

1) Qu'est-ce qu'une API REST ?

Imagine que tu as **un petit magasin**.

Les gens viennent et te demandent :

- “Donne-moi la liste des produits.” → **GET**
- “Ajoute ce nouveau produit.” → **POST**
- “Change ce produit.” → **PUT**
- “Supprime ce produit.” → **DELETE**

Une API REST fait exactement la même chose mais avec des données.
Elle reçoit des demandes et renvoie des réponses.

Spring Boot, c'est **le serveur** qui gère ce magasin.

2) Que fait ce TP exactement ?

Dans ce TP, nous avons construit une **petite application** qui gère des “Articles”.

On a créé :

- un **contrôleur** (controller) : c'est la **porte d'entrée**, là où les clients frappent.
- un **service** : c'est **la personne dans la maison** qui comprend la demande et fait le travail logique.
- un **DAO** : c'est **le placard** où sont rangés les articles.
- un **DTO** : c'est **la boîte** avec laquelle tu donnes les articles aux clients, proprement.
- un **ExceptionHandler** : c'est **quelqu'un qui s'excuse poliment** quand il y a une erreur.

Grâce à ça, l'application fonctionne comme un vrai mini système.

3) Le workflow — Comment une requête circule ?

Disons qu'un client demande :

“Je veux tous les articles !”

Voici le chemin :

1. Il frappe à la **porte** → le **controller**.
2. La porte appelle la personne dans la **pièce principale** → le **service**.
3. Le service va chercher les articles **dans le placard** → le **DAO**.
4. Une fois trouvés, le service les remet dans une **boîte propre (DTO)**.
5. Le controller donne cette boîte au client au format JSON.

Tout est organisé comme une chaîne bien réglée.

4) Architecture en couches (expliqué très simplement)

Pourquoi on sépare l'application en plusieurs couches ?

Parce que c'est **beaucoup plus propre**.
C'est comme une maison bien rangée :

- **La porte** : on accueille les gens → Controller
- **Le salon** : on discute et on réfléchit → Service
- **Le placard** : on range les objets → DAO
- **La boîte de présentation** : on sert le résultat → DTO

On ne mélange pas :

- les données avec la logique,
- ni la logique avec la présentation.

C'est plus clair, plus propre, plus facile à réparer.

5) Injection de dépendances (expliqué comme à un enfant)

Normalement, si tu veux demander à quelqu'un de t'aider, tu dois l'appeler toi-même.

Mais avec Spring :

C'est Spring qui t'amène directement la bonne personne.

Exemple :

Tu écris juste :

```
@Autowired  
private IService service;
```

Et hop !

Spring t'amène automatiquement un objet `ServiceImpl`.

C'est comme si tu disais :

“J'ai besoin du service.”

Spring répond : “Ne t'inquiète pas, je le fais venir.”

6) Différence entre BO et DTO (explication super simple)

BO = l'objet interne, la vraie donnée.

DTO = la version propre qu'on montre au client.

C'est comme la cuisine :

- Tu cuisines ta soupe dans une grosse marmite → **BO**
- Tu ne vas pas servir la marmite au client !
- Tu verses la soupe dans un joli bol → **DTO**

Le DTO sert à protéger l'intérieur du système et à contrôler ce qu'on donne au client.

7) Validation — Pourquoi c'est important ?

Tu ne veux pas qu'un client envoie un article :

- **sans description,**
- **ou avec un prix = 0,**
- **ou une quantité négative.**

La validation, c'est comme un **gardien à l'entrée** :

“Désolé, votre produit n'est pas valable.”

Si les données sont mauvaises, Spring renvoie une erreur 400 très propre.

8) Exception Handler — explication simple

Quand quelque chose ne va pas, au lieu d'envoyer :

```
NullPointerException at line 42...
```

ce qui serait horrible pour un client, on envoie :

```
{
  "message": "Erreur fonctionnelle",
  "details": ["Article non trouvé"]
}
```

C'est comme si tu disais poliment :

“Désolé, ça n'existe pas.”

9) Les tests — pourquoi on les fait ?

Les tests servent à vérifier que :

- la porte s'ouvre bien,
- la logique fonctionne,
- les données sont bien manipulées,
- les erreurs sont bien gérées.

On teste :

✓ Tests unitaires

On vérifie des parties individuelles (comme si on testait un bouton).

✓ Tests d'intégration

On teste tout ensemble (comme si on testait toute la machine).

✓ Postman

On joue le rôle du client réel :
on envoie des requêtes et on regarde les réponses.

10) Résumé

Voici le TP résumé en **4 phrases super simples** :

- On a construit une maison (l'application).
- Les gens frappent à la porte (controller).
- La personne dans le salon gère la demande (service).
- Le placard garde les articles (DAO).

Et tout est servi proprement (DTO), avec politesse (Exception Handler),
et on vérifie que tout marche (tests).