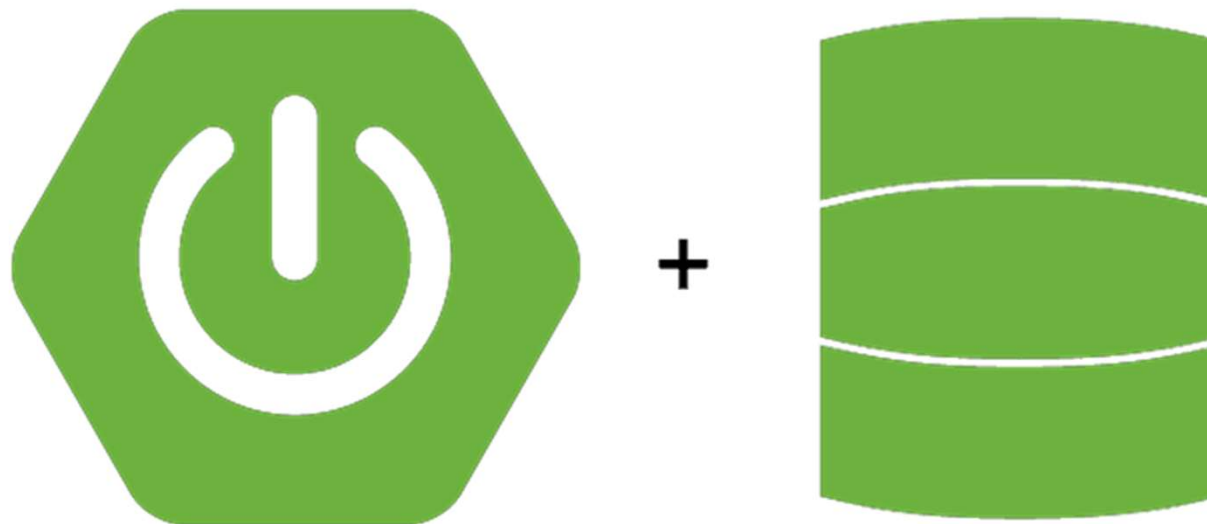


# SPRING DATA JPA – CrudRepository



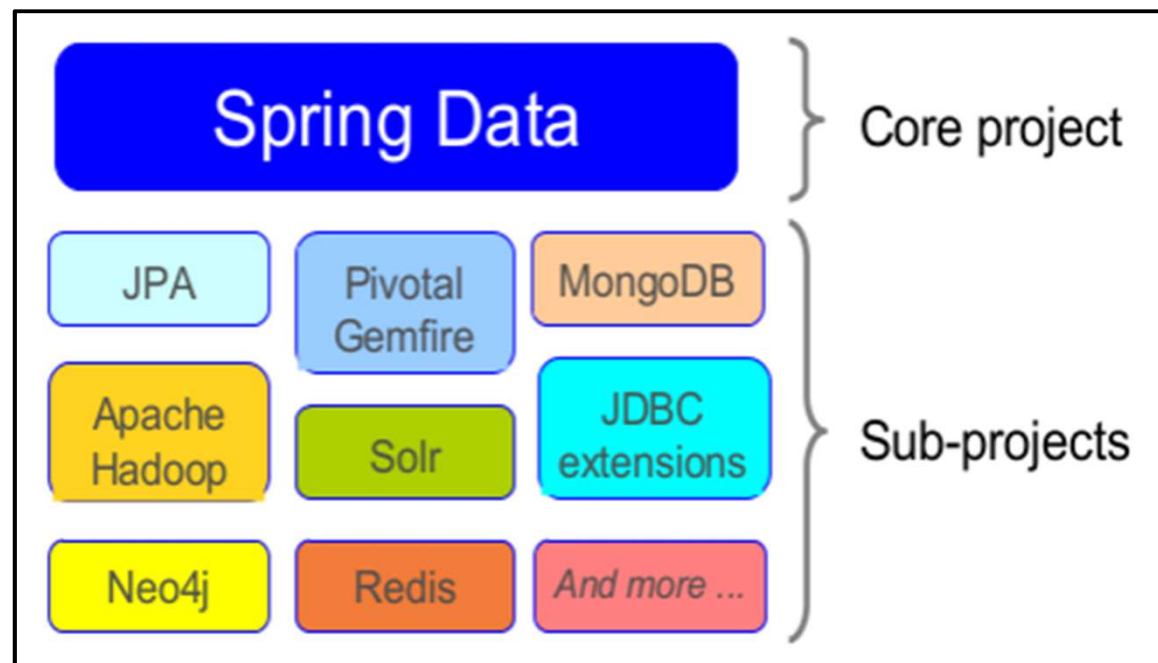
UP ASI  
Bureau E204

# Plan du Cours

- Spring Data
- Spring Data JPA
- CRUD Repository interface
- Créer et Utiliser un Repository
- TP Spring Boot + Spring Data JPA

