

GUIDE POUR LA RÉALISATION DU MINI PROJET “PFE” EN SPRING MVC/SPRING IOC/ JPA

- ❖ La validation du module “Frameworks” est totalement liée à la réalisation de ce mini-projet.
- ❖ Le travail peut être fait par binôme.
- ❖ Les étapes de ce Mini Projet

MINI PROJET PFE “GESTION BANKS” 45 PTS	
CREATION DU PROJET MAVEN	5 PTS
CREATE,READ,UPDATE,DELETE D'UNE CLASSE MODELE	
DAO LAYER (JPA)	10 PTS
SERVICE LAYER	5 PTS
PRESENTATION LAYER (SPRING MVC)	10 PTS
INTEGRATION TEMPLATE	5 PTS
INTERCEPTEURS SPRING MVC	5 PTS
VALIDATION DES INPUTS	5 PTS

- ❖ L'objectif Final de ce Mini Projet est de réaliser les opérations CRUD d'une classe Modèle : Compte bancaire dans notre cas d'illustration.

CRUD Operation

[+ New](#) [PDF](#)

Show: 10 entries Search:

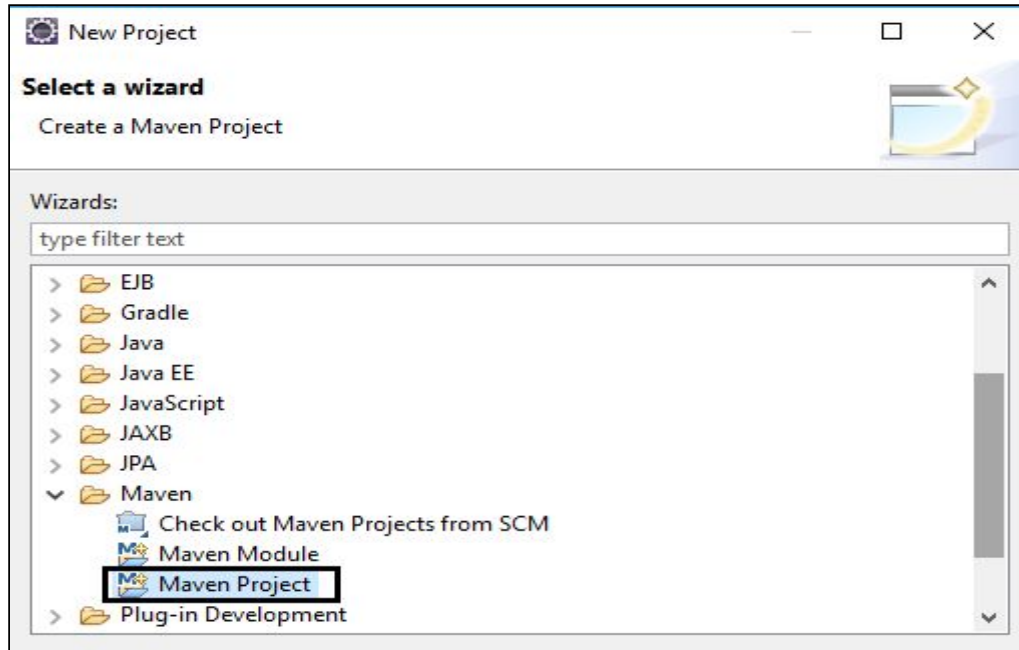
ID	Firstname	Lastname	Address	Action
1	neovic	devierte	slay city	Edit Delete
2	gemalym	cépe	caimen, bohöl	Edit Delete
3	lee	apilinga	bacolod	Edit Delete
4	julyñ	divnagracia	eb magalona	Edit Delete
5	cristine	demapanag	talisay	Edit Delete
38	John	doe	epss	Edit Delete

