

# Atelier 01 Spring Boot :

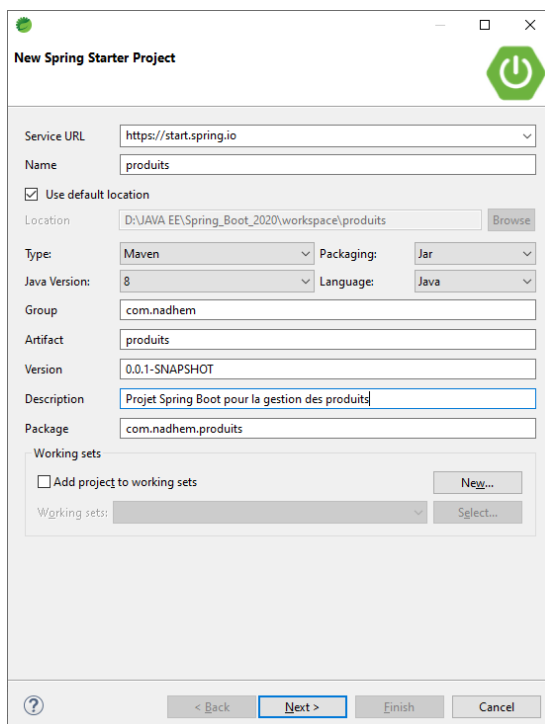
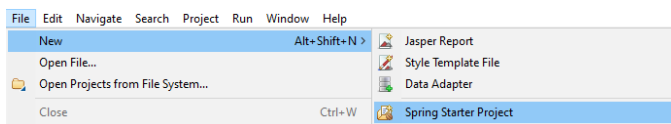
## Les couches Modèle, Persistance et Service d'une application Web MVC CRUD

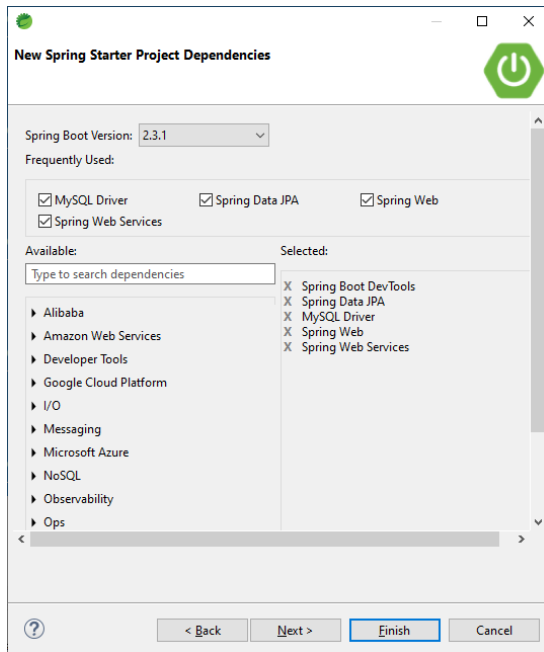
### Objectifs :

1. Création d'un premier projet Spring boot,
2. Création de l'entité *Produit* et de son interface Repository,
3. Configuration du fichier ***application.properties***,
4. Tester les opérations **CRUD** sur l'entité Produit,
5. Création de la couche **Service**,

### Création d'un premier projet Spring boot

1. Créer un projet « Spring Starter Project » (si vous ne le trouvez pas cherchez dans Others)





## Création de l'entité Produit et de son interface Repository

### 2. Créer l'entité Produit dans le package com.nadhem.produits.entities

```
package com.nadhem.produits.entities;

import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Produit {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long idProduit;
    private String nomProduit;
    private Double prixProduit;
    private Date dateCreation;

    public Produit() {
        super();
    }

    public Produit(String nomProduit, Double prixProduit, Date dateCreation) {
        super();
        this.nomProduit = nomProduit;
        this.prixProduit = prixProduit;
        this.dateCreation = dateCreation;
    }
}
```

```

    public Long getIdProduit() {
        return idProduit;
    }

    public void setIdProduit(Long idProduit) {
        this.idProduit = idProduit;
    }

    public String getNomProduit() {
        return nomProduit;
    }

    public void setNomProduit(String nomProduit) {
        this.nomProduit = nomProduit;
    }

    public Double getPrixProduit() {
        return prixProduit;
    }

    public void setPrixProduit(Double prixProduit) {
        this.prixProduit = prixProduit;
    }

    public Date getDateCreation() {
        return dateCreation;
    }

    public void setDateCreation(Date dateCreation) {
        this.dateCreation = dateCreation;
    }

    @Override
    public String toString() {
        return "Produit [idProduit=" + idProduit + ", nomProduit=" +
nomProduit + ", prixProduit=" + prixProduit
        + ", dateCreation=" + dateCreation + "]";
    }
}

