Bài 6
JPA

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài "Database ORM"

Tóm tắt lại các phần đã học từ bài "Database ORM"

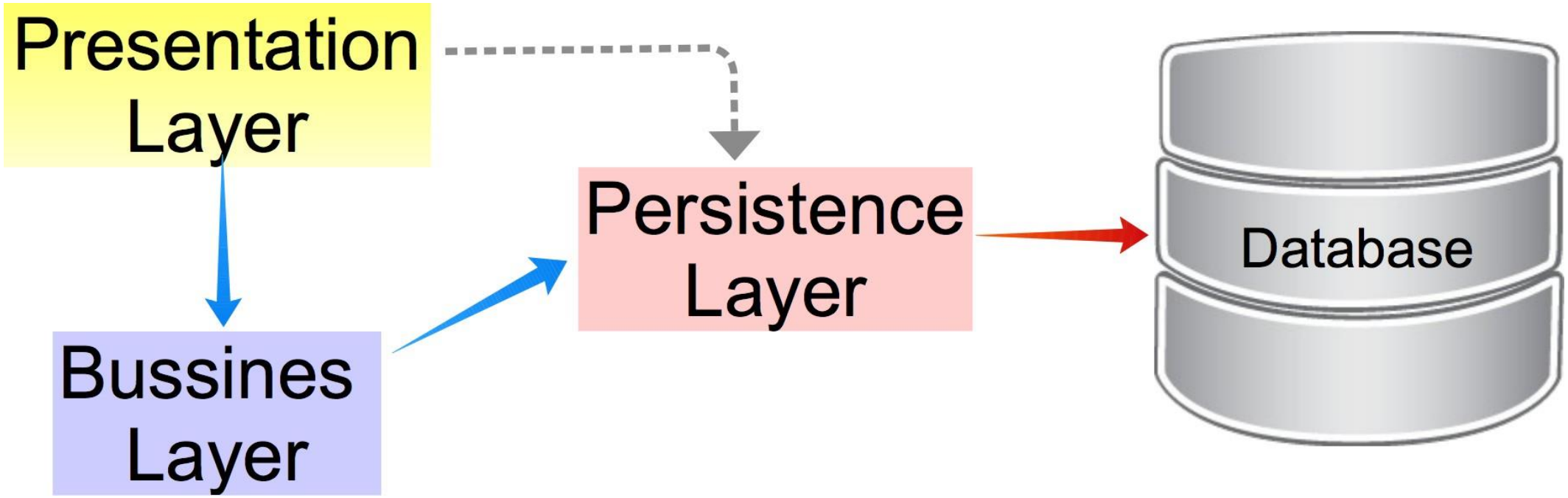
Mục tiêu

- Trình bày được JPA
- Trình bày được Entity
- Trình bày được Entity Manager
- Trình bày được hoạt động của Spring Data JPA
- Triển khai được Spring Data JPA để thao tác cơ bản với CSDL

