aspects .

Quels sont *Aspect* , *Advice* , *Pointcut* et *JoinPoint* dans AOP?

- ***Aspect*** : une classe qui implémente des préoccupations transversales, telles que la gestion des transactions
- ***Conseil*** : les méthodes qui s'exécutent lorsqu'un *JoinPoint* spécifiqueavec *Pointcut*correspondantest atteint dans l'application
- ***Pointcut*** : un ensemble d'expressions régulières qui correspondent à *JoinPoint* pour déterminer si le *conseil* doit être exécuté ou non
- ***JoinPoint*** : un point lors de l'exécution d'un programme, tel que l'exécution d'une méthode ou la gestion d'une exception

Les annotations de Spring

https://springframework.guru/spring-framework-annotations/

@Autowired

Cette annotation est appliquée aux champs, aux méthodes setter et aux constructeurs. L' annotation **@Autowired** injecte la dépendance d'objet implicitement.
Lorsque vous utilisez **@Autowired** sur des champs et que vous transmettez les valeurs des champs à l'aide du nom de la propriété, Spring affecte automatiquement les champs aux valeurs transmises.

@Configuration

Cette annotation est utilisée sur les classes qui définissent les beans. **@configuration** est un fichier XML de configuration analogique - c'est une configuration utilisant la classe Java. La classe Java annotée avec **@Configuration** est une configuration en elle-même et aura des méthodes pour instancier et configurer les dépendances.

@bean

Cette annotation est utilisée au niveau de la méthode. L' annotation @ Bean fonctionne avec @Configuration pour créer des beans Spring. Comme mentionné précédemment, @ Configuration aura des méthodes pour instancier et configurer les dépendances. Ces méthodes seront annotées avec @ Bean . La méthode annotée avec cette annotation fonctionne comme ID de bean et crée et renvoie le bean réel.

@lazy

Cette annotation est utilisée sur les classes de composants. Par défaut, toutes les dépendances auto-générées sont créées et configurées au démarrage. Mais si vous voulez initialiser un bean paresseusement, vous pouvez utiliser l' annotation @Lazy sur la classe. Cela signifie que le bean sera créé et initialisé seulement quand il est demandé pour la première fois. Vous pouvez également utiliser cette annotation sur les classes de @configuration . Ceci indique que toutes les méthodes @ Bean dans @ Configuration doivent être paresseusement initialisées.

@ Component

Cette annotation est utilisée sur les classes pour indiquer un composant Spring. L' annotation @Component marque la classe Java en tant que bean ou dit composant afin que le mécanisme d'analyse des composants de Spring puisse être ajouté au contexte de l'application.

- ## Devops

Microservice:

Les microservices désignent à la fois une architecture et une approche de développement logiciel, qui consiste à décomposer les applications en éléments les plus simples, indépendants les uns des autres

Kafka:

Kafka est un système de messagerie de type publication/abonnement gérée par la fondation Apache.

Docker:

Docker est un projet Open Source proposant une surcouche qui automatise et simplifie le déploiement d'applications dans des conteneurs virtuels.

Container:

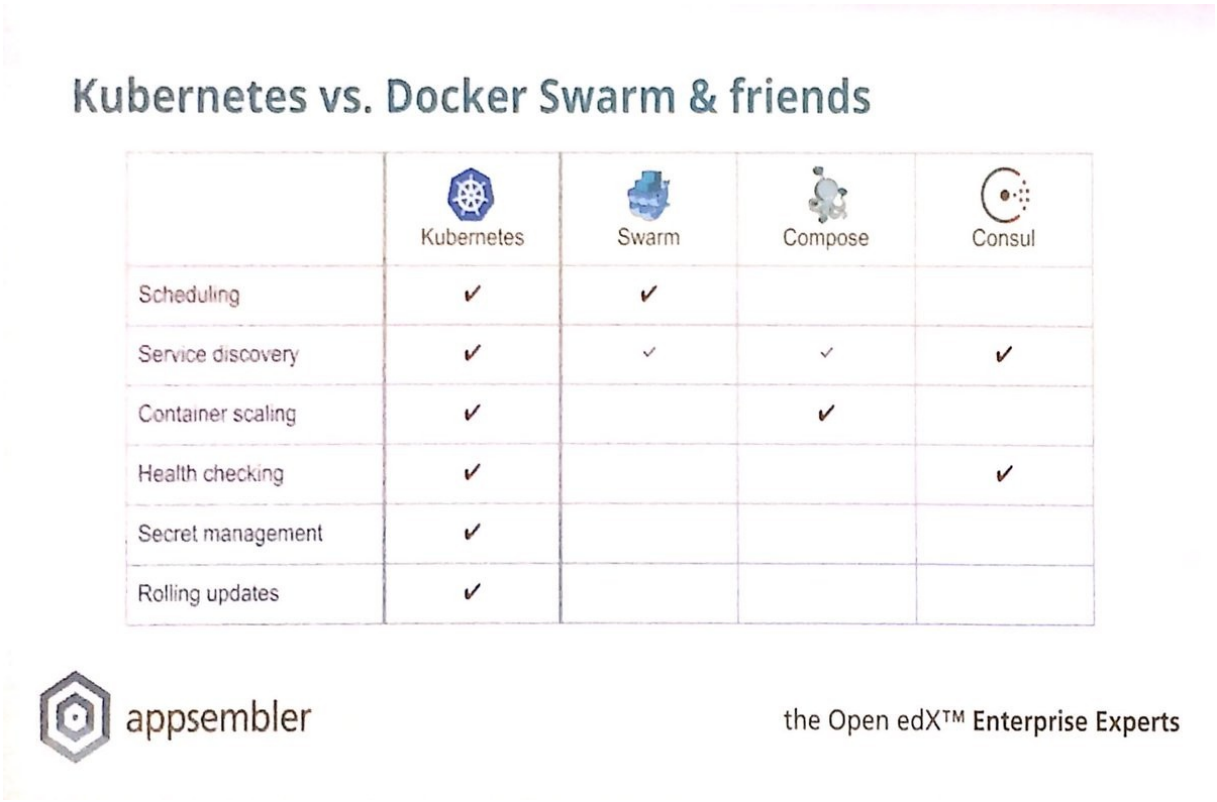
Un conteneur est une enveloppe virtuelle qui permet de packager une application avec tous les éléments dont elle a besoin pour fonctionner : fichiers source, runtime, librairies, outils et fichiers. Ils sont packagés en un ensemble cohérent et prêt à être déployé sur un serveur et son OS.

