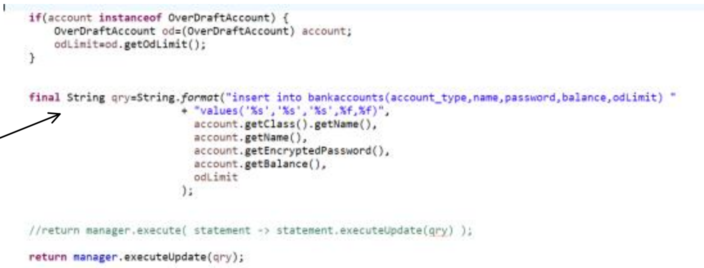
Spring JPA

Thursday, November 5, 2020 5:31 PM

Evolution of Database Access

Generation 1 — Plain JDBC Calls

- User Makes JDBC calls using Jdbc and Drivers
  - Connection
  - Statement
  - ResultSet
- Developer need to generate queries based on User defined Objects
- Developer need to get the value from Resultset and created Object from those values
  - You need to match column to field
  - You need to convert data type
  - You need to plan for Different type of Object

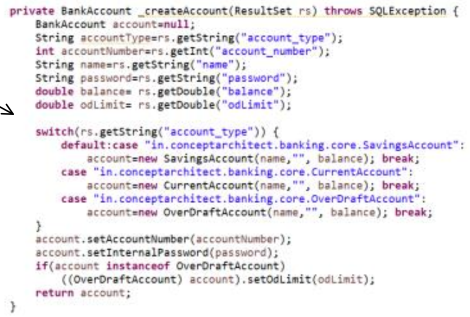


```
if(account instanceof OverDraftAccount) {
    OverDraftAccount od=(OverDraftAccount) account;
    odLimit=od.getOdLimit();
}

final String qry=String.format("insert into bankaccounts(account_type,name,password,balance,odLimit) "
                        + "values('%s','%s','%s',%f,%f)",
                          account.getClass().getName(),
                          account.getName(),
                          account.getEncryptedPassword(),
                          account.getBalance(),
                          odLimit
                        );

//return manager.execute( statement -> statement.executeUpdate(qry) );

return manager.executeUpdate(qry);
```



```
private BankAccount _createAccount(ResultSet rs) throws SQLException {
    BankAccount account=null;
    String accountType=rs.getString("account_type");
    int accountNumber=rs.getInt("account_number");
    String name=rs.getString("name");
    String password=rs.getString("password");
    double balance= rs.getDouble("balance");
    double odLimit= rs.getDouble("odLimit");

    switch(rs.getString("account_type")) {
        default:case "in.conceptarchitect.banking.core.SavingsAccount":
            account=new SavingsAccount(name,"", balance); break;
        case "in.conceptarchitect.banking.core.CurrentAccount":
            account=new CurrentAccount(name,"", balance); break;
        case "in.conceptarchitect.banking.core.OverDraftAccount":
            account=new OverDraftAccount(name,"", balance); break;
    }
    account.setAccountNumber(accountNumber);
    account.setInternalPassword(password);
    if(account instanceof OverDraftAccount)
        ((OverDraftAccount) account).setOdLimit(odLimit);
    return account;
}
```

Can These two steps be Automated?

- Can we automatically generate Query by looking into Object?
- Can we automatically create object and get the values from Result Set?

Solution

- We can do it by using Reflection
- Generating the Query
  - We can find all the fields of a class
  - We can generate a insert or update query based on the field names
  - This is assuming that the field name and the column name are exactly same
    - What if they are not same
- Fetching the records from ResultSet
  - We can find all fields of a class
  - We can search for a matching column in Resultset
  - We can get the data from the those columns.

Generation 2 — ORM and Hibernate

- ORM is a broad conception to represent Object-Relationship-Mapping
- We need a way to simplify database interaction by providing functionalities like
  - Auto generate tables
  - Auto insert/update/delete Objects into the table
  - Automatically fetch Object from the table
- But world of Object Oriented Programming is different from the world of database

Features	OO Programming	Database	Remark
Fundamental Elements	Object and Class	Table	
Data is stored as	Objects and properties	Rows and colums	
Object Identity	hashCode Not a field of the class	Primary key	

Relationship	Author has many books
	class Author{
	String name
	List<Book> books;
	}
	class Book{
	String title;
	Author author;
	}
		Tables have relationship
		table Authors
		ID PRIMARY KEY
		NAME VARCHAR(255)
		Table Books
		ID PK
		TITLE VARCHAR(255)
		AUTHORID FK
			• No ID required in Author Class
			• No way to represent List of books in BookTable
			○ Represented by foregin key in Book table
			• No primary key or foreign key required in java class

