ORM / Spring Data JPA

Silas Berger (Claudio Corrodi)

ESE 2017

Traditional DB connections

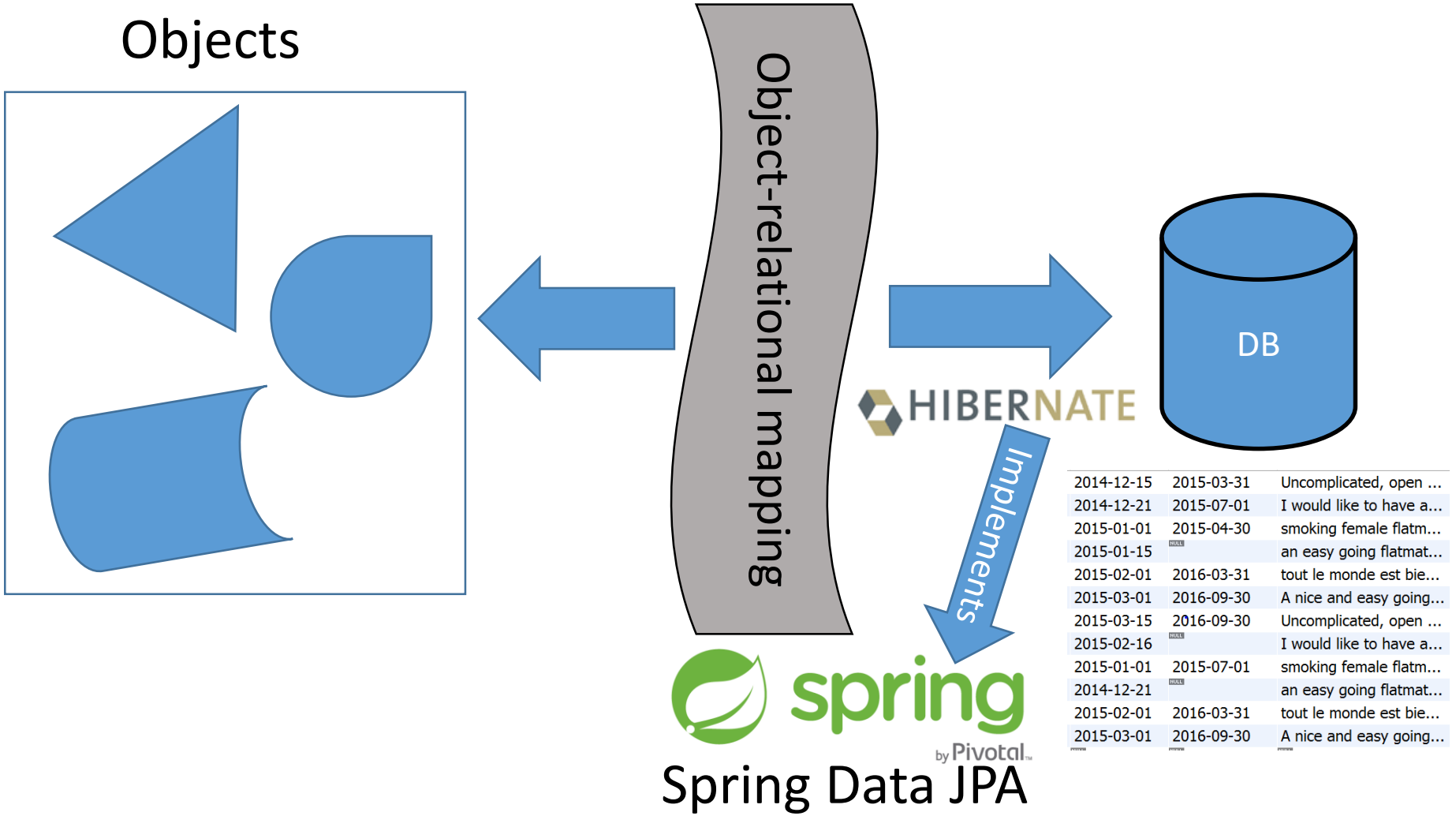
```
import java.sql.*;

public class JDBCDemo {
    public static void main(String[] args) {

        Connection connection;
        PreparedStatement statement;
        ResultSet resultSet;

        try {
            connection = DriverManager.getConnection("jdbc:mysql://localhost/team1?" +
                    "user=root&password=");
            statement = connection.prepareStatement(
                  "select city from ad where prizePerMonth > ?;");
            statement.setInt(1, 750); // parameter 1 has value 750
            resultSet = statement.executeQuery();
            while (resultSet.next()) {
                System.out.printf("City: %s\n", resultSet.getString(1));
            }
        } catch (SQLException e) {
            // handle exception
        }
    }
}
```

