Introduction to Spring Data JPA

Setting Up Your Spring Boot Project ---

Gradle:

```
dependencies {
   implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
   implementation 'org.springframework.boot:spring-boot-starter-web'
   runtimeOnly 'com.h2database:h2'
}
```

Defining the Entity Class ---

```
import jakarta.persistence.*;

@Entity
public class User {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private String firstName;
   private String lastName;
   private String email;

   // constructors
   // Getters and Setters
}
```

Creating the Repository Interface ---

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {}
```

Custom Queries in Spring Data JPA ---

In addition to the built-in methods, you can define custom queries using method names, JPQL, or native SQL.

Method Name Queries --
Spring Data JPA can derive queries from method names:

```
import java.util.List;

public interface UserRepository extends JpaRepository<User, Long> {
   List<User> findByLastName(String lastName);
}
```

JPQL Queries --
Use the @Query annotation for JPQL queries:

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface UserRepository extends JpaRepository<User, Long> {
   @Query("SELECT u FROM User u WHERE u.email = :email")
   User findByEmail(@Param("email") String email);
}
```

Native Queries --
For complex queries, you might use native SQL:

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface UserRepository extends JpaRepository<User, Long> {
   @Query(value = "SELECT * FROM User u WHERE u.email = :email", nativeQuery = true)
   User findByEmailNative(@Param("email") String email);
}
```

Pagination and Sorting --
Spring Data JPA supports pagination and sorting out of the box. Use the Pageable and Sort parameters in repository methods.

Pagination Example

```
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

public interface UserRepository extends JpaRepository<User, Long> {
   Page<User> findByLastName(String lastName, Pageable pageable);
}
```

Sorting Example

```
import java.util.List;
import org.springframework.data.domain.Sort;

public interface UserRepository extends JpaRepository<User, Long> {
   List<User> findByFirstName(String firstName, Sort sort);
}
```

Link ---
https://dev.to/nikhilxd/creating-spring-data-repositories-for-jpa-a-detailed-guide-23p5