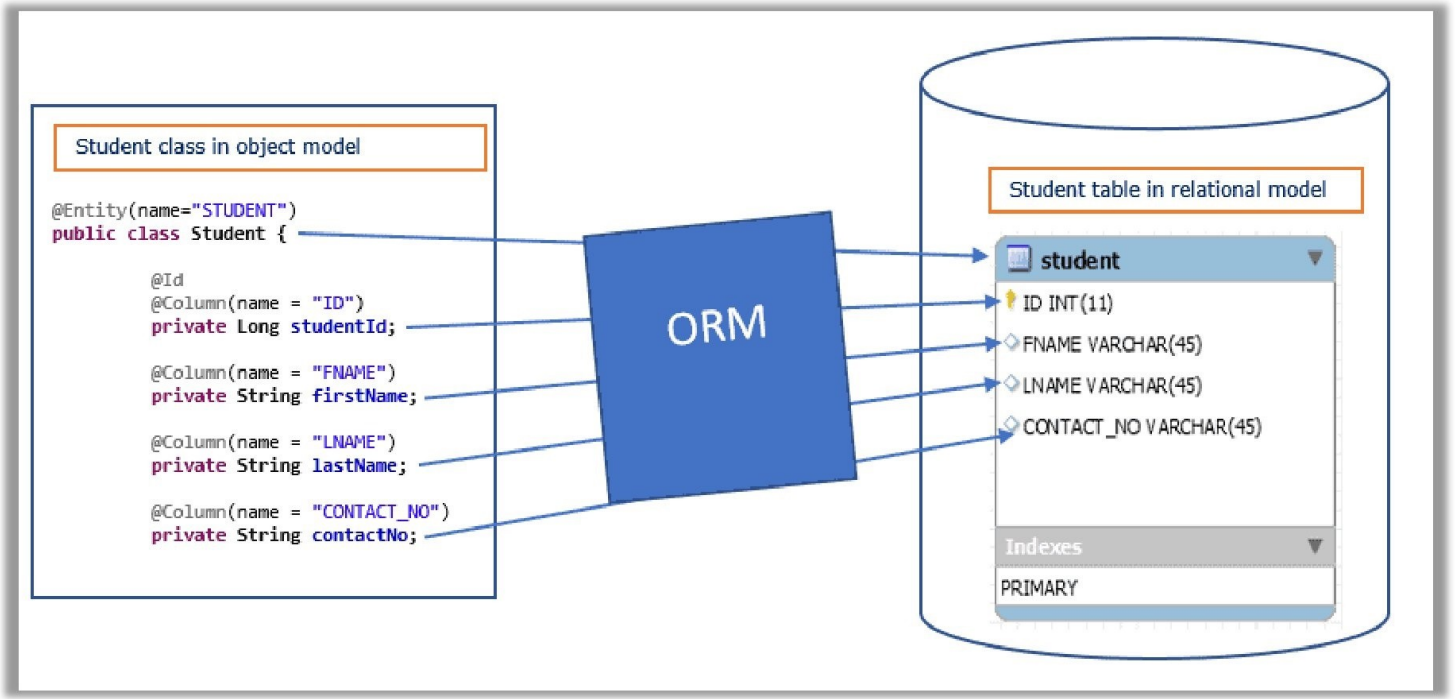
Java Persistence API

Part-I

Object Relation Mapping (ORM)

- هي تقنية تقوم بعمل Mapping بين Object وقاعدة البيانات بحيث يمكن حفظ بيانات Object في قاعدة البيانات واسترجاعها منها.



Java Persistence API

- عبارة عن Java Specification تعمل على أدارة البيانات بين تطبيقات الجافا و Relational Database من خلال تقنية ORM والتي تعمل على الربط بين Object Oriented Domain و Relational Database Systems .
- تقدم مجموعة من classes و methods لحفظ البيانات في قاعدة البيانات.
- يوجد مجموعة من Implementation لل JPA اشهرها:
  - Hibernate
  - Toplink

Entity

- Entity هي Object الذي يتم عمل mapping له مع قاعدة البيانات ليتم حفظه فيها واسترجاعه منها.
- Entity هي Object التي تبقى لفترة بسيطة في الذاكرة وبشكل مستمر في قاعدة البيانات.
- هي عبارة POJO Class لديها مجموعة من Attributes يتم التعامل معها من خلال getters و setters.

```
@Entity
public class Book {

  @Id @GeneratedValue
  private Long id;
  private String title;
  private Float price;
  private String description;
  private String isbn;
  private Integer nbOfPage;
  private Boolean illustrations;

  public Book() {
  }

  // Getters, setters
}
```

