

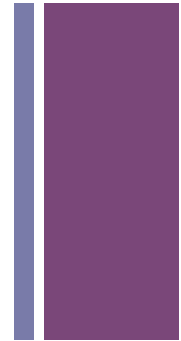
Master 2 Informatique

Java EE



Présentation

- Thierry Poutrain
- thierry.poutrain@kearis.fr
- 48h de formation (13 sessions)
- Théorie / Pratique
- Examen
- Projets en groupe





Déroulement

- Java Persistence API (ORM / Bean Validation)
- Enterprise JavaBean
- JSP et JSF
- Web Service
- Messaging
- Autour de Java EE





Le langage Java

- Orienté objet (1995)
- Sun puis Oracle
- API et bibliothèque de base
- Multi-plateforme via JVM
- Compilation
- Fortement typé
- GC
- Encapsulation / Polymorphisme





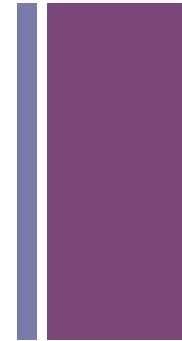
Le langage Java

- Java SE 6 : 2006
- Java SE 7 : 2011
- Java SE 8 : 2014
- Java SE 9 : 2016
- Et Java EE... c'est quoi ?





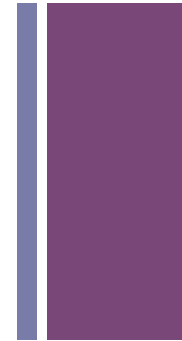
Java EE



- APIs Java : Collection / Hashmap / IO / ...
 - On les utilise et on invente rien
- Java EE : idem
- Avec des fonctionnalités d'entreprise :
 - persistance / sécurité / webservice ...
- Spécifications dont l'implémentation permet de créer des applications d'entreprise
- Spécifications selon processus standardisé
 - établies selon le *Java Community Process*
 - regroupées au travers des *Java Specification Requests*



Java EE



■ Les *Java Specification Requests* :

■ Pourquoi :

- Système normalisé
- Rôles bien précis
- Optionnelle / Obligatoire

■ Sur chaque techno de Java

■ <https://www.jcp.org/en/jsr/all>

■ Mis en avant via le Java Community Process

■ <https://www.jcp.org/en/home/index>

■ Exemples :

- NIO, *JDBC*, Java Compiler, OSGi (Java SE)
- JMS, EJB, JPA, JSF (Java EE)
- Java USB, Bluetooth, MMAPI (Java ME)



Java EE

- Les *Java Specification Requests* :
 - Historique :

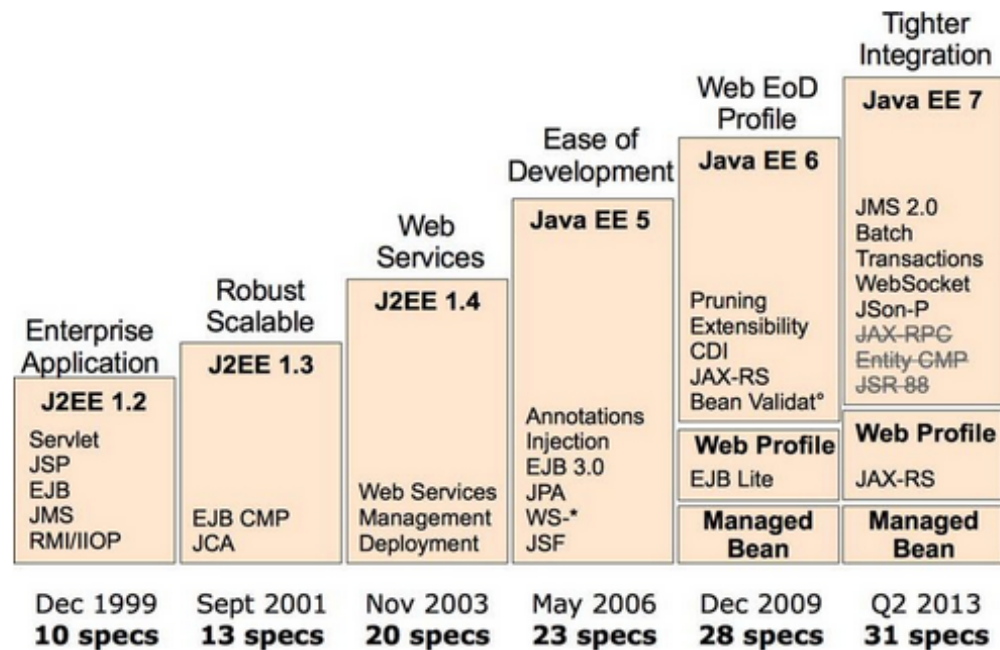
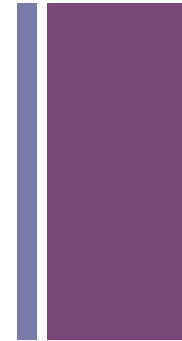