Thảo luận

JPA – Java Persisntence API

Persistence Layer

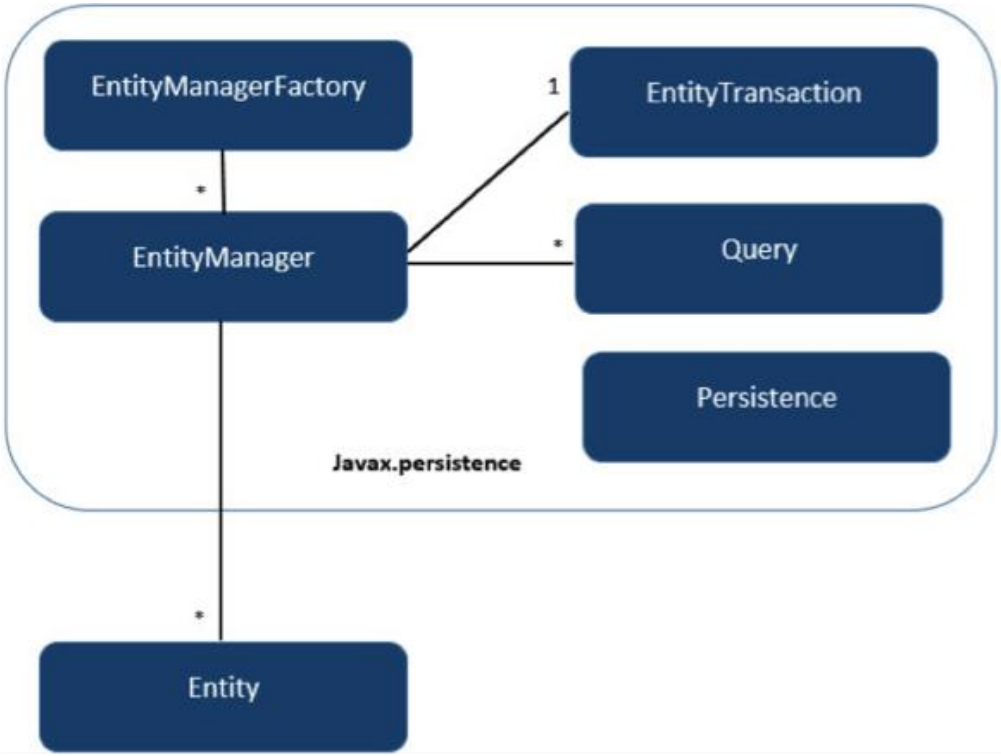


Entity

- Entity là đối tượng đại diện cho dữ liệu ở trong ứng dụng
- Entity thường là POJO (Plain Old Java Object)
- Entity sẽ được ánh xạ (mapping) tới một bảng trong CSDL
- Trong JPA, một entity cần tuân thủ:
  - Được gắn với annotation javax.persistence.Entity
  - Có một constructor là public và không có tham số
  - Không được khai báo final
  - Các biến đối tượng cần được khai báo là private, protected hoặc ở mức package-private

Persistence Context & Entity Manager

- Persistence Context là tập các thể hiện của entity được quản lý, tồn tại trong một kho dữ liệu
- Interface EntityManager:
  - Khai báo các phương thức để tương tác với persistence context
  - Tạo hoặc xoá các thể hiện của entity
  - Tìm kiếm entity theo khoá chính
  - Thực thi các câu lệnh truy vấn lên entity



Quản lý Entity

- Container-managed Entity Manager:

```
@PersistenceContext
EntityManager em;
```

- Application-managed Entity Manager:

```
@PersistenceUnit
EntityManagerFactory
emf;

EntityManager em = emf.createEntityManager();
```

Tìm một Entity

- Ví dụ:

```
@PersistenceContext
EntityManager em;
public void enterOrder(int custID, Order
  newOrder) { Customer cust =
  em.find(Customer.class, custID);
  cust.getOrders().add(newOrder);
  newOrder.setCustomer(cust);
}
```

Lưu trữ Entity

- Ví dụ:

```
@PersistenceContext
EntityManager em;
...
public LineItem createLineItem(Order order, Product product, int
   quantity) { LineItem li = new LineItem(order, product,
   quantity); order.getLineItems().add(li);
   em.persist(li);
   return li;
}
```

Xoá Entity

- Ví dụ:

```
public void removeOrder(Integer orderId) {
  try {
    Order order = em.find(Order.class, orderId);
    em.remove(order);
  }...
```

Câu lệnh truy vấn động

- Phương thức createQuery() của lớp EntityManager giúp tạo các câu truy vấn động (dynamic query)
- Ví dụ:

```
public List findWithName(String name) {
  return em.createQuery(
      "SELECT c FROM Customer c WHERE c.name LIKE
      :custName")
      .setParameter("custName", name)
      .setMaxResults(10)
      .getResultList();
}
```

Named Query

- Phương thức createNamedQuery() của lớp EntityManager giúp tạo các câu truy vấn tĩnh (static query)
- Ví dụ, khai báo Named Query:

```
@NamedQuery(
    name="findAllCustomersWit
    hName",
    query="SELECT c FROM Customer c WHERE c.name LIKE :custName"
)
```

- Sử dụng Named Query

```
@PersistenceContext
public EntityManager em;
    ...
    customers = em.createNamedQuery("findAllCustomersWithName")
    .setParameter("custName", "Smith")
    .getResultList();
```

Named Parameter

- Tên của các tham số bắt đầu bằng dấu (:)
- Ví dụ:

```
public List findWithName(String name) {
  return em.createQuery(
       "SELECT c FROM Customer c WHERE c.name LIKE :custName")
       .setParameter("custName", name)
       .getResultList();
}
```

- Sử dụng phương thức setParameter() để truyền giá trị

Positional Parameter

- Vị trí của các tham số bắt đầu bằng dấu (?)
- Ví dụ:

```
public List findWithName(String name) {
  return em.createQuery(
      "SELECT c FROM Customer c WHERE c.name LIKE ?1")
      .setParameter(1, name)
      .getResultList();
}
```

Demo

JPA – Java Persisntence API

Quản lý Transaction

```
@PersistenceContext
  EntityManagerFactory
  emf; EntityManager
  em; @Resource
  UserTransaction utx;
  ......
        em = emf.createEntityManager();
        try {
              utx.begin();
              em.persist(SomeEntit
              y);
              em.merge(AnotherEn
              tity);
              em.remove(ThirdEntit
              y); utx.commit();
        } catch (Exception e) {
              utx.rollback();
        }
```