Entity

- هي عبارة عن domain object يمثل جدول في قاعدة البيانات.
- كل instance منها يمثل صف من الجدول.
- تستخدم مجموعة من annotations لعمل mapping لها مع قاعدة البيانات وهي موضحة بالجدول التالي:

Annotation Target	Usage
Class	@Entity, @Table
Field, Method	@Id, @Column, @GeneratedValue, @JoinColumn, @OneToOne, @OneToMany, @ManyToMany, @ManyToOne

Anatomy of An Entity

```java
@Entity
public class Book implements Serializable{


    @Id
    @GeneratedValue()
    private int id;
    private String title;
    private String author;
    private Float price;
    private Date publishingDate;
    private int number;
    private String type;

    public Book() {
    }

   // getter and setter
```

- يجب عمل Annotation على class باستخدام @Enity .
- استخدام @Id لتحديد primary key .
- يجب ان تحتوي على no-arg constructor .
- لا يجب ان تكون final class .
- أن تقوم بعمل implementation لل Serializable interface.

EntityManager

- يستخدم لإدارة Entities الموجودة في Persistence Context.
- يستخدم لأجراء عمليات الكتابة ، القراءة ، الحذف و الاستفسار عن Entities .
- يتم الحصول عليه من خلال Container ويتم عمل Inject له في المكان الذي نريد استخدامه فيه.

Persistence Context

- هو عبارة عن مجموعة من entities instances و كل Entity تكون ممثل فيه ب instance واحد فقط، ومن خلالها تتم إدارة دورة حياة entities .

Persistence Unit

- يتم فيها تحديد نوع قاعدة البيانات المستخدمة و connection parametrs وغيرها من الاعداد التي تساعد EntityManager في التعامل مع قاعدة البيانات.

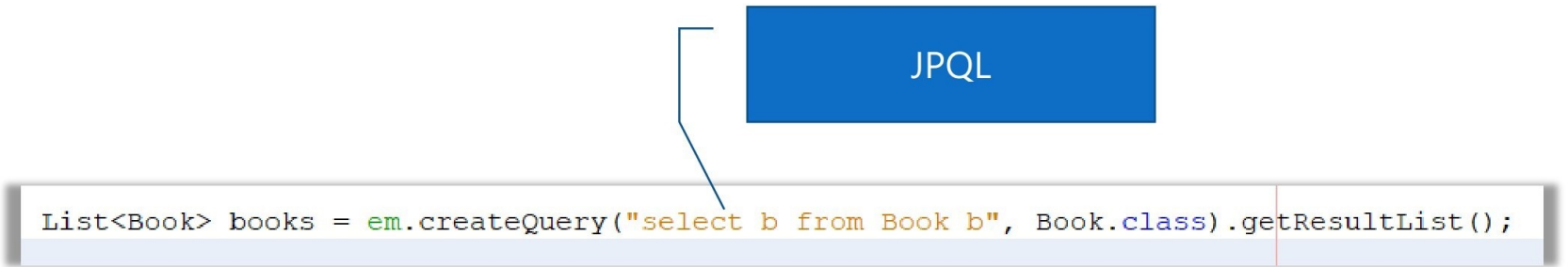
```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2" xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
              http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">

  <persistence-unit name="JPALecturePU" transaction-type="JTA">
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.jdbc.url"
                value="jdbc:mysql://localhost:3308/jpalecture?zeroDateTimeBehavior=CONVERT_TO_NULL"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
      <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.password" value="12345"/>
      <property name="javax.persistence.schema-generation.database.action" value="drop-and-create"/>
    </properties>
  </persistence-unit>

</persistence>
```

Java Persistence Query Language (JPQL)

- هي عبارة عن Object Oriented Query Language تستخدم لاجراء database operations على < persistence entities
- تستخدم SQL like syntax ولكنها لا تعمل على قاعدة البيانات مباشرة بل على entites.



Integration with Bean Validation

- يمكن استخدام Bean Validation ليتم من خلالها التحقق التلقائي للبيانات المستخدم في Entity قبل استعمالها.

```
@Entity
public class Book {

  @Id @GeneratedValue
  private Long id;
  @NotNull
  private String title;
  private Float price;
  @Size(min = 10, max = 2000)
  private String description;
  private String isbn;
  private Integer nbOfPage;
  private Boolean illustrations;

  // Constructors, getters, setters
}
```

Example

- المثال التالي يبين استخدام JPA في حفظ بيانات كتاب وعرض كل الكتب المحفوظة بالإضافة إلى عرض بيانات كتاب.

```
@Entity
public class Book implements Serializable{


    @Id
    @GeneratedValue()
    private int id;
    private String title;
    private String author;
    private Float price;
    private Date publishingDate;
    private int number;
    private String type;

    public Book() {
    }

   // getter and setter
```