Figure 1-4. History of J2EE/Java EE



Java EE

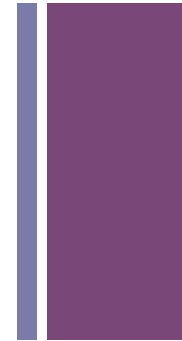


- Chaque implémentation doit supporter :
 - Applet (swing dans navigateur)
 - Application (gui/batch)
 - Web application
 - Enterprise application

- Installation du socle de développement
 - JDK 7
 - IDE Eclipse
 - Implémentation Glassfish 4
 - Maven 3



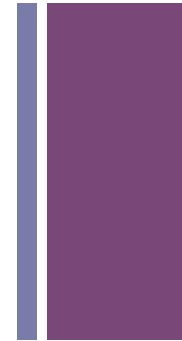
Java Persistence API



- Pourquoi persister des données
 - Manipuler / Enregistrer / Rechercher
 - Procédure stockée / Index / Relation
- Comment l'intégrer dans un langage OO
 - Manipule des objets
 - Encapsule des états
 - Constructeur et GC mais ne perdure pas
 - Utilise alors un Object-Relation Mapping
- Les Frameworks
 - Hibernate
 - TopLink
 - ...
- Utilisation Java EE 7 et JPA 2.1
- IR = EclipseLink 2.5 (avec XML Binding)



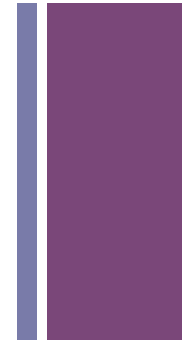
Java Persistence API



- JPA 2.1
 - JPA depuis Java EE 5
 - JDBC sans SQL
- Composé de :
 - ORM (manipuler les données)
 - EntityManager / JPQL (requêtage)
 - JTA (transactions)
 - Callback / listener
- Historique
 - Avant hibernate
 - Hibernate
 - Java EE 5 : JPA 1.0
 - Java EE 6 : JPA 2.0
 - Java EE 7 : JPA 2.1



JPA - Entity



- Objets à persister
- Mapping via annotations
 - @Entity - Les entités à sauvegarder
 - @Id - Les identifiants de ces entités
 - @Table - Les tables du SGBD
 - @Column - Les colonnes des tables



```
@Entity
@Table(name = "car")
public class Voiture {

    @Id
    private Long id;

    @Column(name = "color")
    private String couleur;

    // =====
    // =          Constructors, Getters & Setters          =
    // =====
}
```



JPA - Entity

- Clés primaires :
 - ID Auto
- Tables secondaires
- Clés composées :
 - Embeddable (EmbeddedId)
 - IdClass
- Basic et Large Object





```
@Entity
public class Voiture {

    @Id
    @GeneratedValue(strategy =
        GenerationType.AUTO)
    private Long id;

    //...
}
```



```
@Entity
@Table(name = "address")
@SecondaryTables({
    @SecondaryTable(name = "city"),
    @SecondaryTable(name = "country")
})
public class Adresse {

    @Id
    private Long id;

    private String rue;

    @Column(table = "city")
    private String ville;

    // ...
}
```





@Embeddable

```
public class NouvellesId {
```

```
    private String titre;
```

```
    private String langage;
```

```
    // ...
```

```
}
```

```
@Entity
```

```
@Table(name = "news")
```

```
public class Nouvelles {
```

```
    @EmbeddedId
```


```
    private NouvellesId id;
```

```
    private String contenu;
```

```
    // ...
```

```
}
```



 @Embeddable
public class NouvellesId {

private String titre;

private String langage;

// ...

}

@Entity
@Table(name = "news")
@IdClass(NouvellesId.class)
public class Nouvelles {

@Id
private String titre;

@Id
private String langage;

private String contenu;

// ...

