Hibernate & Spring Data JPA

Beginner to Guru

JPA Inheritance

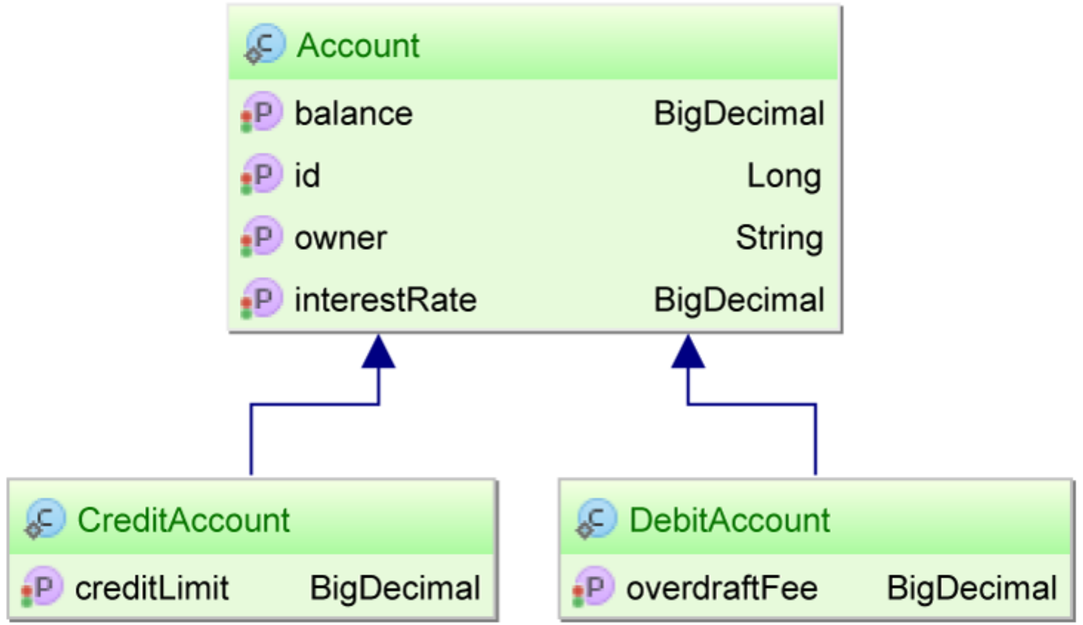
JPA Inheritance

- Inheritance is a fundamental concept of Object Oriented Programming
  - A class can inherit properties and behaviors from other classes
- The relational model does not exactly support inheritance
- In a JPA context with inheritance, you are concerned with the persistence of properties

JPA Inheritance

- **Mapped Super Class** - Top level entities inherit properties and have one table per top level entity
- **Single Table** - Object hierarchy is mapped to a single table.
  - Default when using `@Inheritance` mapping annotation
- **Joined Table** - Base Classes and subclasses have their own table, fetching subclass requires a join
- **Table Per Class** - Each object is mapped to a table, joins are used to create top level entities

Mapped Superclass



SPRING FRAMEWORK GURU 2021

Source: Hibernate Documentation

Mapped Superclass

```
@MappedSuperclass
public static class Account {

        @Id
        private Long id;

        private String owner;

        private BigDecimal balance;

        private BigDecimal interestRate;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Mapped Superclass

```java
@Entity(name = "DebitAccount")
public static class DebitAccount extends Account {

        private BigDecimal overdraftFee;

        //Getters and setters are omitted for brevity

}
```

```java
@Entity(name = "CreditAccount")
public static class CreditAccount extends Account {

        private BigDecimal creditLimit;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Mapped Superclass

```
CREATE TABLE DebitAccount (
    id BIGINT NOT NULL ,
    balance NUMERIC(19, 2) ,
    interestRate NUMERIC(19, 2) ,
    owner VARCHAR(255) ,
    overdraftFee NUMERIC(19, 2) ,
    PRIMARY KEY ( id )
)
```

```
CREATE TABLE CreditAccount (
    id BIGINT NOT NULL ,
    balance NUMERIC(19, 2) ,
    interestRate NUMERIC(19, 2) ,
    owner VARCHAR(255) ,
    creditLimit NUMERIC(19, 2) ,
    PRIMARY KEY ( id )
)
```

Source: Hibernate Documentation

Single Table

```
@Entity(name = "Account")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public static class Account {

        @Id
        private Long id;

        private String owner;

        private BigDecimal balance;

        private BigDecimal interestRate;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Single Table

```java
@Entity(name = "DebitAccount")
public static class DebitAccount extends Account {

        private BigDecimal overdraftFee;