Showing 1 to 6 of 6 entries

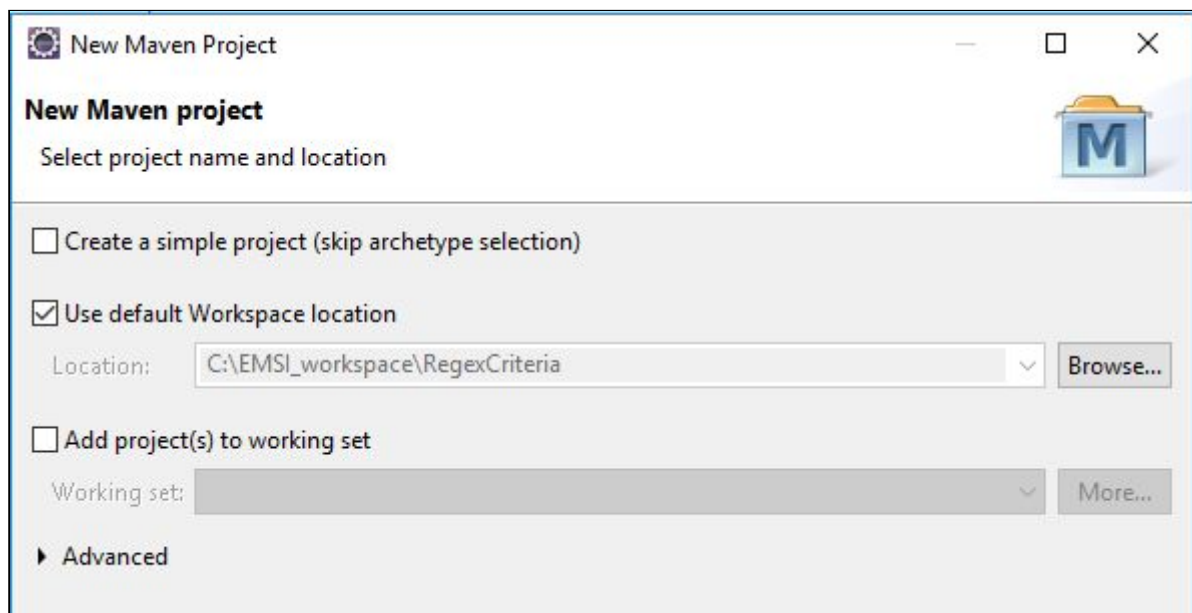
Previous **1** Next

CREATION DU PROJET MAVEN

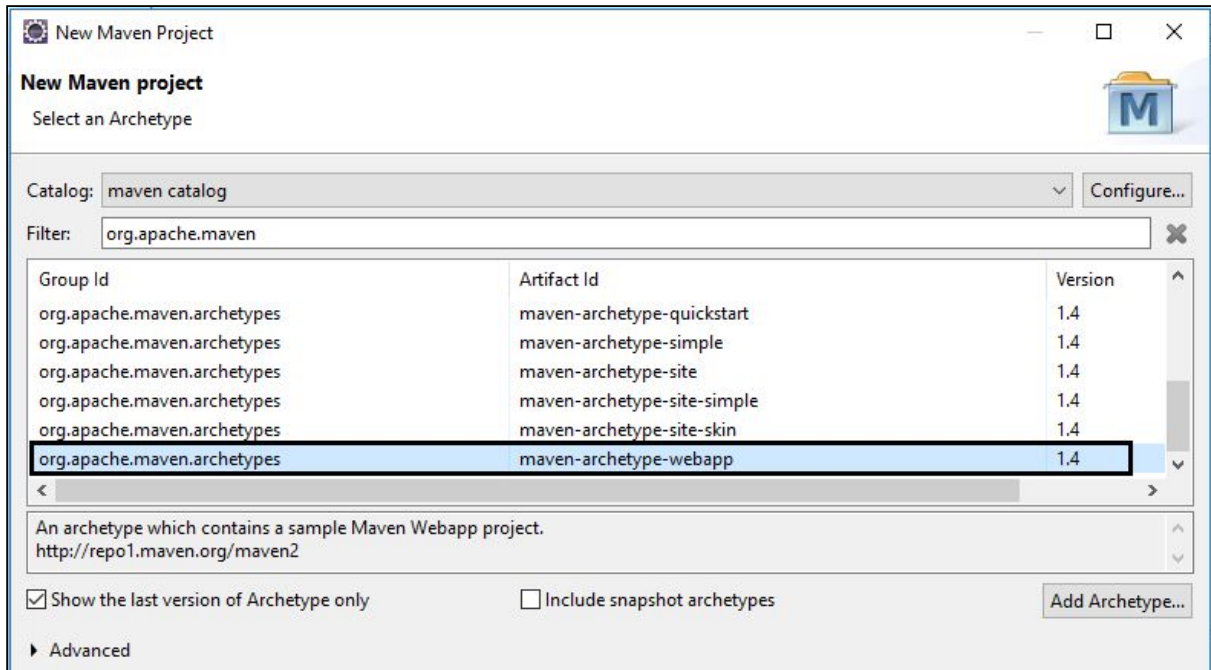
I. File>NEW>Project>Maven Project



II. >>Next



III. Choisir le Archetype ID "maven-archetype-webapp". Voir ci-dessous



New Maven Project

New Maven project

Select an Archetype

Catalog: maven catalog Configure...