Thảo luận

Spring Data JPA

Spring Data JPA

Cải tiến JPA tiêu chuẩn

Đơn giản hoá tầng truy xuất dữ liệu

Tự tạo repository

Truy vấn DSL

Ghi log, phân trang

Có thể tuỳ biến nếu cần thiết

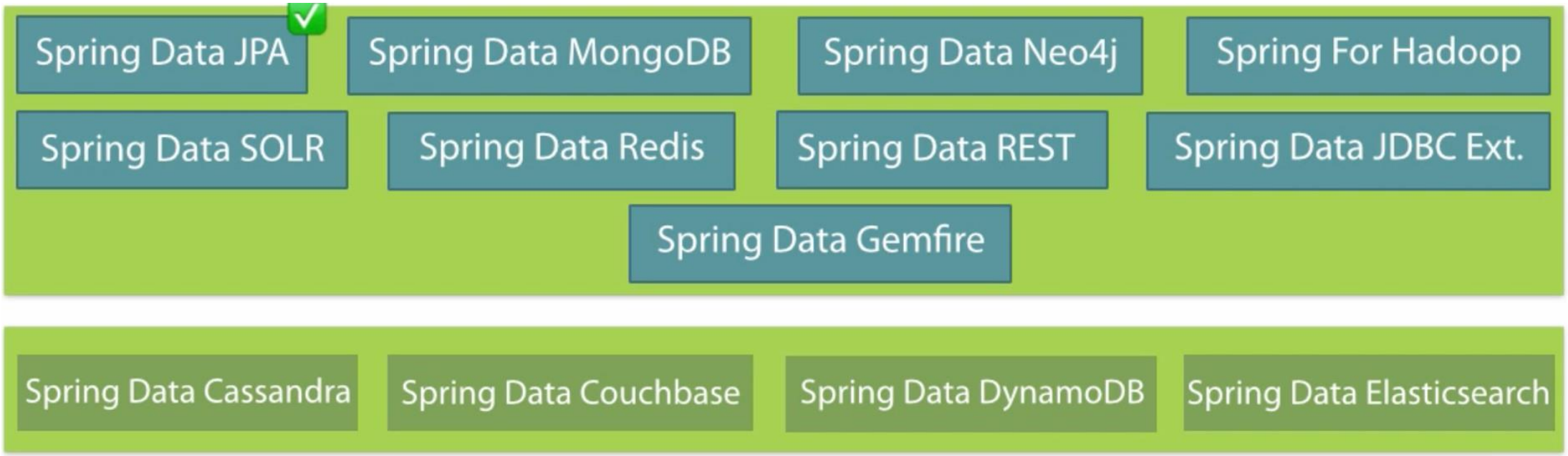
Lựa chọn Tầng truy xuất dữ liệu

JDBC Spring JDBC	JEE 7 Batch Spring Batch Hadoop	ORM JPA/Hibernate Spring Data JPA	NoSQL MongoDB Spring Data Mongo
• Đơn giản • Thuần SQL	• Rất nhiều câu lệnh ghi SQL được thực hiện	• Dễ truy xuất các mối quan hệ	• Nhóm các dữ liệu có quan hệ với nhau

Spring Data

Spring Data Commons

- Repository
- Cross-Store persistency
- Dynamic query generation



Cấu hình Spring Data JPA

build.gradle

```
dependencies {
  compile group: 'org.springframework.data', name: 'spring-data-jpa', version:
  '2.0.7.RELEASE'
  compile group: 'jstl', name: 'jstl', version: '1.2'
  compile group: 'mysql', name: 'mysql-connector-java', version: '8.0.11'
  testCompile group: 'org.junit.jupiter', name: 'junit-jupiter-engine', version:
  '5.2.0'
}
```

Cấu hình DataSource

- Có thể cấu hình thông qua XML hoặc Annotation:

```
@Bean
public DataSource dataSource(){
   DriverManagerDataSource dataSource = new
   DriverManagerDataSource();
   dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
   dataSource.setUrl("jdbc:mysql://localhost:3306/phone_store");
   dataSource.setUsername( "root" );
   dataSource.setPassword( "123456" );
   return dataSource;
}
```

Cấu hình Entity Manager

```
@Bean
@Qualifier(value = "entityManager")
public EntityManager entityManager(EntityManagerFactory entityManagerFactory) {
  return entityManagerFactory.createEntityManager();
}

@Bean
public LocalContainerEntityManagerFactoryBean entityManagerFactory()
  { LocalContainerEntityManagerFactoryBean em
       = new LocalContainerEntityManagerFactoryBean();
  em.setDataSource(dataSource());
  em.setPackagesToScan(new String[] { "com.codegym.phonestore.model" });

  JpaVendorAdapter vendorAdapter = new
  HibernateJpaVendorAdapter();
  em.setJpaVendorAdapter(vendorAdapter);
  em.setJpaProperties(additionalProperties());
  return em;
}
```