}





```
@Entity
public class Musique {

    @Basic(fetch = FetchType.LAZY)
    @Lob
    private byte[] wav;

    //...
}
```

+

JPA - Entity



TP 1 à 9



JPA - Entity

- Temporal
- Transient
- Enumeration
- Access Type (FIELD / PROPERTY)
- Collection / Map of Basic
- XML Mapping





```
@Entity
public class Voiture {

    @Temporal(TemporalType.DATE)
    private Date dateCreation;

    @Transient
    private Integer puissance;

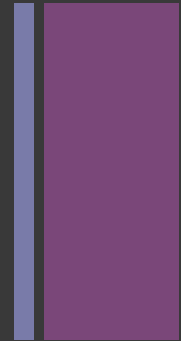
    @Temporal(TemporalType.TIMESTAMP)
    private Date dateImmatriculation;

    //...
}
```



```
public enum Couleur {  
  
    ROUGE,  
    JAUNE,  
    BLANC  
  
}
```

```
@Entity  
public class Voiture {  
  
    @Enumerated(EnumType.STRING)  
    private Couleur couleur;  
  
    //...  
}
```





```
@Entity
public class Livre {

    @ElementCollection(fetch = FetchType.LAZY)
    @CollectionTable(name = "tag")
    @Column(name = "tag_value")
    private List<String> tags = new ArrayList<>();

    //...
}
```




```
@Entity
public class CD {

    @ElementCollection
    @CollectionTable(name = "cd")
    @MapKeyColumn(name = "position")
    @Column(name = "titre")
    private Map<Integer, String> chanson = new HashMap<>();

    //...
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
+<entity-mappings xmlns="http://xmlns.jcp.org/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://xmlns.jcp.org/xml/ns/persistence/orm http://xmlns.jcp.org/xml/ns/persistence/orm_2_1.xsd"
  version="2.1">
```

```
<entity class="org.kearis.formation.javaee7.chapter01.ex25.Book25">
  <table name="livre"/>
  <attributes>
    <basic name="titre">
      <column name="titre_livre" nullable="false" updatable="false"/>
    </basic>
    <basic name="description">
      <column length="2000"/>
    </basic>
    <basic name="nbPage">
      <column name="nb_page" nullable="false"/>
    </basic>
  </attributes>
</entity>

</entity-mappings>
```

```
@Entity
@Table(name = "pas_pris_en_compte")
public class Book25 {

    @Id
    private Long id;
    private String title;
    @Column(length = 500)
    private String description;
    private Integer nbOfPage;

}
```

+

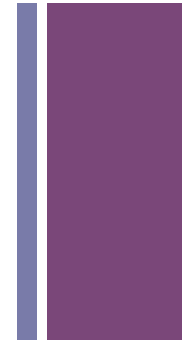
JPA - Entity



TP 14 à 25



JPA - Relationnel



- Relation entre objets et tables
- Orienté pour les BDD relationnelles
- Plusieurs type de relations :
 - OneToOne
 - OneToMany
 - ManyToOne
 - ManyToMany
- Uni / Bi directionelle
- Définir qui porte la relation
- Stratégie de FETCH, CASCADE
- OrderBy >> OrderColumn



```
@Entity
public class Client {

    @OneToOne (fetch = FetchType.LAZY)
    @JoinColumn(name = "address_fk", nullable = false)
    private Adresse adresse;

    //...
}
```



@Entity

```
public class Commande {
```

```
    @OneToMany
```

```
    @JoinTable(name = "ord_line",
```

```
        joinColumns = @JoinColumn(name = "order_fk"),
```

```
        inverseJoinColumns = @JoinColumn(name = "order_line_fk"))
```

```
    private List<LigneCommande> lignesCommande;
```

```
    //...
```

```
}
```

@Entity

```
public class Commande {
```

```
    @OneToMany(fetch = FetchType.EAGER)
```

```
    @JoinColumn(name = "order_fk")
```

```
    @OrderBy("numeroLigne DESC")
```

```
    private List<LigneCommande> lignesCommande;
```

```
    //...
```

```
}
```





```
@Entity
public class CD {

    @ManyToMany(mappedBy = "apparitionSurCd")
    private List<Artist> artistes;

}
```

```
@Entity
public class Artiste {

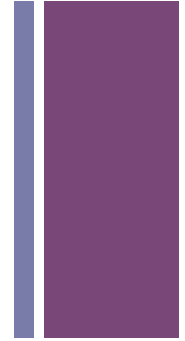
    @ManyToMany
    @JoinTable(name = "jnd_art_cd",
        joinColumns = @JoinColumn(name = "artist_fk"),
        inverseJoinColumns = @JoinColumn(name = "cd_fk"))
    private List<CD> apparitionSurCd;

}
```



+

JPA - Relationnel



TP 34 à 51



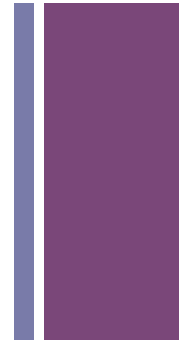
JPA - Héritage



- Stratégie:
 - SINGLE TABLE (défaut, une seule table)
 - JOINED (une table par classe)
 - TABLE PER CLASS (une table par classe concrète)

+

JPA - Héritage



TP 53 à 66