Filter: org.apache.maven X

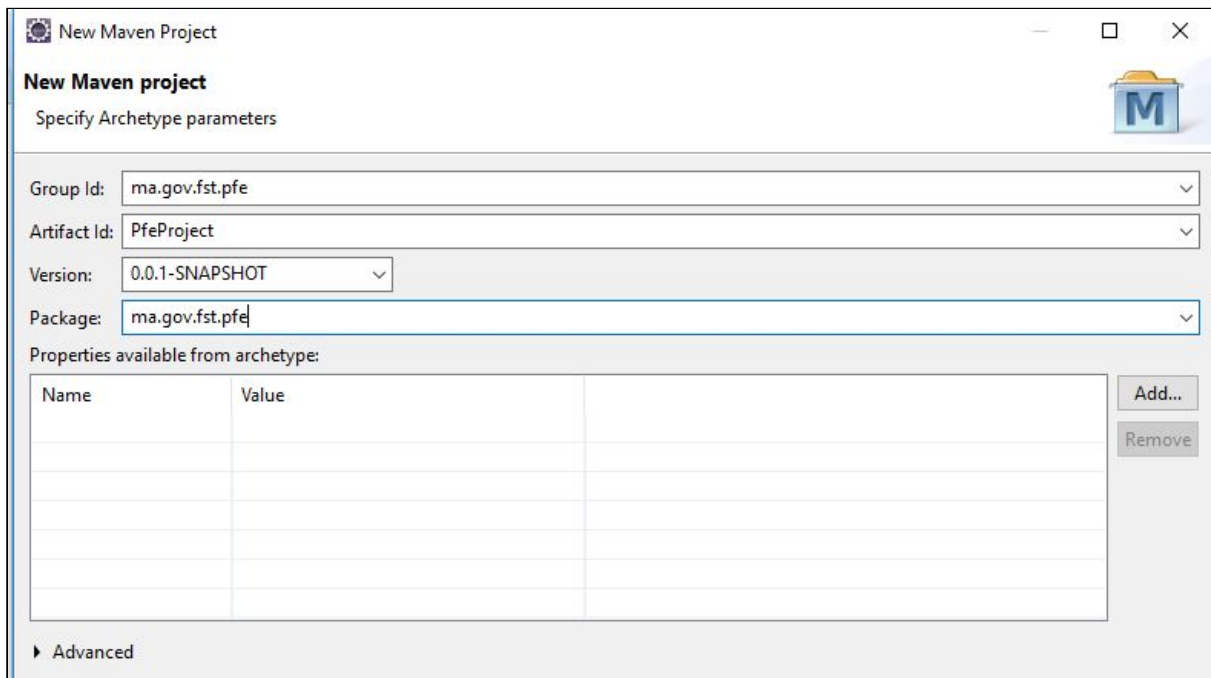
Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-quickstart	1.4
org.apache.maven.archetypes	maven-archetype-simple	1.4
org.apache.maven.archetypes	maven-archetype-site	1.4
org.apache.maven.archetypes	maven-archetype-site-simple	1.4
org.apache.maven.archetypes	maven-archetype-site-skin	1.4
org.apache.maven.archetypes	maven-archetype-webapp	1.4

An archetype which contains a sample Maven Webapp project.
<http://repo1.maven.org/maven2>

☒ Show the last version of Archetype only ☐ Include snapshot archetypes Add Archetype...