        //Getters and setters are omitted for brevity

}
```

```java
@Entity(name = "CreditAccount")
public static class CreditAccount extends Account {

        private BigDecimal creditLimit;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Single Table

```
CREATE TABLE Account (
    DTYPE VARCHAR(31) NOT NULL ,
    id BIGINT NOT NULL ,
    balance NUMERIC(19, 2) ,
    interestRate NUMERIC(19, 2) ,
    owner VARCHAR(255) ,
    overdraftFee NUMERIC(19, 2) ,
    creditLimit NUMERIC(19, 2) ,
    PRIMARY KEY ( id )
)
```

Source: Hibernate Documentation

Single Table

```
INSERT INTO Account (balance, interestRate, owner, overdraftFee, DTYPE, id)
VALUES (100, 1.5, 'John Doe', 25, 'DebitAccount', 1)

INSERT INTO Account (balance, interestRate, owner, creditLimit, DTYPE, id)
VALUES (1000, 1.9, 'John Doe', 5000, 'CreditAccount', 2)
```

Source: Hibernate Documentation

Joined Table

```
@Entity(name = "Account")
@Inheritance(strategy = InheritanceType.JOINED)
public static class Account {

        @Id
        private Long id;

        private String owner;

        private BigDecimal balance;

        private BigDecimal interestRate;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Joined Table

```java
@Entity(name = "DebitAccount")
public static class DebitAccount extends Account {

        private BigDecimal overdraftFee;

        //Getters and setters are omitted for brevity

}
```

```java
@Entity(name = "CreditAccount")
public static class CreditAccount extends Account {

        private BigDecimal creditLimit;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Joined Table

```
CREATE TABLE Account (
    id BIGINT NOT NULL ,
    balance NUMERIC(19, 2) ,
    interestRate NUMERIC(19, 2) ,
    owner VARCHAR(255) ,
    PRIMARY KEY ( id )
)
```

```
CREATE TABLE CreditAccount (
    creditLimit NUMERIC(19, 2) ,
    id BIGINT NOT NULL ,
    PRIMARY KEY ( id )
)

CREATE TABLE DebitAccount (
    overdraftFee NUMERIC(19, 2) ,
    id BIGINT NOT NULL ,
    PRIMARY KEY ( id )
)
```

Source: Hibernate Documentation

Table Per Class

```
@Entity(name = "Account")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public static class Account {

        @Id
        private Long id;

        private String owner;

        private BigDecimal balance;

        private BigDecimal interestRate;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Table Per Class

```java
@Entity(name = "DebitAccount")
public static class DebitAccount extends Account {

        private BigDecimal overdraftFee;

        //Getters and setters are omitted for brevity

}
```

```java
@Entity(name = "CreditAccount")
public static class CreditAccount extends Account {

        private BigDecimal creditLimit;

        //Getters and setters are omitted for brevity

}
```

Source: Hibernate Documentation

Table Per Class

```
CREATE TABLE Account (
    id BIGINT NOT NULL ,
    balance NUMERIC(19, 2) ,
    interestRate NUMERIC(19, 2) ,
    owner VARCHAR(255) ,
    PRIMARY KEY ( id )
)
```

```
CREATE TABLE CreditAccount (
    id BIGINT NOT NULL ,
    balance NUMERIC(19, 2) ,
    interestRate NUMERIC(19, 2) ,
    owner VARCHAR(255) ,
    creditLimit NUMERIC(19, 2) ,
    PRIMARY KEY ( id )
)
```

```
CREATE TABLE DebitAccount (
    id BIGINT NOT NULL ,
    balance NUMERIC(19, 2) ,
    interestRate NUMERIC(19, 2) ,
    owner VARCHAR(255) ,
    overdraftFee NUMERIC(19, 2) ,
    PRIMARY KEY ( id )
)
```

Source: Hibernate Documentation

JPA Inheritance Downsides

- **Leakage** - A term to describe when the concepts of one paradigm 'leak' into another
  - The Primary Key / Id property is an example of the relational model 'leaking' into the object model
- **Mapped Superclass** - Does not allow for polymorphic queries
- **Single Table** - Requires discriminator column and columns for all properties of child classes
- **Joined Table** - Requires SQL Joins and object id in child object tables
- **Table Per Class** - Requires complex union joins to support polymorphic queries

Which to Use?

- **Leakage** is unavoidable
- Single Table, Joined Table, and Table Per Class 'Leak' into the relational model and can be complex for non-Hibernate clients
- **K.I.S.S.** - **K**eep **I**t **S**imple **S**tupid
- Mapped Superclass is the simplest and addresses probably 85% of use cases
- Be aware of the other options, and use when you feel its appropriate

SPRING FRAMEWORK

GURU