```

### 3. Créer l'interface ProduitRepository dans le package com.nadhem.produits.repos

```

package com.nadhem.produits.repos;

import org.springframework.data.repository.JpaRepository;
import com.nadhem.produits.entities.Produit;

public interface ProduitRepository extends JpaRepository<Produit, Long> {
}

```

## Configuration du fichier application.properties

4. Modifier le fichier src/main/resources/application.properties comme suit :

```
spring.datasource.url=jdbc:mysql://localhost:3306/spring_DB?createDatabaseIfNotExist=true&useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

## Tester les opérations CRUD sur l'entité Produit

5. Exécuter l'application Run as / Spring boot Application, vérifier la création de la table produit dans la BD spring\_DB
6. Ecrire la méthode pour tester l'ajout d'un produit à la base de données, modifier la classe src/main/test/ com.nadhemb.products.ProduitsApplicationTests comme suit :

```
package com.nadhemb.products;

import java.util.Date;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.nadhemb.products.entities.Produit;
import com.nadhemb.products.repos.ProduitRepository;

@SpringBootTest
class ProduitsApplicationTests {
    @Autowired
    private ProduitRepository produitRepository;

    @Test
    public void testCreateProduit() {
        Produit prod = new Produit("PC Dell", 2200.500, new Date());
        produitRepository.save(prod);
    }
}
```

7. Sélectionnez la méthode `testCreateProduit` puis clic droit/Run as /JUnit test, puis vérifier l'insertion d'un produit dans la table produit,
8. De la même manière tester les autres opérations :

```
@Test
public void testFindProduit()
{
    Produit p = produitRepository.findById(1L).get();
    System.out.println(p);
}

@Test
public void testUpdateProduit()
{
    Produit p = produitRepository.findById(1L).get();
    p.setPrixProduit(1000.0);
    produitRepository.save(p);
}

@Test
```

```

    public void testDeleteProduit()
    {
        produitRepository.deleteById(1L);
    }

@Test
    public void testListerTousProduits()
    {
        List<Produit> prods = produitRepository.findAll();
        for (Produit p : prods)
        {
            System.out.println(p);
        }
    }

```

## Création de la couche Service

Créer, dans le package `com.nadhem.produits.service` l'interface `ProduitService`

```

package com.nadhem.produits.service;

import java.util.List;

import com.nadhem.produits.entities.Produit;

public interface ProduitService {

    Produit saveProduit(Produit p);
    Produit updateProduit(Produit p);
    void deleteProduit(Produit p);
    void deleteProduitById(Long id);
    Produit getProduit(Long id);
    List<Produit> getAllProduits();
}

```

9. Créer l'implémentation de cette interface : la classe `ProduitServiceImpl` :

```

package com.nadhem.produits.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import com.nadhem.produits.entities.Produit;
import com.nadhem.produits.repos.ProduitRepository;

@Service
public class ProduitServiceImpl implements ProduitService {

    @Autowired
    ProduitRepository produitRepository;

    @Override
    public Produit saveProduit(Produit p) {
        return produitRepository.save(p);
    }
}

```

```

    }

    @Override
    public Produit updateProduit(Produit p) {
        return produitRepository.save(p);
    }

    @Override
    public void deleteProduit(Produit p) {
        produitRepository.delete(p);
    }

    @Override
    public void deleteProduitById(Long id) {
        produitRepository.deleteById(id);
    }

    @Override
    public Produit getProduit(Long id) {
        return produitRepository.findById(id).get();
    }

    @Override
    public List<Produit> getAllProduits() {
        return produitRepository.findAll();
    }
}

```