► Advanced

IV. Spécifier le Group Id et l'Artifact Id. Voir ci-dessus



New Maven Project

New Maven project

Specify Archetype parameters

Group Id: ma.gov.fst.pfe

Artifact Id: PfeProject

Version: 0.0.1-SNAPSHOT

Package: ma.gov.fst.pfe

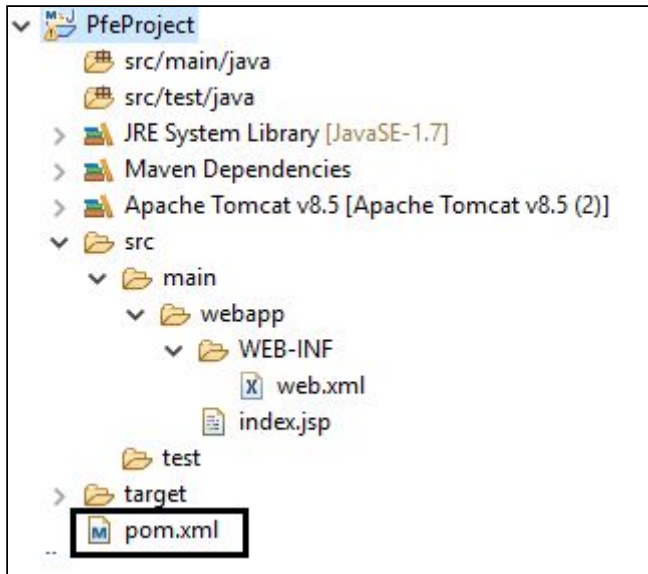
Properties available from archetype:

Name	Value

Add...
Remove

► Advanced

V. >>Finish donnera l'arborescence du projet suivante:



VI. Mettre à jour les dépendances de votre projet dans le “pom.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>ma</groupId>
  <artifactId>PfeProjet</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>Gbanks4 Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
    <spring.version>4.0.1.RELEASE</spring.version>
    <jstl.version>1.2</jstl.version>
    <servletapi.version>3.1.0</servletapi.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <!-- jstl -->
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>${jstl.version}</version>
    </dependency>

    <!-- Spring dependencies -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>
  </dependencies>
</project>
```

```

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${spring.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.2.10.Final</version>
</dependency>

<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>5.2.10.Final</version>
</dependency>
<!-- Spring Security -->
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-web</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-taglibs</artifactId>
  <version>${spring.version}</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.6</version>
</dependency>

<!-- Logging -->
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.7.5</version>
</dependency>

</dependencies>

<build>
  <finalName>Gbanks4</finalName>
  <pluginManagement><!-- lock down plugins versions to avoid using Maven
    defaults (may be moved to parent pom) -->
    <plugins>
      <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>3.1.0</version>
      </plugin>
      <!-- see
http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_war_packaging -->
      <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <version>3.0.2</version>
      </plugin>
    </plugins>
  </pluginManagement>
</build>

```

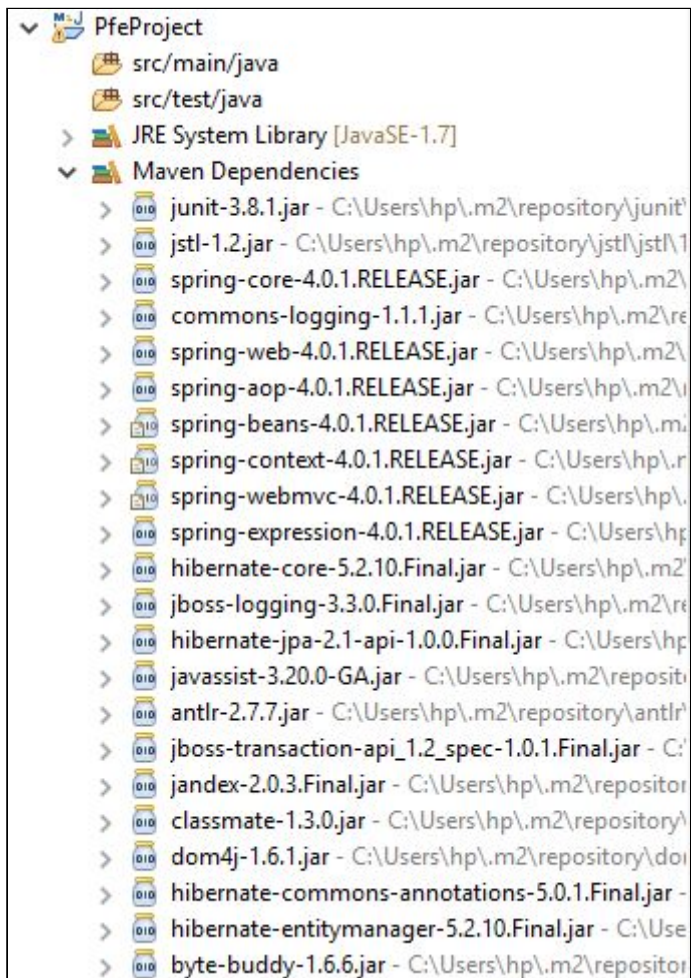


