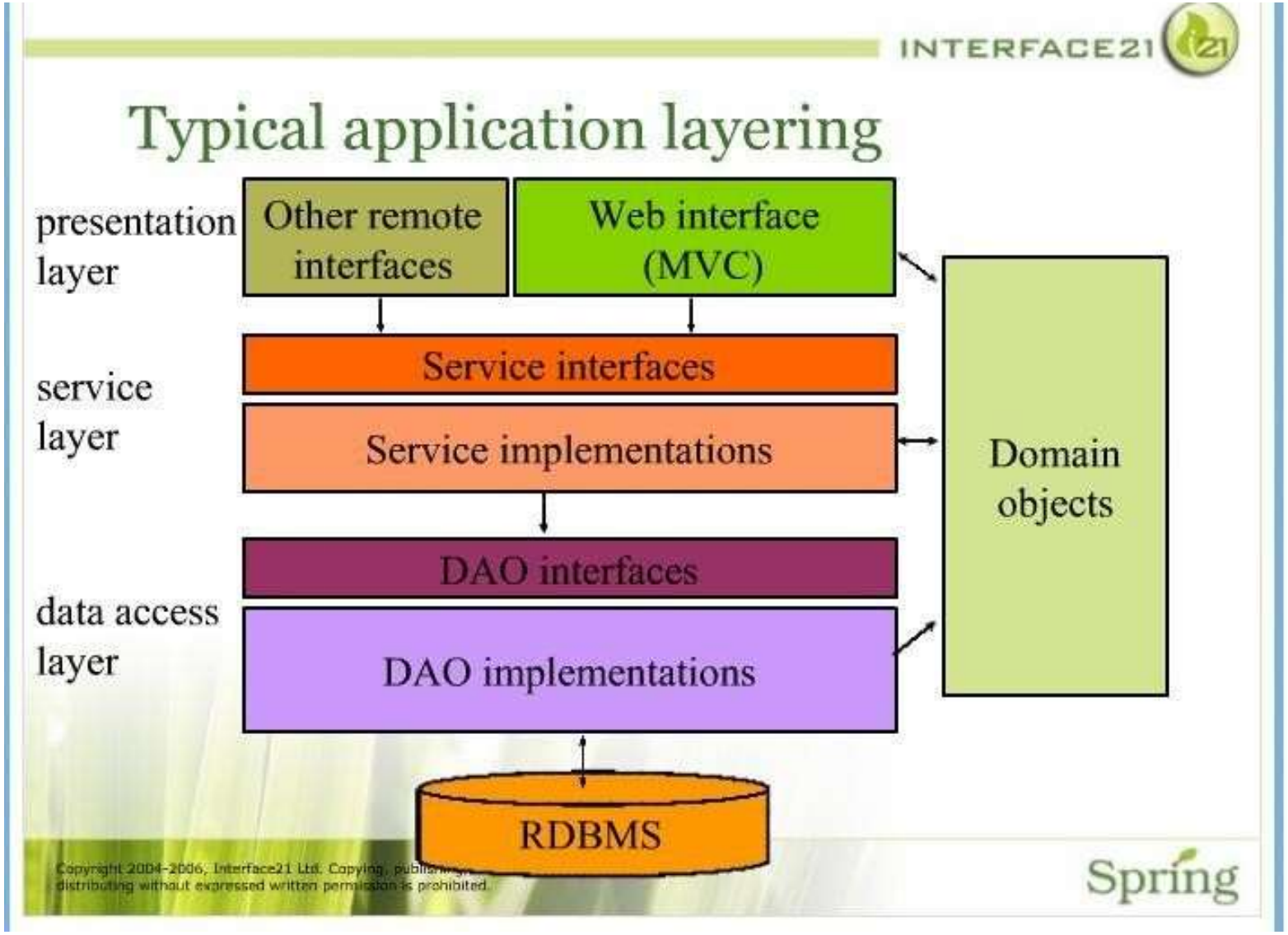