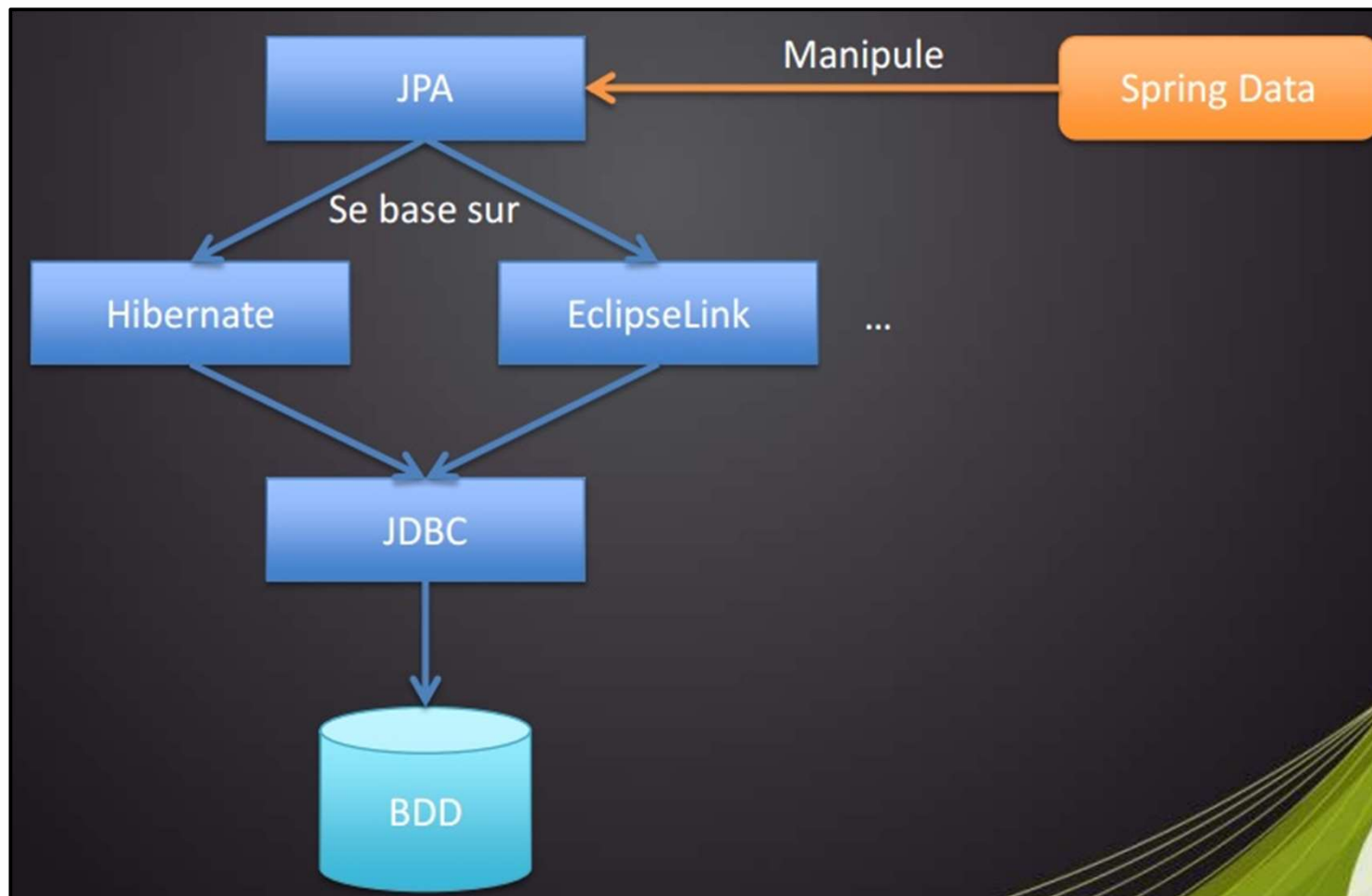
# SPRING DATA

- C'est un module Spring qui a pour but de :
  - Faciliter l'écriture des couches d'accès aux données.
  - Offrir une abstraction commune pour l'accès aux données quelle que soit la source de données (SQL ou NoSQL).



# SPRING DATA JPA

- Spring Data JPA est un sous projet du projet Spring Data.



# SPRING DATA JPA

- Code répétitif avec les DAO (Data Access Object) avant Spring Data JPA:

```
public class ProjetDAO {  
  
    public Projet findById (Long id) {  
        //code ...  
    }  
  
    public List<Projet> findAll{  
        //code ...  
    }  
  
}
```

```
public class EntrepriseDAO{  
  
    public Entreprise findById (Long id) {  
        //code ...  
    }  
  
    public List<Entreprise> findAll{  
        //code ...  
    }  
  
}
```

# SPRING DATA JPA

- Avec Spring Data JPA, il vous suffit de définir une interface qui permet de manipuler une entité, en étendant l'interface `CrudRepository <T, ID>` et de déclarer les méthodes pour manipuler cette entité.
- Spring Data JPA créera une classe qui implémente cette interface pour vous.
- Exemple :

```
public interface ProjetRepository extends JpaRepository<Projet, Long> {..}
```

- L'interface **ProjetRepository** étend l'interface **CrudRepository<Projet, Long>**. Elle contient les méthodes pour manipuler l'entité `Projet`.
- Spring Data JPA va automatiquement créer une classe qui implémente cette interface au moment de l'exécution de la l'application.