```

        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.0</version>
        </plugin>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.1</version>
        </plugin>
        <plugin>
            <artifactId>maven-war-plugin</artifactId>
            <version>3.2.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-install-plugin</artifactId>
            <version>2.5.2</version>
        </plugin>
        <plugin>
            <artifactId>maven-deploy-plugin</artifactId>
            <version>2.8.2</version>
        </plugin>
    </plugins>
</pluginManagement>
</build>
</project>

```

VII. Vérifier la récupération des JARs par maven.



CREATE, READ, UPDATE, DELETE D'UNE CLASSE MODELE

Nous considérons dans cette partie que PFE="GESTION BANKS" et que la classe à traiter est la classe "Compte" définie par l'Id, le numéro, la description et le capital.

VIII. Créer la classe modèle "Compte" dans le package "ma.gov.gbanks.models"

```
package ma.gov.gbank.models;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name = "TCOMPTE")
public class Compte {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String num;
    private String description;
    private String capital;
    public Compte(Long id, String num, String description) {
        super();
        this.id = id;
        this.num = num;
        this.description = description;
    }
    public Compte(String num, String description, String capital) {
        super();
        this.num = num;
        this.description = description;
        this.capital = capital;
    }
    public String getCapital() {
        return capital;
    }
    public void setCapital(String capital) {
        this.capital = capital;
    }
    public Compte(String num) {
        super();
        this.num = num;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public Compte() {
        super();
    }
    @Override
    public String toString() {
        return "Compte [num=" + num + ", description=" + description + "]";
    }
}
```

```

    }
    public Compte(String num, String description) {
        super();
        this.num = num;
        this.description = description;
    }
    public String getNum() {
        return num;
    }
    public void setNum(String num) {
        this.num = num;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}

```

IX. Créer l'interface Dao et son implémentation dans "ma.gov.gbanks.dao"

L'interface

```

package ma.gov.gbank.dao;
import java.util.List;
import ma.gov.gbank.models.Compte;
public interface IDao {
    boolean insert(Compte c) ;
    boolean update(Compte c);
    boolean delete(Compte c);
    List<Compte> selectAll() ;
    Compte selectById(Long id);
}

```

L'implémentation

```

package ma.gov.gbank.dao;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;
import org.springframework.stereotype.Repository;
import ma.gov.gbank.models.Compte;
@Repository
public class DaoImplJPA implements IDao{
    EntityManagerFactory emf= Persistence.createEntityManagerFactory("BanqueU");
    EntityManager em = emf.createEntityManager();
    @Override
    public boolean insert(Compte c) {
        EntityTransaction tx= null;
        try {
            tx = em.getTransaction();

```



```

        tx.begin();
        em.persist(c);
        tx.commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        tx.rollback();
        return false;
    }
}

@Override
public boolean update(Compte c) {
    EntityTransaction tx= null;
    try {
        tx=em.getTransaction();
        tx.begin();
        Compte c1 = (Compte)em.find(Compte.class, c.getId());
        c1.setDescription(c.getDescription());
        c1.setNum(c.getNum());
        tx.commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        tx.rollback();
        return false;
    }
}

@Override
public boolean delete(Compte c) {
    EntityTransaction tx= null;
    try {
        tx=em.getTransaction();
        tx.begin();
        Compte c1 = (Compte)em.find(Compte.class, c.getId());
        em.remove(c1);
        tx.commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        tx.rollback();
        return false;
    }
}

@Override
public List<Compte> selectAll() {
    Query q=em.createQuery("from Compte c");
    return q.getResultList();
}

@Override
public Compte selectById(Long id) {
    Compte c = (Compte) em.find(Compte.class, id);
    return c;
}
}

```

X. Ajouter le fichier “persistence.xml” dans “src/META-INF”

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">
  <persistence-unit name="BanqueU">
    <!-- <class>banque.models.Banque</class> -->
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/BanqueDB" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password" value="root" />
      <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="false" />
      <property name="hibernate.hbm2ddl.auto" value="update" />
    </properties>
  </persistence-unit>
</persistence>
```