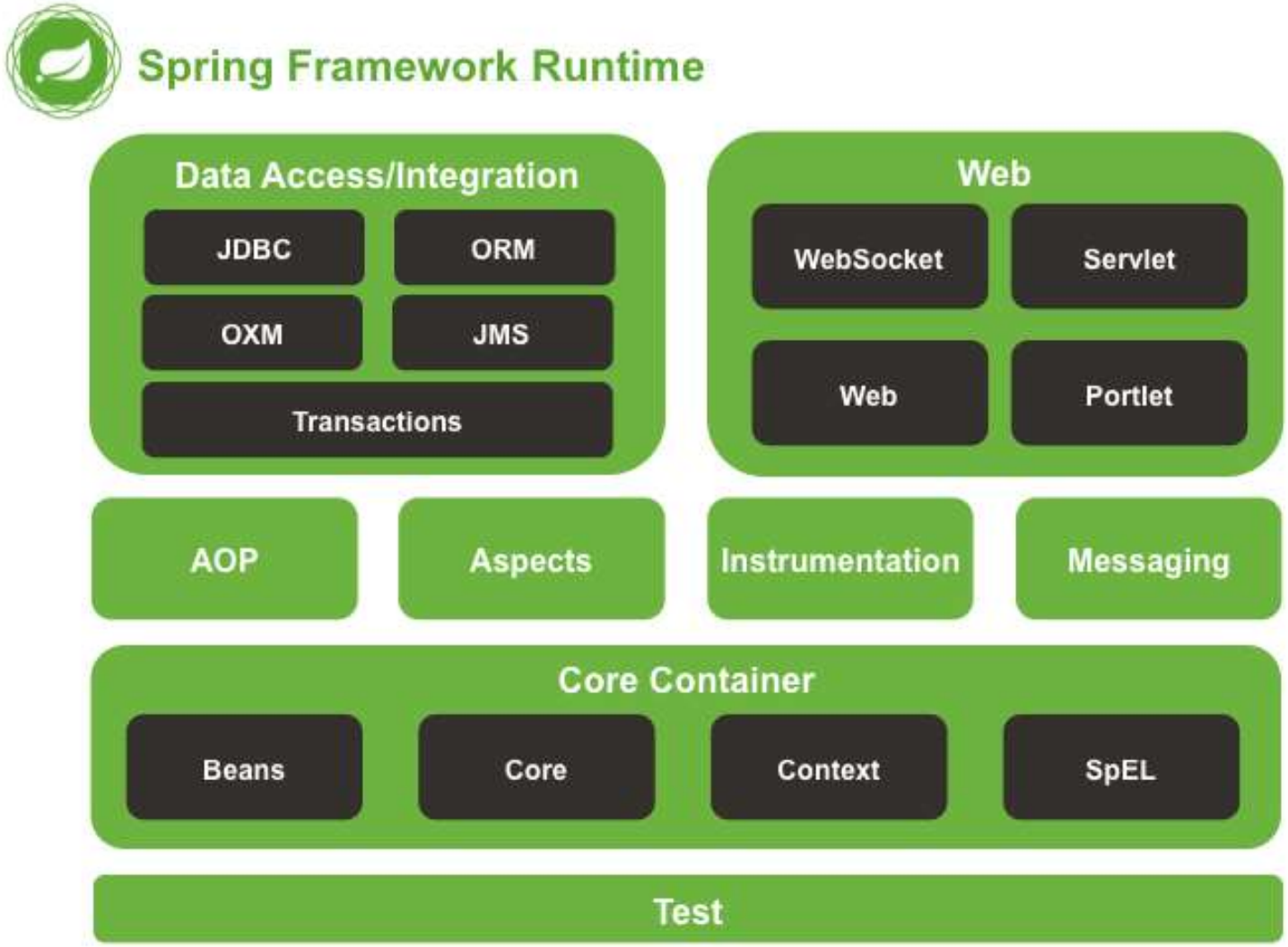
Spring : retour en 2004