Class Hierarcy	• Inhertiance • Interface	Not supported	

- More features
  - Performance Optimization
  - Automatic change detection
  - Caching of records already pulled from the database

Hibernate

- Hiberante is one of the most popular ORM framework available for Java Applications
- It is developed as an open source by third party company — not java not spring
- Provides extensive support for Object Oriented interaction with database
  - You can Map Java class to Database
  - Automatically generates Query
  - Provides a special Object Oriented Query **HQL (Hiberanate Query Language)**
    - Provides query syntax based on Java classes and other features
  - Support for 1:1, 1:M, M:M relationship
  - Support for mapping inheritance
  - Support for different databases
  - Facility to cache the already pulled records
  - Automatic change detection in object
- As a developer you don't execute insert query
  - You save an object
- You query for objects and get a list of Object

Other ORMs

- There are several other lesser popular ORMs available

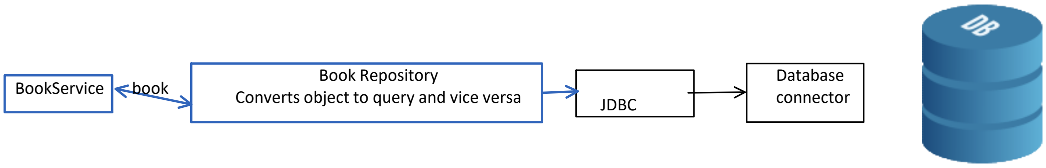
Generation 3 — JPA

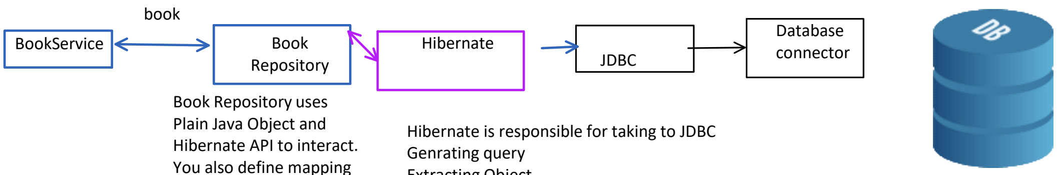
- Java created a standard abstraction on the top of ORM frameworks
- Java standard is known as **Java Persistence API (JPA)**
- You can think of JPA as an standardization of ORM.
- Inspired greatly by Hibhernate
- Provides its own query syntax call **JPQL —> Java Persistence Query Language!**
- Provides standard set of @Annotations and interface to describe how you should interact with any JPA framework
- Hibernate supports JPA
- Hibernate can be viewed as JPA+ framework
  - It supports everything JPA + Additional features

Generation 4 — Spring JPA

- It is an additional layer on the the top of JPA.
- It tries to make the JPA conifuguration and use lot simpler by defining its own set of classes and auto generators
- You often need to write no code to get the data in and out of database

- Note
  - Spring JPA doesn't aim at replacing hibernate
  - In fact it will use along with hibernate
  - Actual JPA interaction and heavy weight lifting is done by Hibernate or other JPA providers
  - Spring simply provides its own functionality to consume JPA and provide easy access to them



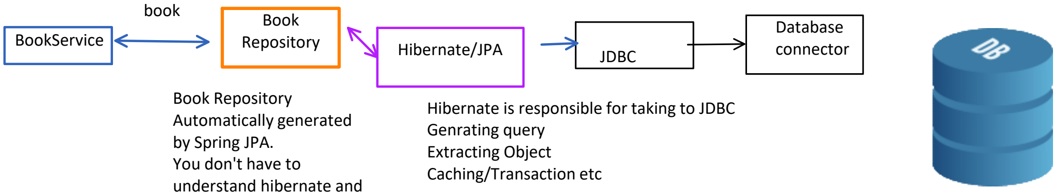


Book Repository uses
Plain Java Object and
Hibernate API to interact.
You also define mapping
and configuration

Hibernate is responsible for taking to JDBC
Genrating query
Extracting Object
Caching/Transaction etc

You also define mapping
and configuration

Extracting Object
Caching/Transaction etc



Book Repository
Automatically generated by Spring JPA.
You don't have to understand hibernate and very little JPA to make the call. You just define the interface suggesting what data you need. Spring will automatically implement the required logic.

Hibernate is responsible for taking to JDBC
Genrating query
Extracting Object
Caching/Transaction etc

Spring JPA Requirement

```
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-jpa</artifactId>
    <version>2.3.5.RELEASE</version>
</dependency>
```