# SPRING DATA JPA

- Spring Data JPA va créer le code pour les méthodes que tu souhaites utiliser. Il suffit de lui indiquer les méthodes à utiliser :

Keyword	Sample	Equivalent to
<b>GreaterThan</b>	<code>findBydateDebutGreaterThan(Date dateC);</code>	Select p from ProjetDetail p where p.dateDebut > :dateC
<b>LessThan</b>	<code>findBydateDebutLessThan(Date dateN);</code>	Select p from ProjetDetail p where p.dateDebut < :dateN
<b>Between</b>	<code>findBydateDebutBetween(Date dFrom, Date dTo);</code>	Select p from ProjetDetail p where p.dateDebut between :dFrom and :dTo
<b>IsNotNull, NotNull</b>	<code>findByCout_provisoireNotNull();</code>	Select p from ProjetDetail p where p.cout_provisoire is not null
<b>IsNull, Null</b>	<code>findByDescriptionNull();</code>	Select p from ProjetDetail p where p.description is null
<b>Like</b>	<code>findByTechnologieLike(String technologie);</code>	Select p from ProjetDetail p where p. technologie like :technologie
<b>(No keyword)</b>	<code>findByTechnologie (String technologie);</code>	Select p from ProjetDetail p where p. technologie= :technologie
<b>(No keyword)</b>	<code>findOne(ID primaryKey);</code>	Select p from ProjetDetail p where p.id =:primaryKey
<b>(No keyword)</b>	<code>Long count();</code>	Select count(*) From Projet;

# EXAMPLE

@Repository

```
public interface ProjetDetailRepository
extends JpaRepository<ProjetDetail, Long>
{
```

```
// SELECT * FROM ProjetDetail WHERE technologie LIKE '%in%';
```

```
List<ProjetDetail> findByTechnologieLike(String technologie);
```

```
List<ProjetDetail> findByTechnologieContains(String technologie);
```

```
List<ProjetDetail> findByTechnologieContaining(String technologie);}
```

```
@Entity
@Table(name = "T_PROJET_DETAIL")
public class Projet_Detail implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "PD_ID")
    private Long id; // Identifiant projet detail (Clé p
    @Column(name = "PD_DESCRIPTION")
    private String description;
    @Column(name = "PD_TECHNOLOGIE")
    private String technologie;
    @Column(name = "PD_COUT_PROVISIOIRE")
    private Long cout provisoire;
    @Temporal(TemporalType.DATE)
    private Date dateDebut;
    @OneToOne(mappedBy = "projetDetail")
    private Projet projet;
}
```



# EXEMPLE

- Afficher la liste des projets qui ont une technologie précise.

@Repository

```
public interface ProjetRepository extends JpaRepository<Projet, Long> {  
    List<Projet> findByProjetDetailTechnologieContains(String technologie);  
}
```

association                      attribut dans la table Projet Detail

```
@Entity  
@Table(name = "T_PROJET")  
public class Projet implements Serializable {  
    private static final long serialVersionUID = 1L;  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(name = "PROJET_ID")  
    private Long id; // Identifiant projet (Clé primaire)  
  
    @Column(name = "PROJET_SUJET")  
    private String sujet;  
  
    @OneToOne  
    private ProjetDetail projetDetail;  
}
```

# EXEMPLE

- Afficher la liste des projets d'une équipe.

@Repository

```
public interface ProjetRepository {
```

```
    extends JpaRepository<Projet, Long> {
```

```
        List<Projet> findByEquipesIdEquipe(Long equipeId);
```

```
    }
```

```
@Entity
@Table(name = "T_PROJET")
public class Projet implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "PROJET_ID")
    private Long id; // Identifiant projet (Clé primaire)

    @Column(name = "PROJET_SUJET")
    private String sujet;

    @ManyToMany(mappedBy = "projets", cascade = CascadeType.ALL)
    private Set<Equipe> equipes;
```

# EXEMPLE

- Afficher la liste des projets d'une équipe dont la description est non nulle.

@Repository

```
public interface ProjetRepository extends JpaRepository<Projet, Long> {  
    List<Projet>  
    findByEquipesIdEquipeAndProjetDetailDescriptionNotNull(Long equipeId);  
}
```

# EXEMPLE

@Repository

```
public interface ProjetRepository extends JpaRepository<Projet, Long> {
```

- Afficher la liste des projets par équipe et entreprise.

```
List<Projet> findByEquipesIdEquipeAndEquipesEntrepriseIdEntreprise  
(Long equipeId, Long entrepriseId);
```

- Afficher la liste des projets par la spécialité d'une équipe et l'adresse de l'entreprise.

```
List<Projet>  
findByEquipesSpecialiteContainsAndEquipesEntrepriseAdresseContains  
(String specialite, String adresse);  
}
```

# SPRING DATA JPA (CrudRepository)

Si vous avez des questions, n'hésitez pas à nous contacter :

**Département Informatique**  
**UP Architectures des Systèmes d'Information**  
Bureau E204