I - Objectifs

L'objectif de ce TP est d'exposer des objets via un service `REST` à l'aide de `Spring MVC` en les sérialisant
et dé-sérialisant automatiquement en flux `XML`.

Le service sera associé à une `URL` du type `/SEPA/resume` avec les comportements suivants :

URL	Méthode	Action
`/SEPA/resume`	`GET`	renvoie un flux `XML` contenant la liste des transactions identifiée par `PmtId` et `InstdAmt`.
`/SEPA/resume/id`	`GET`	renvoie un flux `XML` décrivant le détail de la transactions d'identifiant Id avec `Id` = `PmtId`, (éviter d'utiliser des doublons).
`/SEPA/insert`	`POST`	reçoit un flux `XML` décrivant une transaction, crée l'objet correspondant et retourne la valeur `PmtId`.

Quelques références utiles :

- http://docs.spring.io/spring/docs/4.0.6.RELEASE/spring-framework-reference/htmlsingle/
- https://spring.io/guides/gs/rest-service/
- http://howtodoinjava.com/spring/spring-restful/spring-rest-hello-world-xml-example/

Remarques :
- La mise en œuvre peut varier selon les versions utilisées, et est valide pour la configuration suivante :

```
Java 8 - maven (2.2 ou) 3 - tomcat (7 ou) 8
```

- Privilégiez la compilation de votre projet avec `maven`

- Si l'intégration de `Tomcat` avec `Eclipse` ne se fait pas correctement, effectuez le déploiement manuellement.
- En cas de problème persistant, il est également possible d'installer un autre serveur tel que "`Glassfish`", et l'utiliser directement via son interface d'administration (port `4848`). Veuillez toutefois à ne pas générer de conflit sur le port `8080`.

- Pour le test de l'application en mode REST, utilisez un client Rest approprié.
Exemple d'outils :

postman, resteasy, curl, netcat
greffons firefox : HttpRequester, RestClient, …
...

II - Création du projet

Créez un nouveau projet dans `Eclipse` en respectant les consignes suivantes :
- Projet `maven` avec archetype `quickstart`
- Votre application devra générer un fichier war, qui sera déployé sur un serveur `Tomcat`
- Créez l'arborescence nécessaire :

```
src → main → java → {groupID} → {artifactid} → config
                                                controller
                                                model
```

- Configurez votre projet (pom.xml) pour utiliser les dépendances suivantes :

dépendance	version	scope
`java`	`1.8`	
`spring-context`	`4.1.6.RELEASE`	
`spring-webmvc`		
`spring-orm`		
`cglib-nodep`	`2.2.2`	`runtime`
`javax.servlet-api`	`3.1.0`	`provided`

```
build / plugin
- maven-war-plugin          2.6
     (configuration : failOnMissingWebXml = false)

packaging : war
```

III - Affichage de la page de base

Vous devez maintenant créer le projet de base afin d'afficher une page d'accueil, présentant l'utilisation de l'application, et résumée dans le tableau présenté en début du sujet.

Question III.1 - Création de l'application de base

Pour cela, créez les classes suivantes : `controller →  HomeController`
`config  →  AppConfig`
`ServletInitializer`

HomeController

Ce contrôleur affiche un message d'accueil lors de la connexion à l'application via le serveur tomcat. Il répond à une requête utilisant la méthode `GET`.

AppConfig

Code de la classe :

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = {"urouen.SEPA.controller"})
public class AppConfig {
}
```

ServletInitializer

```
public class ServletInitializer extends
          AbstractAnnotationConfigDispatcherServletInitializer {
  @Override
  protected Class<?>[] getRootConfigClasses() {
    return new Class<?>[0];
  }
```

```
  @Override
  protected Class<?>[] getServletConfigClasses() {
    return new Class<?>[]{AppConfig.class};
  }

  @Override
  protected String[] getServletMappings() {
    return new String[]{"/"};
  }
}
```

Validez votre application et testez son fonctionnement.
Vous devez affichez le mode d'emploi lors de la connexion à la racine de l'application.

Question III.2 - Export d'un objet simple

Vous allez maintenant développer le contrôleur renvoyant une transaction d'après son identifiant (second cas). Pour l'instant, nous ne tiendrons pas compte de la valeur de l'ID et renverrons le flux XML du même objet à chaque appel.

Question III.3 - Classe "SEPA"

Créez dans le paquetage modèle de votre projet, classe nommée "SEPA" qui contient, pour l'instant, une version unique de transaction telle que celle décrite par l'extrait XML ci-dessous. Vous annoterez la classe pour qu'elle soit sérialisée en XML selon vos souhaits.

Exemple de flux XML correspondant à une transaction de base :

```
<DrctDbtTxInf>
   <PmtId>REF OPE BBBB</PmtId>
   <InstdAmt Ccy="EUR">2150.08</InstdAmt>
   <DrctDbtTx>
     <MndtRltdInf>
       <MndtId>MANDAT NO 666666</MndtId>
       <DtOfSgntr>1989-07-03</DtOfSgntr>
     </MndtRltdInf>
   </DrctDbtTx>
   <DbtrAgt>
     <FinInstnId>
       <BIC>BANKGBUL</BIC>
     </FinInstnId>
   </DbtrAgt>
   <Dbtr>
     <Nm>Mr Debiteur N2</Nm>
   </Dbtr>
   <DbtrAcct>
     <Id>
       <IBAN>GB29NWBK60161331926819</IBAN>
     </Id>
   </DbtrAcct>
   <RmtInf>Facture reference ISO 654321</RmtInf>
 </DrctDbtTxInf>
```

Question III.4 - Contrôleur "SEPAController"

Créez dans le paquetage contrôleur, une classe "`SepaController`" comportant une méthode répondant aux requêtes du type `resume/id`.

Ce contrôleur renvoie une transaction créée automatiquement lors de l'appel (création d'un objet unique), quelque soit le paramètre fourni. (utilisez la transaction précédente comme modèle)

Validez le fonctionnement en effectuant une requête REST.

Question III.5 - Gestion de l'identifiant.

Modifiez votre contrôleur afin de prendre en charge l'identifiant fourni (`Id = PmtId`)
Générez la date en utilisant la date courante (aucune contrainte de mise en forme)

Validez le fonctionnement, en vous assurant que les valeurs `title` et `date` sont bien actualisées.

Question III.6 - Retour des listes de transactions

Modifiez votre application afin de pouvoir retourner des listes de transactions.
Ajoutez une entrée à votre contrôleur pour l'url "`resume/`"

A la réception de cette URL, votre contrôleur retournera la liste des transactions enregistrées, en affichant les valeurs `PmtId` et `InstdAmt`.

(créez arbitrairement 3 ou 4 STB selon le schéma utilisé précédemment)

IV - Réception d'une STB sous forme de flux XML

Question IV.1 - Réception d'une transaction

Ajoutez un nouveau contrôleur afin de pouvoir recevoir une transaction au format `XML,` transmise avec la méthode "`HTTP POST`" et le lire sous forme d'objet. En retour, l'application retournera un message avec le contenu transmis.