Spring en 2022



Spring Data JPA

- Spring repose sur JPA pour la persistance des données.
- Hibernate est une implémentation de JPA.
- Pour comprendre JPA, il faut parcourir les annotations du mémento JPA, validation, Spring

Spring Data

- **L'objectif de Spring Data est de simplifier la mise en œuvre de la couche DAO.**
- **Spring Data comprend :**
- **1- Trois interfaces dont nos interfaces pourront hériter**
- **2- des implémentations de ces 3 interfaces**
- **3- l'annotation @Query**
- **4- les requêtes par dérivation (désigné en anglais par « Query methods ») : le nom de la méthode Java est interprété par Spring Data et traduit en requête HQL**
- **5- une solution performante pour les tris et la pagination**

Spring Data : les interfaces Repository

- **- CrudRepository**
- **- PagingAndSortingRepository**
- **- JpaRepository**

https://github.com/spring-projects/spring-data-jpa/blob/main/spring-data-jpa/src/main/java/org/springframework/data/jpa/repository/JpaRepository.java

https://github.com/spring-projects/spring-data-jpa/blob/main/spring-data-jpa/src/main/java/org/springframework/data/jpa/repository/JpaRepository.java

Spring Data : les interfaces Repository

- Toutes les interfaces DAO de nos projets hériterons d'une de ces 3 interfaces.
- Dans nos services, la récupération d'un objet utilise les méthodes findById et orElse comme suit :

```
@Override
public Utilisateur recupererUtilisateur(Long id) {
    return utilisateurDao.findById(id).orElse(null);
}
```

Spring Data 2.x intègre la notion d'Optional (nouveau concept de Java 8) : la méthode findById renvoie un objet de type Optional

Spring Data : implémentation des interfaces Repository

- Spring Data utilise SimpleJpaRepository comme point de départ pour implémenter nos DAO :

https://docs.spring.io/spring-data/data-jpa/docs/current/api/org/springframework/data/jpa/repository/support/SimpleJpaRepository.html

https://docs.spring.io/spring-data/data-jpa/docs/current/api/org/springframework/data/jpa/repository/support/SimpleJpaRepository.html

Spring Data : l'annotation @Query

- L'annotation @Query se place au dessus des méthodes des interfaces de la couche DAO (aka Repositories). Elle s'accompagne d'une requête HQL :

```
@Query("FROM Theme t ORDER BY
size(t.enquetes) DESC")
List<Theme> findThemesSortedByNbEnquetes();
```

Des requêtes SQL peuvent être utilisées grâce à l'attribut nativeQuery affecté à vrai :

```
@Query(value="SELECT id, nom FROM theme
ORDER BY id DESC", nativeQuery=true)
List<Theme> findThemes2();
```

Spring Data : Query method

Les requêtes par dérivation :

Table 3. Supported keywords inside method names

Keyword	Sample	JPQL snippet
`Distinct`	`findDistinctByLastnameAndFirstname`	`select distinct … where x.lastname = ?1 and x.firstname = ?2`
`And`	`findByLastnameAndFirstname`	`… where x.lastname = ?1 and x.firstname = ?2`
`Or`	`findByLastnameOrFirstname`	`… where x.lastname = ?1 or x.firstname = ?2`
`Is`, `Equals`	`findByFirstname`, `findByFirstnameIs`, `findByFirstnameEquals`	`… where x.firstname = ?1`
`Between`	`findByStartDateBetween`	`… where x.startDate between ?1 and ?2`
`LessThan`	`findByAgeLessThan`	`… where x.age < ?1`
`LessThanEqual`	`findByAgeLessThanEqual`	`… where x.age <= ?1`

Spring Data : Query method

---

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation

Spring Data : Tri et pagination

Spring Data apporte une solution élégante au besoin de pagination grâce aux interfaces Slice, Page, PageRequest et Pageable