Book EJB

```java
@Stateless
public class BookEJB {

    @PersistenceContext
    EntityManager em;

    public void createBook(Book book) {
        em.persist(book);
    }

    public List<Book> getBooks() {

        return em.createQuery("select b from Book b", Book.class).getResultList();

    }
}
```

Book Managed Bean

```java
@Named(value = "bookBean")
@RequestScoped
public class BookBean {

    private Book book = new Book();

    @Inject
    BookEJB bookEJB;

    public BookBean() {
    }

    public Book getBook() {
        return book;
    }

    public String showBook(Book book) {
        this.book = book;
        return "view_book";
    }

    public String createBook() {
            bookEJB.createBook(book);

        return "list_all_books";
    }

     public List<Book> getBooks() {
        return bookEJB.getBooks();
    }

}
```

Create Book Page

```
<h1>
    <h:outputText  value="Create Book Page"/>
</h1>
<h:messages />
<h:form>
    <h:panelGrid columns="2">
        <h:outputLabel value="Title: "/>
        <h:inputText value="#{bookBean.book.title}" required="true"
                     requiredMessage="Please enter book title"/>
        <h:outputLabel value="Author: "/>
        <h:inputText value="#{bookBean.book.author}" required="true"
                     requiredMessage="Please enter author name" />
        <h:outputLabel value="Publishing Date: "/>
        <h:inputText value="#{bookBean.book.publishingDate}" required="true"
                     requiredMessage="Please enter publishing date" >
            <f:convertDateTime pattern="yyyy/mm/dd"/>
        </h:inputText>
        <h:outputLabel value="Pages: "/>
        <h:inputText  value="#{bookBean.book.number}" required="true"
                      requiredMessage="Please enter number of pages"
                      validatorMessage="Please enter page number between 1-1000">
            <f:validateLongRange minimum="1" maximum="100" />
        </h:inputText>
        <h:outputLabel value="Price: "/>
        <h:inputText value="#{bookBean.book.price}" required="true"
                     requiredMessage="Please enter   price"/>

        <h:outputLabel value="Type: "/>
        <h:selectOneMenu value="#{bookBean.book.type}">
            <f:selectItem itemValue = "History" itemLabel="History"/>
            <f:selectItem itemValue = "Comics" itemLabel="Comics"/>
            <f:selectItem itemValue = "Scifi" itemLabel="Scifi"/>
        </h:selectOneMenu><br/>
    </h:panelGrid>
    <h:commandButton value="submit"  action="#{bookBean.createBook()}"/>
</h:form>
```