Docker Swarm:

Docker Swarm est un outil conçu par Docker permettant de gérer un cluster de Container très facilement. En plus d'être simple à implémenter, Docker Swarm est extrêmement performant. Il peut supporter mille noeuds et cinquante mille container sans aucune dégradation de performance.

Kubernetes

Kubernetes est l'orchestrateur de conteneurs de référence, développé par google. Il facilite le déploiement, la résilience et la scalabilité de vos applications, y compris dans une stratégie hybride ou multicloud.



Maven:

Maven est un outil permettant d'automatiser la gestion de projets Java.

Il offre entre autres les fonctionnalités suivantes :

-Compilation et déploiement des applications Java (JAR, WAR)

-Gestion des librairies requises par l'application

-Exécution des tests unitaires

Jenkins:

Jenkins est un outil logiciel d'intégration continu. Il s'agit d'un logiciel open source, développé à l'aide du langage de programmation Java. Il permet de tester et de rapporter les changements effectués sur une large base de code en temps réel

Intégration continue:

L'intégration continue est une pratique de développement permettant aux développeurs d'apporter des changements à un code source dans un dossier partagé plusieurs fois par jour ou plus fréquemment. Les changements sont ensuite intégrés.

WebService

Service Web:

Web Services est une technologie de communication entre deux appareils électroniques sur Internet ou intranet utilisant la collecte de protocoles standards pour le World Wide Web (WWW), les données de service s'échangent typiquement au format XML ou JSON.

Un service Web standard utilisant les composants ci-dessous:

- XML (langage de balisage extensible)
- SOAP (Simple Object Access Protocol)
- UDDI (description universelle, découverte et intégration)
- WSDL (langage de description des Web)

XML signifie eXtensible Markup Language. L'invention concerne un mécanisme de représentation et d'échange de données uniforme qui définit un ensemble de règles pour coder des documents dans un format à la fois humain et lisible par une machine .

SOAP signifie Simple Object Access Protocol. SOAP est une spécification de protocole basée sur HTTP pour échanger des informations structurées sous forme XML, interagir sur tous les réseaux informatiques, en particulier HTTP (Hypertext Transfer Protocol).

UDDI signifie Universal Description, Discovery et Integration. UDDI est une plate-forme indépendante, UDDI est un protocole réseau basé sur XML, où le service Web enregistre leurs services Web et le client peut rechercher, trouver et invoquer des services Web. SOAP est le protocole de communication le plus populaire.

WSDL est l' abréviation de Web Services Description Language, WSDL est écrit au format XML et est utilisé comme métalangage standard pour décrire les services Web offerts. WSDL est un programme / langage qui décrit les services du fournisseur comme un ensemble de messages contenant des documents orientés document. ou des informations orientées sur la procédure. Le même service Web peut avoir plusieurs points de terminaison utilisant différents protocoles et quel protocole de communication est utilisé pour parler à ce service.

Types de services Web:

1. SOAP (Simple Object Access Protocol)

2. REST (transfert d'état représentationnel)

SOAP signifie Simple Object Access Protocol. SOAP est une spécification de protocole basée sur HTTP pour échanger des informations structurées sous forme XML, interagir sur tous les réseaux informatiques, en particulier HTTP (Hypertext Transfer Protocol).

