Spring REST
Spring Data JPA
Hibernate

Contents

- Spring REST API
- Spring Data JPA
- Hibernate Seeding
- Hibernate Relationships

Spring REST API Annotations

@Controller

@RestController

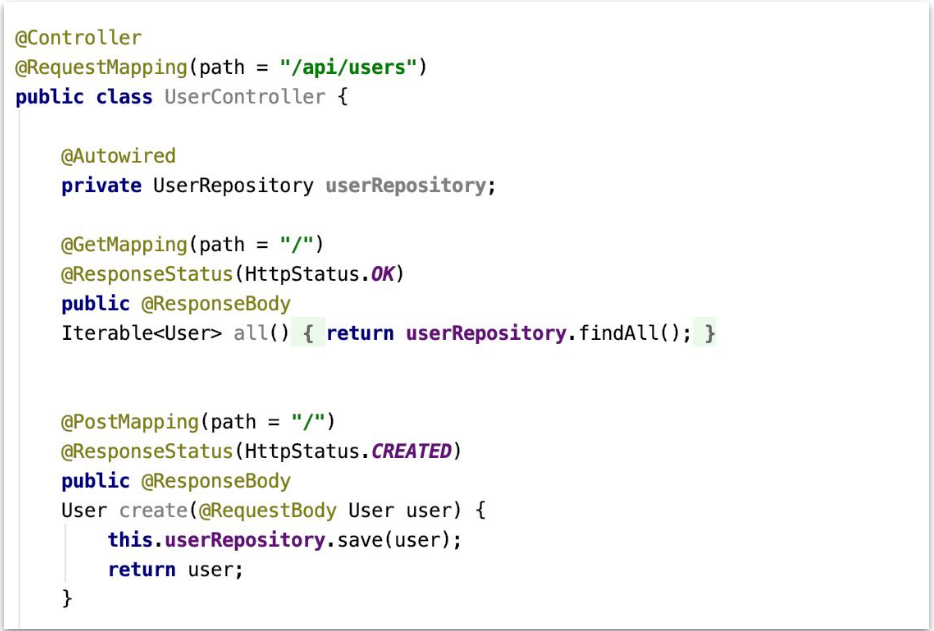
@ResponseStatus

@RequestBody

@ResponseBody

@PathVariable

@Controller, @ResponseBody, @RequestBody



```
@Controller
@RequestMapping(path = "/api/users")
public class UserController {

    @Autowired
    private UserRepository userRepository;

    @GetMapping(path = "/")
    @ResponseStatus(HttpStatus.OK)
    public @ResponseBody
    Iterable<User> all() { return userRepository.findAll(); }


    @PostMapping(path = "/")
    @ResponseStatus(HttpStatus.CREATED)
    public @ResponseBody
    User create(@RequestBody User user) {
        this.userRepository.save(user);
        return user;
    }
```