Cấu hình Model

```
@Entity
@Table(name =
"phones") public
class Phone {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Long id;

   @Column(nullable =
   false) private String
   name;

   @ManyToOne(targetEntity = Manufacture.class, cascade = {CascadeType.PERSIST,
   CascadeType.REMOVE})
   private Manufacture manufacture;

   //Constructors
   //Getters/Setters
}
```

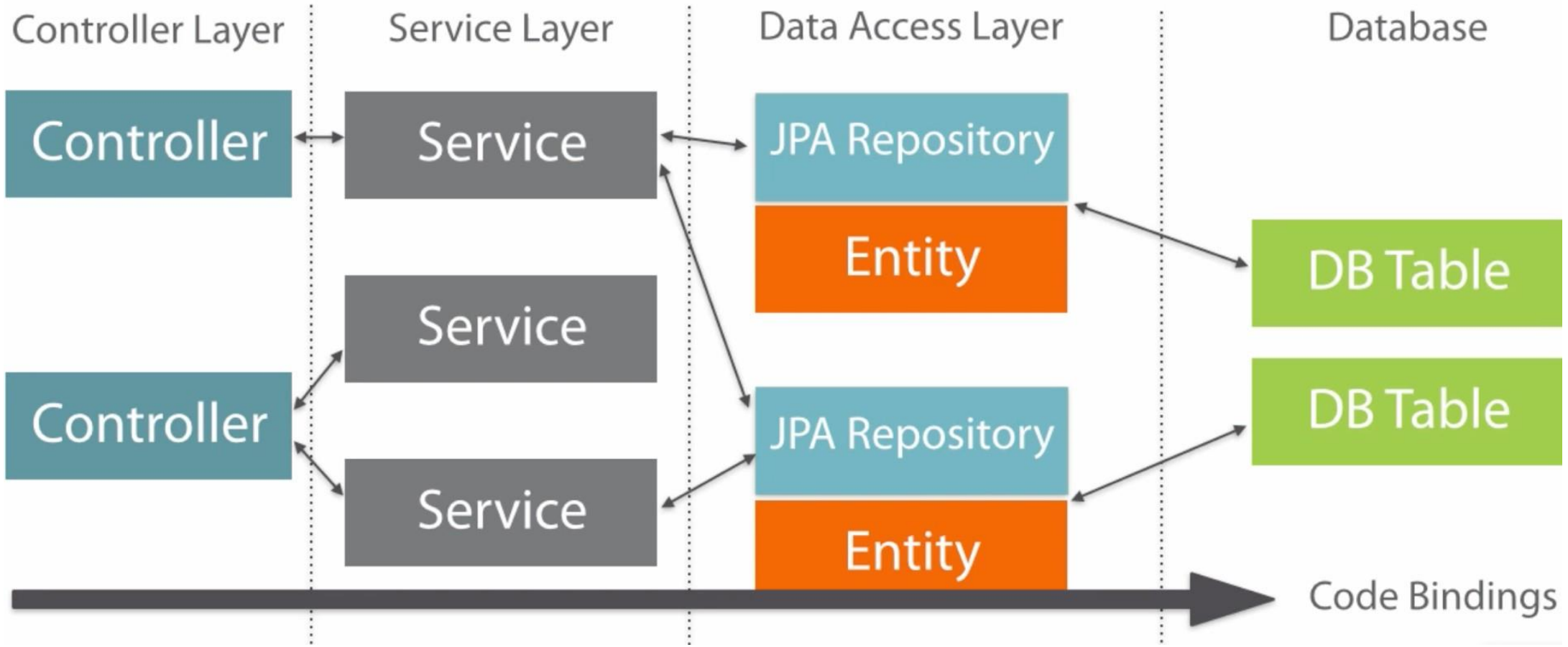
Truy xuất dữ liệu qua Entity Manager

```
@Transactional
public class CountryRepositoryImpl implements CountryRepository {

   @PersistenceContext
   EntityManager em;

   @Override
   public List<Country> findAll() {
     TypedQuery<Country> query = em.createQuery("select c from Country c",
     Country.class);
     return query.getResultList();
   }

   @Override
   public Country findById(Long id) {
     return em.find(Country.class, id);
   }
  .......
}
```

Các thao tác cơ bản

```
public interface Repository
   <T> { List<T> findAll();

   T findById(Long

   id); T save(T

   model);

   void remove(Long id);
}
```

Kiến trúc Repository



Demo

Spring Data JPA

Tổng kết

- JPA là đặc tả của Java dành cho các thao tác với dữ liệu
- Entity là các đối tượng đại diện cho dữ liệu
- Entity Manager là đối tượng quản lý các entity
- Spring Data JPA là framework của Spring hỗ trợ JPA, giúp cho việc triển khai JPA trở nên thuận tiện hơn

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: Spring Data Repository