REST signifie Representational State Transfer. REST est un principe architectural pour la création d'un service Web à l'aide du protocole HTTP à l'aide de méthodes HTTP de base telles que POST, GET, PUT et DELETE (CRUD - Créer, Lire, Mettre à jour et Supprimer).

Le service Web REST prend en charge des formats tels que JSON, XML, HTML et texte

- REST utilise des méthodes standard comme

Quelles sont les différences entre les services Web SOAP et les services Web REST ?

Le service Web SOAP et REST ne peut pas être comparé directement, voir ci-dessous les différences.

S.No	SAVON	DU REPOS
1	SOAP signifie **Simple Object Access Protocol** .	REST signifie **Representational State Transfer** .
2	SOAP est un protocole	REST est un *style architectural*
3	SOAP peut utiliser presque toutes les demandes de transport	REST utilise HTTP et HTTPS
4	SOAP prend en charge les appels complets d'état	REST est totalement opérations sans état
5	SOAP est un contrat plus strict sous la forme de WSDL	REST n'a pas de contrat strict

	SOAP définit sa propre sécurité	REST héritera de la sécurité du niveau de transport sur HTTP.
6	La réponse SOAP ne peut pas être mise en cache	La réponse REST peut être mise en cache
7	Le format SOAP sera limité à XML	REST prendra en charge des formats tels que XML, JSON, HTML, Text
8	API Java pour les services Web SOAP, JAX-WS	API Java pour les services Web REST, JAX-RS
9	SOAP nécessite un wrapper XML pour toutes les requêtes et réponses	REST est plus léger, cette raison REST est préparé pour les appareils mobiles
10	SOAP peu plus lent et moins préparé	REST est plus rapide que SOAP, plus préparé, simple à utiliser et léger

Hibernate

Définition ?

Hibernate est une solution open source de type ORM (Object Relational Mapping) qui permet de faciliter le développement de la couche persistance d'une application. Hibernate permet donc de représenter une base de données en objets Java et vice versa.

- `FetchType.LAZY` : indique que la relation doit être chargée à la demande ;
- `FetchType.EAGER` : indique que la relation doit être chargée en même temps que l'entité qui la porte.

Angular

Quelle est la différence entre la one-way binding et la two-way binding ?

- **one-way binding** implique que la variable scope dans le html sera définie sur la première valeur à laquelle son modèle est lié (assignée à)
- **two-way binding** implique que la variable scope changera sa valeur chaque fois que son modèle est assigné à un autre valeur

Dans 30 secondes, définissez Angular.

AngularJS est un framework JavaScript open-source conçu pour créer des applications de pages Web dynamiques uniques avec moins de lignes de code.

Lazy Loading Angular.

Charger les modules à la demande , Une fois que l'utilisateur naviguera vers cette route, Angular chargera le chunk du module correspondant.

Data binding.

- **Property binding:** est un mécanisme de Data Binding "one way",il permet de répercuter dans le DOM les valeurs des propriétés du Component

- **Event binding:** U

- **Module:** U

Composent Angular.

- **Module:** Un module Angular est un mécanisme permettant de regrouper des composants (mais aussi des services, directives, pipes etc...)

- **Component:** les composantes de base d'une application Angular : une application est une arborescence de plusieurs components.

- **Directive:** Un

- **Service:** Un

- **Pipe:** Un

Mentionnez quelques avantages de l'utilisation du framework AngularJS.

Certains des principaux avantages de l'utilisation du framework AngularJS comprennent:

- Il fournit une excellente expérience de type "desktop" à l'utilisateur final.
- En libérant le développeur de devoir enregistrer les rappels manuellement ou d'écrire des tâches répétitives de manipulation DOM de bas niveau, il économise des mois de développement.
- En séparant la manipulation DOM de la logique applicative, il rend le code modulaire et facile à tester.

- Il prend en charge la liaison de données bidirectionnelle.

Qu'est-ce que l'injection de dépendance?

L'injection de dépendances est le processus par lequel les objets dépendants sont injectés plutôt que d'être créés par le consommateur.

Qu'est-ce qu'une Single Page Application (SPA)?

SPA est le concept selon lequel les pages sont chargées à partir du serveur, non pas en effectuant des publications, mais en créant une seule page de shell ou une page maître et en chargeant les pages Web dans la page maître.

Expliquer les décorateurs de composants dans Angular4.

Un décorateur est le concept de base lors du développement d'un cadre angulaire avec la version 2 et supérieure. Il pourrait bientôt devenir une fonctionnalité du langage de base pour JavaScript. Dans l'angle 4, les décorateurs sont largement utilisés et sont également utilisés pour compiler un code. Il y a 4 types de décorateurs différents:

Décorateurs de classe

Décorateurs de propriétés

Décorateurs de méthode

Décorateurs de paramètres