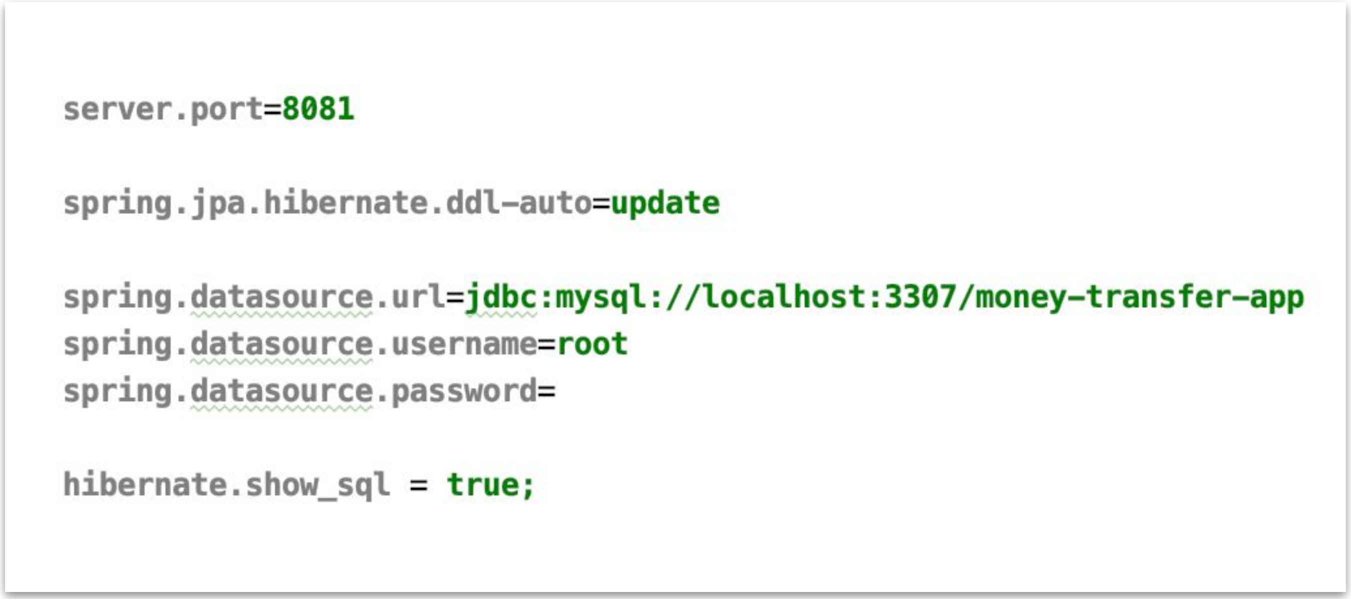
Spring Data JPA

- The Java Persistence API (JPA) is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database.
- JPA itself is just a specification, not a product; it cannot perform persistence or anything else by itself. JPA is just a set of interfaces, and requires an implementation.

Spring Data JPA Installation

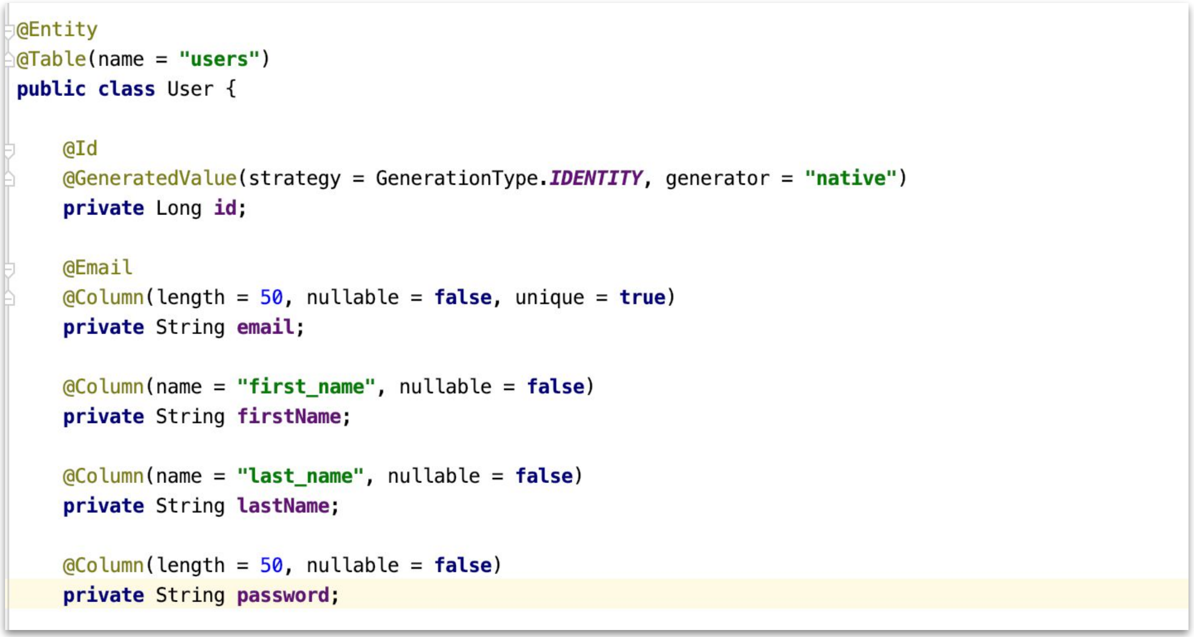
```xml
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
```

