Fiche 2 : Architecture Logicielle

BUT 2 Info : R4.01

kamel.bouchefra@univ-paris13.fr

2024 - 2025

Nom - Prénom(s) :

Règles du TP

- Les travaux sont à réaliser de manière autonome, **individuelle**.
- Tout support, recherches internet **encouragés**.
- Il faut réaliser vos projets dans un répertoire **de votre dossier « personnel »**.
- Il faut sauvegarder vos projets dans un répertoire **du dossier « mes montages »**.

Cet énoncé est une introduction au développement d'applications « RESTful » :

- Des applications client - serveur.
- Exécutant des requêtes HTTP.

Les développements abordés incluent :

- la création d'un projet Spring Boot ;
- la configuration du fichier de propriétés ;
- l'ajout de dépendances ;
- l'ajout de classes ;
- L'exécution mettant en œuvre un navigateur Internet.

1 Exercice 1

L'objectif de cet exercice est de reproduire l'application du support de en cours : pages 47 à 59.

1. Création d'un projet, en utilisant « https://start.spring.io/ »
   - Utilisant « Gradle », « Java ».
   - Sans dépendances.

- ≪ Spring Boot 3.4.4 ≫.
- Avec ces metadata :
  - Group : ≪ fr.info ≫
  - Artifact : ≪ annotations ≫
  - Nom : ≪ annotations ≫
  - Description : ≪ Demo Annotations ≫
  - Paquetage : ≪ fr.info.annotations ≫.
- Packaging : ≪ jar ≫.
- Java : ≪ 17 ≫.
- On obtient un fichier ≪ zip ≫, qu'il faut transférer dans son répertoire de travail, où on le décompresse.

2. En vous plaçant à la racine du projet, exécutez la commande suivante dans le terminal : ≪ **tree** ≫. On peut voir l'arborescence créée. Notez l'emplacement des fichiers :
   - ≪ **build.gradle** ≫ : Le fichier où on spécifie les dépendances.
   - ≪ **AnnotationsApplication.java** ≫ : La classe qui contient la méthode principale.
   - ≪ **application.properties** ≫ : Le fichier des propriétés.
3. Chargez le projet dans IntellijIDEA :
   - Le dossier du projet doit se trouver dans un répertoire de votre dossier personnel.
   - Exécutez la commande suivante dans votre terminal : ≪ **intellij2** ≫.
   - Importez le projet Spring Boot :
     - Faire ≪ **open** ≫ : Placez-vous à la racine du projet, puis validez en appuyant sur ≪ OK ≫.
     - Cochez la case ≪ Trust projects in . . . ≫, puis appuyez sur ≪ Trust Project ≫
   - L'IDE scanne le répertoire et met en place l'environnement correspondant.
   - Localisez dans l'arborescence du projet les trois fichiers mentionnés au point 2 ci-dessus. Vous pouvez les ouvrir.
   - Première exécution : Faire un ≪ clic droit ≫ avec la souris sur le fichier java contenant la méthode principale. Puis faites ≪ Run ≫.
4. Premier développement :
   - Modifiez les fichiers de configuration :
     - Ajoutez un numéro de port pour le serveur.
     - Ajoutez cette dépendance : ≪ implementation 'org.springframework.boot :spring-boot-starter-web' ≫
   - Ajoutez un contrôleur :
     - Au niveau du répertoire contenant le programme Java, faites un clic droit, puis ≪ new Package ≫.
     - Créez le paquetage ≪ controleur ≫.
     - Puis, dans ce paquetage, avec ≪ new, puis Java Class ≫, créez la classe ≪ Controleur.java ≫.
     - Reprenez le code donnée dans le support de cours, page 54.

- Exécutez l'application.

5. Second développement :
   - Réalisez l'application décrite à page 56 du support de cours.
   - Exécutez l'application.
6. Troisième développement :
   - Réalisez l'application décrite à page 58 du support de cours.
   - Exécutez l'application.

2 Ré-écriture de l'application

L'objectif de cette partie est de développer une application que l'on pourrait plus facilement étendre.

2.1 Première étape

1. Commencez par dupliquer la classe « Controleur » :
   a) Modifier le nom de la classe initiale : Appelez-là « InterfaceCtrl ».
   b) Donnez ce nom à la copie : « ImplementationCtrl ».
2. Transformez la classe « InterfaceCtrl » en une interface :
   a) On ne change pas les *prototypes*.
   b) Supprimez l'annotation « @RestController » ;
   c) Supprimez l'annotation « @Value » ;
3. Faites de la classe « ImplementationCtrl » une implémentation de l'interface :
   a) On garde l'annotation « @RestController » ;
   b) On garde l'annotation « @Value » ;
   c) On garde les *méthodes* : Il faudra ajouter « @Override » aux prototypes.
4. Exécutez l'application.
5. Dans la classe « Joueur », mettez en commentaire la méthode « getNom » :
   a) Ré-exécutez l'application en utilisant le point de terminaison en rapport avec « joueur ».
   b) Prenez le temps de comprendre le fonctionnement ...

2.2 Seconde étape

1. Dans « **build.gradle** », **ajoutez** les dépendances suivantes :
   - « compileOnly 'org.projectlombok :lombok' »
   - « annotationProcessor 'org.projectlombok :lombok' »
2. Dans la classe « **joueur** », **ajoutez** :
   - Cet import : « lombok.Data »
   - Cette annotation à la classe : « @Data »
3. Dans la classe « Joueur », mettez en commentaire la méthode « getNom » :
   a) Réexécutez l'application en utilisant le point de terminaison en rapport avec « joueur ».
   b) Prenez le temps de comprendre le fonctionnement ...

2.3 Extension de l'application

1. Dans la classe « Joueur », **ajoutez** :
   a) Les attributs **String** : « equipier », « contact ».
   b) Un constructeur qui initialise les trois attributs. On garde le premier constructeur.
2. Dans l'interface « InterfaceCtrl », **ajoutez** :
   a) Un point de terminaison avec les trois paramètres : « nom », « equipier », « contact ». Ce point de terminaison ressemble à ceci :

      /bienvenue/joueur/nom/equipier/contact
3. Dans « ImplementationCtrl », **ajoutez** :
   a) L'implémentation de la méthode en rapport avec le point de terminaison ajouté dans le contrôleur.
4. Exécutez l'application.

2.4 Finalisation de l'application

1. Ajoutez dans l'interface le point de terminaison suivant :

```
1
2 @GetMapping("/bienvenue/joueurs")
3   public ArrayList<Joueur> liste();
```

2. Dans la classe « ImplementationCtrl », **ajoutez** :
   a) L'attribut « listeJoueurs », de type : **ArrayList<Joueur>**
   b) Un constructeur pour créer cet attribut.
   c) Une méthode public void init() : Qui doit être appelée dans le constructeur. Dans la méthode init :
      i. On doit créer ces trois valeurs :

```
1 String n[]={"tom","olive","tintin","asterix","lucky luke","
  gargamel"};
2 String e[]={"jerry","popeye","milou","obelix","jolly jumper","
  azrael"};
3 String c[]={"tom.jerry@gmail.com","olive.popeye@gmail.com","
  tintin.milou@gmail.com",
4             "asterix.obelix@gmail.com","lucky.jolly@gmail.com","
  gargamel.azrael@gmail.com"};
```

      ii. **Ajoutez le code** pour :
         A. Créer les joueurs en utilisant les tableaux de valeurs donnés.
         B. Ajouter chaque joueur dans la liste listeJoueurs.
   d) Implémenter le point de terminaison précédent (voir le paragraphe §2.4.1).
   e) Exécutez l'application.