



Presentation by Tom Naujox

Spring Boot

Spring Framework

Spring Data



Spring Data



Spring Cloud



Spring Cloud Data Flow

Spring Security



Spring Session



Spring Integration

Spring HATEOAS

Spring REST Docs

Spring Batch

Spring AMQP

Spring CredHub

Spring Flo

Spring for Apache Kafka

Spring LDAP

Spring Roo

Spring Shell

Spring Statemachine

Spring Vault

Spring Web Flow

Spring Data JDBC

Spring Data JPA

Spring Data LDAP

Spring Data MongoDB

Spring Data Redis

Spring Data R2DBC

Spring Data REST

Spring Data for Apache
Cassandra

Spring Data for Apache
Geode

Spring Data for Apache Solr

Spring Data for Pivotal
GemFire

Spring Data Couchbase

Spring Data Elasticsearch

Spring Data Envers

Spring Data Neo4j

Spring Data JDBC
Extensions

Spring for Apache Hadoop

SPRING FRAMEWORK

Most popular Java Framework

- aspect oriented programming
- application context
- dependency injection
- Multiple Useful extensions

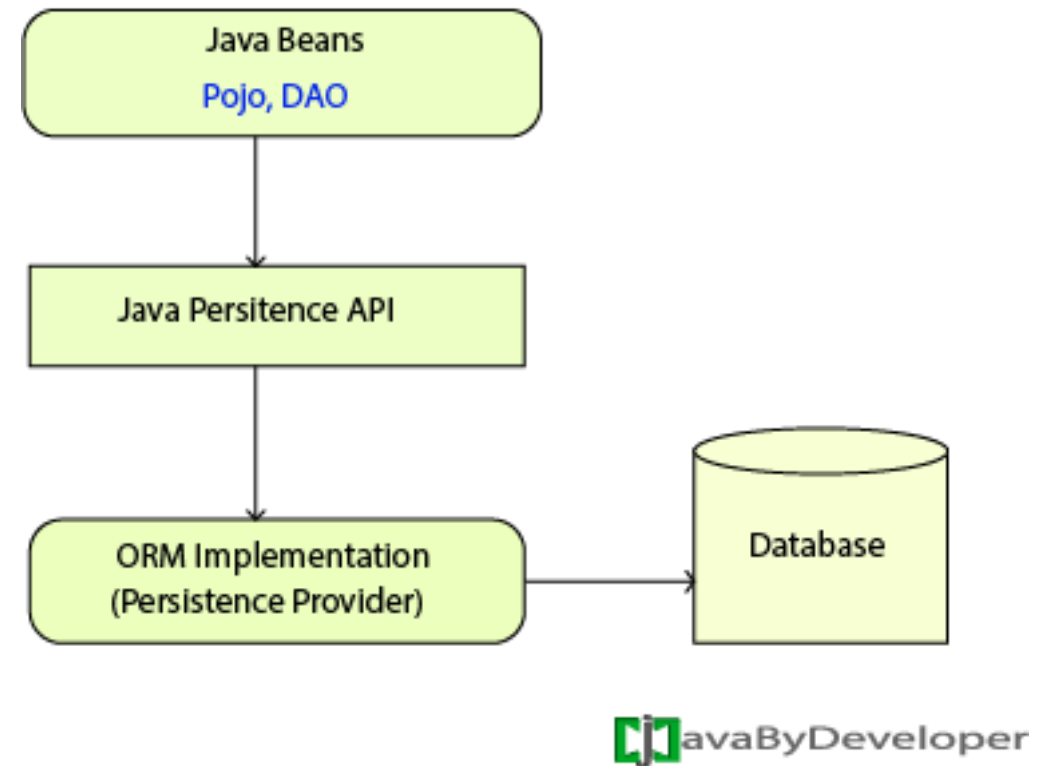
JPA

Jakarta Persistence API

Specification for accessing, persisting and managing data between Java Objects and Relational Database

Needs ORM implementation

```
import javax.persistence.*;
```



IMPLEMENTING SPRING DATA JPA

Dependencies & Database
implementation

Maven™

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
```



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

WORKING WITH SPRING DATA JPA

Entities

- annotate class with @Entity
- annotate one attribute with @Id

```
import javax.persistence.*;