Spring Data Source Configuration



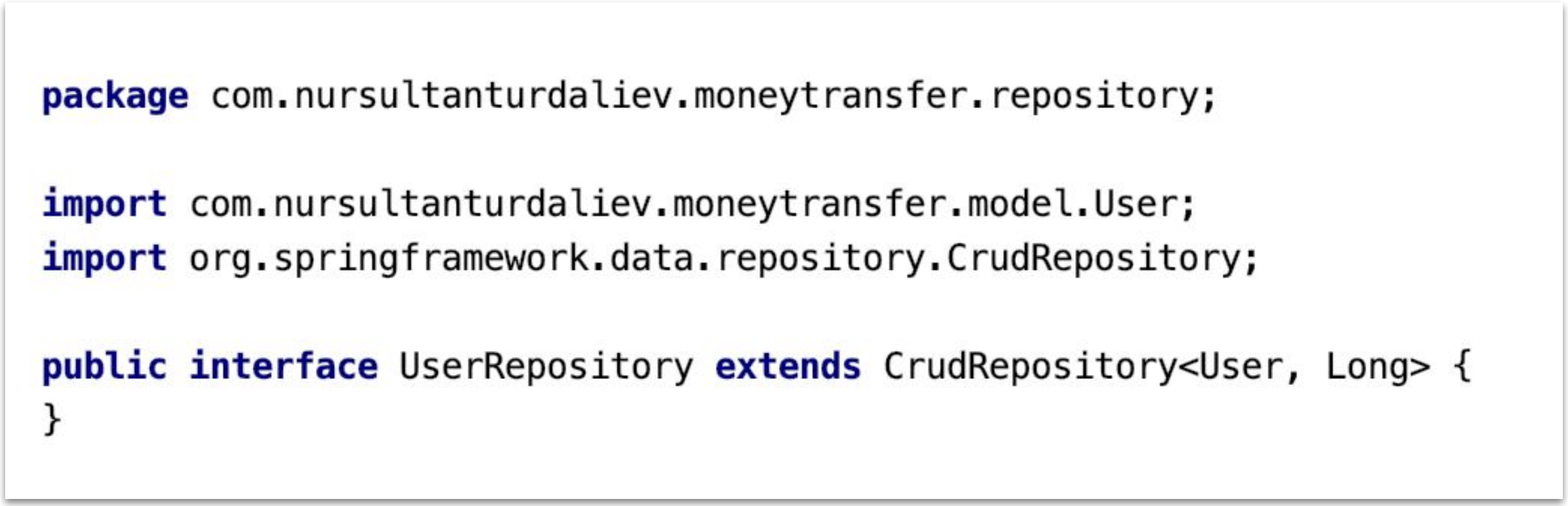
```
server.port=8081

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3307/money-transfer-app
spring.datasource.username=root
spring.datasource.password=

hibernate.show_sql = true;
```

Spring Data JPA In Action



```java
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY, generator = "native")
    private Long id;

    @Email
    @Column(length = 50, nullable = false, unique = true)
    private String email;

    @Column(name = "first_name", nullable = false)
    private String firstName;

    @Column(name = "last_name", nullable = false)
    private String lastName;

    @Column(length = 50, nullable = false)
    private String password;
```