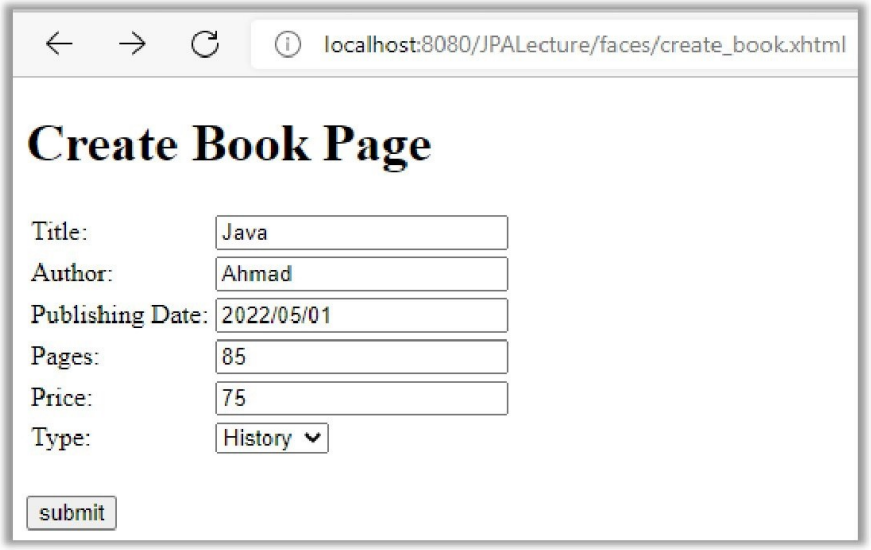
List All Books Page

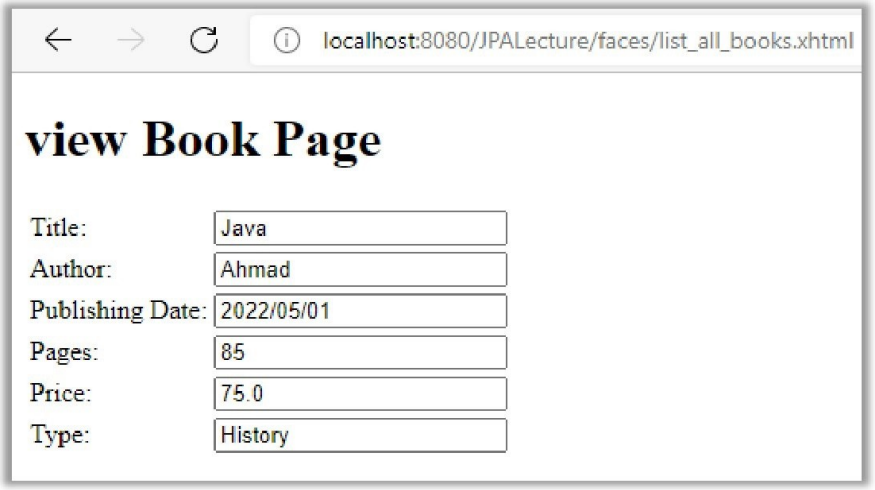
```
<h1>
    <h:outputText  value="List All Books Page"/>
</h1>
<h:form>
    <h:dataTable value="#{bookBean.books}" var="book" border="1"
                 rendered="#{not empty bookBean.books}">
        <h:column>
            <f:facet name="header">Title</f:facet>
                #{book.title}
        </h:column>
        <h:column>
            <f:facet name="header">Author</f:facet>
                #{book.author}
        </h:column>
        <h:column>
            <f:facet name="header">Price</f:facet>
                #{book.price}
        </h:column>
        <h:column>
            <f:facet name="header">publishing Date</f:facet>
                #{book.publishingDate}
        </h:column>
        <h:column>
            <f:facet name="header">No of Pages</f:facet>
                #{book.number}
        </h:column>
        <h:column>
            <f:facet name="header">View</f:facet>
            <h:commandButton value="view"  action="#{bookBean.showBook(book)}"/>
        </h:column>
    </h:dataTable>
</h:form>
```

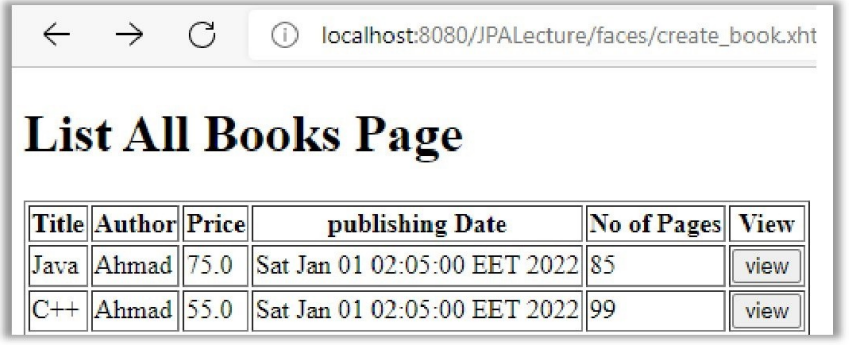
View Book Page

```
        <h1>
            <h:outputText  value="view Book Page"/>
        </h1>
<h:form>
   <h:panelGrid columns="2">
            <h:outputLabel value="Title: "/>
            <h:inputText value="#{bookBean.book.title}" readonly="true" />
            <h:outputLabel value="Author: "/>
            <h:inputText value="#{bookBean.book.author}" readonly="true" />
            <h:outputLabel value="Publishing Date: "/>
            <h:inputText value="#{bookBean.book.publishingDate}" readonly="true" >
            <f:convertDateTime pattern="yyyy/mm/dd"/>
            </h:inputText>
            <h:outputLabel value="Pages: "/>
            <h:inputText  value="#{bookBean.book.number}" readonly="true"/>
            <h:outputLabel value="Price: "/>
            <h:inputText value="#{bookBean.book.price}" readonly="true" />
            <h:outputLabel value="Type: "/>
            <h:inputText value="#{bookBean.book.type}" readonly="true" />
        </h:panelGrid>
    </h:form>
```

Result


localhost:8080/JPALecture/faces/create_book.xhtml
Create Book Page
Title: Java
Author: Ahmad
Publishing Date: 2022/05/01
Pages: 85
Price: 75
Type: History
submit


localhost:8080/JPALecture/faces/list_all_books.xhtml
view Book Page
Title: Java
Author: Ahmad
Publishing Date: 2022/05/01
Pages: 85
Price: 75.0
Type: History


localhost:8080/JPALecture/faces/create_book.xht
List All Books Page
Title Author Price publishing Date No of Pages View
Java Ahmad 75.0 Sat Jan 01 02:05:00 EET 2022 85 view
C++ Ahmad 55.0 Sat Jan 01 02:05:00 EET 2022 99 view