XI. Tester vos méthodes en utilisant la méthode main.

XII. Créer l'interface Service et son implémentation dans "ma.gov.gbanks.service"

L'interface

```
package ma.gov.gbank.service;
import java.util.List;
import ma.gov.gbank.models.Compte;

public interface CompteService {
    boolean ajouterCompte(Compte c);
    List<Compte> selectAll();
}
```

L'implémentation

```
package ma.gov.gbank.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ma.gov.gbank.dao.IDao;
import ma.gov.gbank.models.Compte;

@Service
public class CompteServiceImpl implements CompteService {

    @Autowired
    IDao dao;

    @Override
    public boolean ajouterCompte(Compte c) {
        return dao.insert(c);
    }

    @Override
    public List<Compte> selectAll() {
        return dao.selectAll();
    }
}
```

11/18

```
<servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/springmvc-servlet.xml
  </param-value>
</context-param>
<listener>
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
</web-app>
```

XV. Créer le fichier de configuration du contrôleur principal:springmvc-servlet.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
">
  <context:component-scan base-package="ma.gov.gbanks" />
  <mvc:annotation-driven />
  <mvc:resources mapping="/resources/**" location="/resources/" />
  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix">
      <value>/WEB-INF/views/</value>
    </property>
    <property name="suffix">
      <value>.jsp</value>
    </property>
  </bean>
</beans>
```

XVI. Créer la page JSP contenant le formulaire d'ajout et l'affichage de la liste.

WEB-INF/views/comptes.JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="f"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
```

```

</head>
<body>
    <f:form action="${pageContext.request.contextPath}/compte/crud"
        modelAttribute="compte">
        Numero : <f:input path="num" />
        Description : <f:input path="description" />
        <input type="submit" value="Add" name="Add" />
        <input type="submit" value="Update" name="Update" />
        <input type="submit" value="Search" name="Search" />

    </f:form>
    <br>
    <h3>Persons List</h3>
    <c:if test="${!empty listComptes}">
        <table class="tg">
            <tr>
                <th width="80">Compte Num</th>
                <th width="120">Compte Desc</th>
                <th width="60">Edit</th>
                <th width="60">Delete</th>
            </tr>
            <c:forEach items="${listComptes}" var="cmp">
                <tr>
                    <td>${cmp.num}</td>
                    <td>${cmp.description}</td>
                    <td><a href="<c:url value='/edit/${cmp.id}'
/>">Edit</a></td>
                    <td><a href="<c:url value='/remove/${cmp.id}'
/>">Delete</a></td>
                </tr>
            </c:forEach>
        </table>
    </c:if>
</body>
</html>

```

XVII. Déployer sur Tomcat

- XXIV. Ajouter les dépendances SLF4J.JAR ET LOG4J.JAR dans le “pom.xml”.**
- XXV. Créer une classe “LoggerInterceptor” fille de HandlerInterceptorAdapter**
- XXVI. Redéfinir les trois méthodes preHandle, postHandle et afterCompletion**

```
package ma.gov.gbank.presentation.controllers;
import java.time.Instant;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
public class LoggerInterceptor extends HandlerInterceptorAdapter {
    private static final Logger logger = LoggerFactory.getLogger(LoggerInterceptor.class);

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
Object handler) throws Exception {
        long startTime = Instant.now().toEpochMilli();
        logger.info("Request URL::" + request.getRequestURL().toString() + ":: Start
Time=" + Instant.now());
        request.setAttribute("startTime", startTime);
        return true;
    }
    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response,
Object handler, Exception ex) {
        long startTime = (Long) request.getAttribute("startTime");
        logger.info("Request URL::" + request.getRequestURL().toString() +
":: Time Taken=" + (Instant.now().toEpochMilli() - startTime));
    }
}
```

- XXVII. Configurer spring-servlet pour créer un bean de votre intercepteur.**

```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/*" />
        <mvc:exclude-mapping path="/resources/*" />
        <bean
class="ma.gov.gbank.presentation.controllers.LoggerInterceptor"></bean>
    </mvc:interceptor>
</mvc:interceptors>
```

- XXVIII. Vérifier que les actions “addCompte” et “listCompte” sont associées à cette intercepteur.**

VALIDATION DES FORMULAIRE EN SPRING MVC

- 24. Ajouter la Dépendance des JAR de validation [validation-api et hibernate-validator] dans le fichier pom.xml**

```
<!-- Form Validation using Annotations -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>5.2.4.Final</version>
</dependency>
```