Spring Data JPA Repository



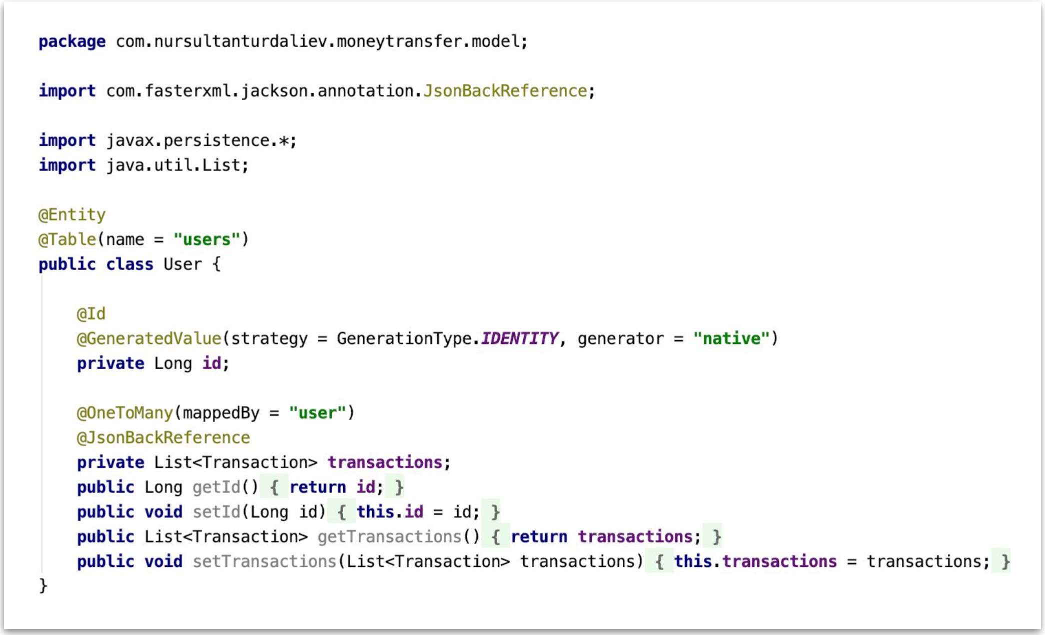
```java
package com.nursultanturdaliev.moneytransfer.repository;

import com.nursultanturdaliev.moneytransfer.model.User;
import org.springframework.data.repository.CrudRepository;

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

Hibernate Seeding

1. Create `resources/import.sql`
2. Restart Application
3. Hibernate automatically executes `import.sql`

Hibernate @OneToMany, @ManyToOne, @OneToOne, @ManyToMany Relationships

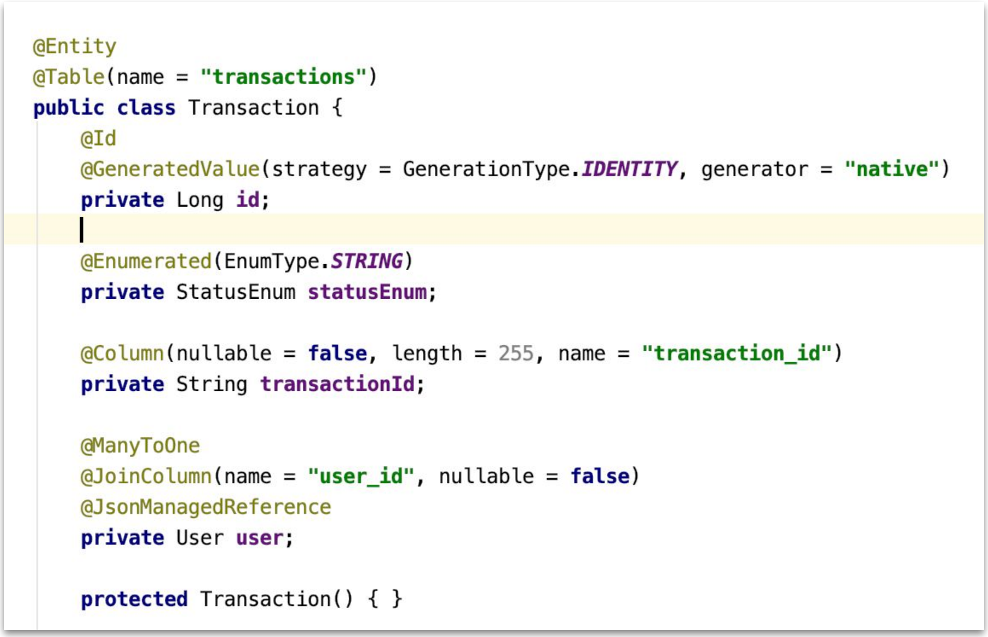
@OneToMany



```java
package com.nursultanturdaliev.moneytransfer.model;

import com.fasterxml.jackson.annotation.JsonBackReference;

import javax.persistence.*;
import java.util.List;

@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY, generator = "native")
    private Long id;

    @OneToMany(mappedBy = "user")
    @JsonBackReference
    private List<Transaction> transactions;
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public List<Transaction> getTransactions() { return transactions; }
    public void setTransactions(List<Transaction> transactions) { this.transactions = transactions; }
}
```

The ***mappedBy*** property is what we use to tell Hibernate which variable we are using to represent the parent class in our child class.

@ManyToOne



```
@Entity
@Table(name = "transactions")
public class Transaction {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY, generator = "native")
    private Long id;
    