@Entity
public class item {
    @Id
    private Long itemNr;
}
```

additional specifications:

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

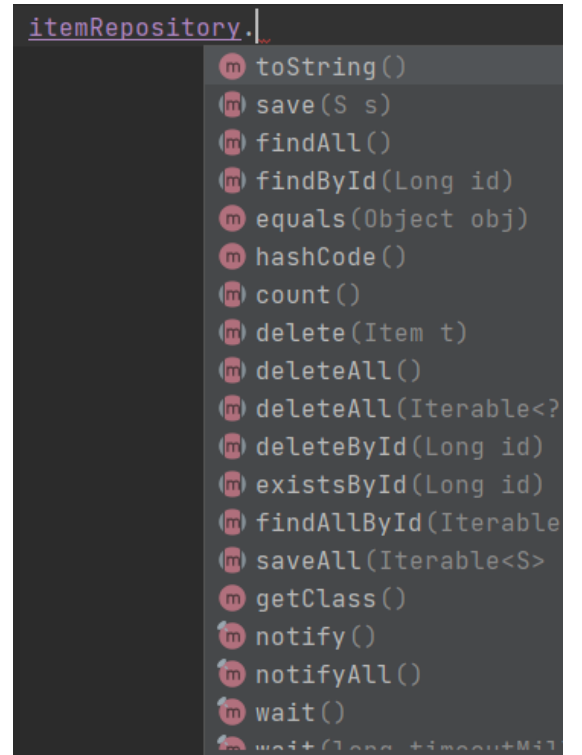
    @NotNull
    @ManyToOne
    @JoinColumn(name = "COURSE_ID")
    private Course course;

    ...
}
```

WORKING WITH SPRING DATA JPA

Repositories

```
import org.springframework.data.repository.CrudRepository;  
  
public interface ItemRepository extends CrudRepository<Item, Long> {  
}
```

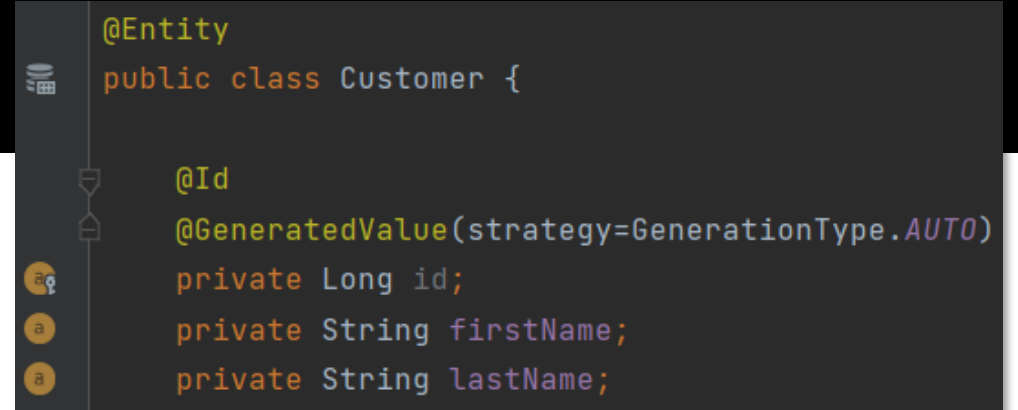


The screenshot shows an IDE with the text `itemRepository.` followed by a dropdown menu of methods. The methods listed are:

- `toString()`
- `save(S s)`
- `findAll()`
- `findById(Long id)`
- `equals(Object obj)`
- `hashCode()`
- `count()`
- `delete(Item t)`
- `deleteAll()`
- `deleteAll(Iterable<?>`
- `deleteById(Long id)`
- `existsById(Long id)`
- `findAllById(Iterable`
- `saveAll(Iterable<S>`
- `getClass()`
- `notify()`
- `notifyAll()`
- `wait()`
- `wait(long timeout+Mil`

WORKING WITH SPRING DATA JPA

Repositories

A code snippet from an IDE showing the definition of a Customer entity. It includes the @Entity annotation, a public class Customer, and three private fields: id (Long, annotated with @Id and @GeneratedValue), firstName (String), and lastName (String).

```
@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String firstName;
    private String lastName;
}
```

```
public interface CustomerRepository extends CrudRepository<Customer, Long> {
    List<Customer> findByLastName (String lastName);
    Customer findById(long id);
}
```

PRACTICAL TASK

Project generator: <https://start.spring.io/>