25. Utiliser les annotations de validation nécessaires dans les classes Modèles

```
@NotNull
@Size(max = 5, min = 2)
private String num;

>Email
private String description;

@NotNull
@Max(100)
@Min(10)
private String capital;
```

26. Changer la signature des méthodes du contrôleur en ajoutant l'annotation @Valid et le paramètre BindingResult

27. Ajouter des traces dans les méthodes du contrôleur pour vérifier s'il y a des erreurs de validation

```
@RequestMapping(value="/crud", params="Add")
public ModelAndView addCompte(@ModelAttribute("compte") @Valid Compte c,
BindingResult br) {
    System.out.println("Call Contoller Add.....");
    if(br.hasErrors()) {
        List l = br.getAllErrors();
        System.out.println("errror :" + l);
    }
    service.ajouterCompte(c);
    ModelAndView mv = new ModelAndView("redirect:/compte/");
    mv.addObject("compte", c);
    return mv;
}
```

28. Remarquer les erreurs de validation dans la console

```
errror :[Field error in object 'compte' on field 'num': rejected value []; codes
[Size.compte.num,Size.num,Size.java.lang.String,Size]; arguments
[org.springframework.context.support.DefaultMessageSourceResolvable: codes
[compte.num,num]; arguments []; default message [num],5,2]; default message [Doit etre
entre ], Field error in object 'compte' on field 'capital': rejected value [null]; codes
[NotNull.compte.capital,NotNull.capital,NotNull.java.lang.String,NotNull]; arguments
[org.springframework.context.support.DefaultMessageSourceResolvable: codes
[compte.capital,capital]; arguments []; default message [capital]]; default message [ne peut
```

pas être nul]]

29. **Afficher les erreurs de validation dans la page JSP en utilisant le tag :**
<f:errors path="" /> ou <f:errors path="num" />
30. **S'il s'agit d'une redirection de type "redirect" il repasser le modèle et les erreurs à l'actions destination. Dans notre cas nous devons changer la méthode addCompte et la méthode selectAllComptes comme suivant:**

La méthode addCompte:

```
@RequestMapping(value = "/crud", params = "Add")
public ModelAndView addCompte(@ModelAttribute("compte") @Valid Compte c,
BindingResult br, RedirectAttributes ra) {
    System.out.println("Call Contoller Add.....");
    ra.addFlashAttribute("compte", c);
    if (br.hasErrors()) {
        List l = br.getAllErrors();
        System.out.println("errror :" + l);

ra.addFlashAttribute("org.springframework.validation.BindingResult.compte", c);
    }
    service.ajouterCompte(c);
    ModelAndView mv = new ModelAndView("redirect:/compte/");
    //mv.addObject("compte", c);
    return mv;
}
```

La méthode selectAllComptes:

```
@RequestMapping("/")
public ModelAndView selectAllComptes( @ModelAttribute("compte") @Valid
Compte c, BindingResult br) {
    System.out.println("Call Contoller Method.....");
    if (c == null)
        c = new Compte();
    List<Compte> list = service.selectAll();
    System.out.println(list);
    ModelAndView mv = new ModelAndView("comptes");
    mv.addObject("compte", c);
    mv.addObject("listComptes", list);
    return mv;
}
```

31. **Utiliser l'affichage des erreurs dans la page JSP contenant le formulaire.**

Numéro : Doit être entre Capital : doit être au maximum égal à 100
doit être au minimum égal à 10 Description : Adresse email mal formée

EXERCICE D'APPLICATION

Créer une action d'authentification des utilisateurs de l'application.

- 1- Il faut créer la classe User contenant le mail et le mot de passe.
- 2- Utiliser les annotations JPA pour associer cette classe avec la table équivalent dans la base de donnée.
- 3- Créer les méthodes DAO
- 4- Créer les méthode Service
- 5- Créer le contrôleur UserController
- 6- Créer la page JSP d'authentification

Règles de validation:

Le mail doit être un mail.

Le mot de passe doit contenir 8 caractère avec un digit au minimum.

- 7- Si l'utilisateur s'est connecté correctement, il faut créer une variable dans la session.

Créer un intercepteur qui vérifie si l'utilisateur est connecté avant d'exécuter des actions du projets.