    @Enumerated(EnumType.STRING)
    private StatusEnum statusEnum;

    @Column(nullable = false, length = 255, name = "transaction_id")
    private String transactionId;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    @JsonManagedReference
    private User user;

    protected Transaction() { }
```

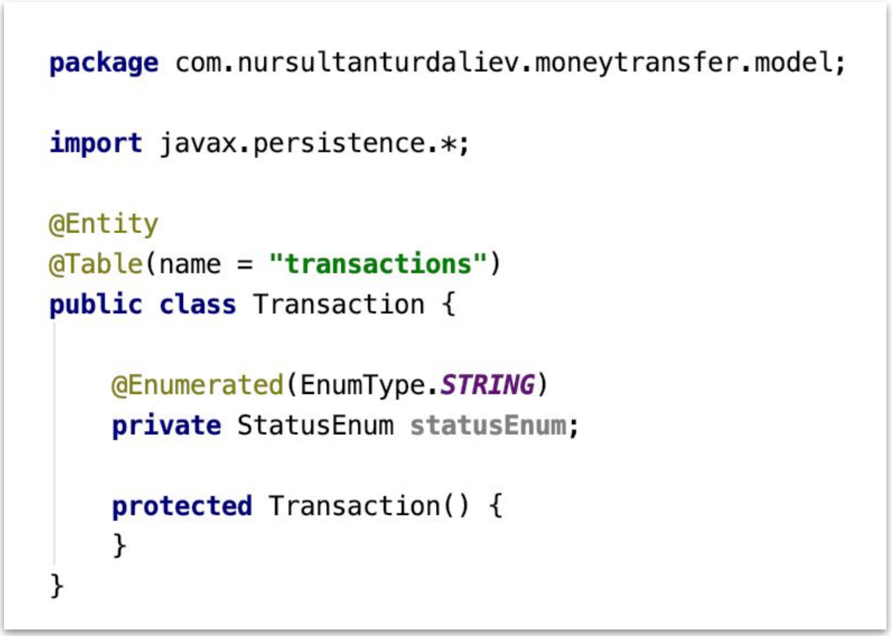
A ***many-to-one*** mapping means that many instances of this entity are mapped to one instance of another entity

@ManyToMany

```
@Entity
public class Vehicle {
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
private int id;
private String name;
@ManyToMany(mappedBy="vehicle")
private Collection<User> user=new ArrayList<>();;
```

```
@Entity
@Table(name="user_details")
public class User {
@GeneratedValue(strategy=GenerationType.AUTO)
@Id
private int id;
private String userName;
@ManyToMany(cascade=CascadeType.ALL)
@JoinTable(name="usr_vehicle",joinColumns=@JoinColumn(name="u
ser_id"),inverseJoinColumns=@JoinColumn(name="vehicle_id") )
private Collection<Vehicle> vehicle=new ArrayList<>();
```

Enum Types



```java
package com.nursultanturdaliev.moneytransfer.model;

import javax.persistence.*;

@Entity
@Table(name = "transactions")
public class Transaction {

    @Enumerated(EnumType.STRING)
    private StatusEnum statusEnum;

    protected Transaction() {
    }
}
```

Spring Data JPA Indexing

```
@Entity

@Table(name = "addresses", indexes = {@Index(name = "addresses_index_city", columnList = "city",
unique = false)})

public class Address {

}
```

@JsonIgnore, @JsonManagedReference, @JsonBackReference, @JsonIgnoreProperties

```
@JsonBackReference
private Collection<Transaction> transactions
```

```
@JsonIgnoreProperties(value = { "intValue" })
```

```
@JsonManagedReference
private User user
```

```
@JsonIgnore
private String firstName
```

Exercises

- User Endpoints
    - Create, Read, Update, Delete
- Transaction Endpoints
    - Create, Read, Read by User, Read All
- Add index to user email, firstname, last name, email
- Create Table with ManyToMany, OneToOne, ManyToOne,

References

- https://www.thymeleaf.org/
- https://dzone.com/articles/spring-boot-with-spring-data-jpa
- https://www.baeldung.com/jpa-entities
- https://www.baeldung.com/hibernate-one-to-many
- https://dzone.com/articles/hibernate-mapping
- https://www.baeldung.com/jackson-ignore-properties-on-serialization