ORM: Object-relational mapping



Spring database connection

src/main/webapp/WEB-INF/configspringData.xml

```
<bean id="mainDataSource" class="com.jolbox.bonecp.BoneCPDataSource" destroy-method="close">
    <property name="driverClass" value="com.mysql.jdbc.Driver" />
    <property name="jdbcUrl" value="jdbc:mysql://localhost/team1?autoReconnect=true&amp;
          createDatabaseIfNotExist=true&amp;useUnicode=true&amp;characterEncoding=utf-8" />
    <property name="username" value="root"/>
    <property name="password" value=""/>
    <property name="idleConnectionTestPeriodInMinutes" value="60"/>
    <property name="idleMaxAgeInMinutes" value="240"/>
    <property name="maxConnectionsPerPartition" value="30"/>
    <property name="minConnectionsPerPartition" value="10"/>
    <property name="partitionCount" value="3"/>
    <property name="acquireIncrement" value="5"/>
    <property name="statementsCacheSize" value="100"/>
    <property name="releaseHelperThreads" value="3"/>
</bean>
```

Spring data mapping

src/main/java/ch/unibe/ese/team1/model/Ad.java

Business logic

```java
/** Describes an advertisement that users can place and search for.
*/
@Entity
public class Ad {

   @Id
   @GeneratedValue
   private long id;

   @Column(nullable = false)
   private String title;

   @Column(nullable = false)
   private String street;
   /* ... */
}
```

Spring data mapping

src/main/java/ch/unibe/ese/team1/model/Ad.java

Business logic

```java
/** Describes an advertisement that users can place and search for.
*/
@Entity
public class Ad {

   @Id
   @GeneratedValue
   private long id;

   @Column(nullable = false)
   private String title;

   @Column(nullable = false)
   private String street;
   /* ... */
}
```

Repository

```java
public interface AdDao extends CrudRepository<Ad, Long> {

   /** this will be used if both
         rooms AND studios are searched */
   public Iterable<Ad>
      findByPrizePerMonthLessThan (int prize);

   /** this will be used if only
         rooms or studios are searched */
   public Iterable<Ad>
      findByStudioAndPrizePerMonthLessThan(boolean studio,
      int i);

   public Iterable<Ad> findByUser(User user);
}
```

src/main/java/ch/unibe/ese/team1/model/dao/AdDao.java

Keyword queries

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is,Equals	findByFirstname,findByFirstnameIs	… where x.firstname = ?1
Between	findByStartDateBetween	… where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull	findByAgeIsNull	… where x.age is null

Saving an Ad to the DB

```
public void saveNewAdToDB(
           String title,
           String street,
           String city,
           double price,
           boolean isStudio){

    Connection connection;
    PreparedStatement statement;

    try{
        connection = DriverManager.getConnection("jdbc:mysql://localhost/project8joke"
                                                 + "?user=root&password=&serverTimezone=MET");
        statement = connection.prepareStatement("INSERT INTO ad "
                    + "(title, street, city, price, isStudio) VALUES (?, ?, ?, ?, ?);");
        statement.setString(1, title);
        statement.setString(2, street);
        statement.setString(3, city);
        statement.setDouble(4, price);
        statement.setBoolean(5, isStudio);
        statement.execute();
        connection.close();
    } catch(SQLException ex){
        ex.printStackTrace();
    }
}
```

Traditional Approach

Saving an Ad to the DB

ORM Approach

```
@Autowired
AdDao adDao;

public void saveNewAdToDB(
            String title,
            String street,
            String city,
            double price,
            boolean isStudio){
    adDao.save(new Ad(title, street, city, price, isStudio));
}
```

Traditional Approach

```
public void saveNewAdToDB(
            String title,
            String street,
            String city,
            double price,
            boolean isStudio){

    Connection connection;
    PreparedStatement statement;

    try{
        connection = DriverManager.getConnection("jdbc:mysql://localhost/project8joke"
                                                 + "?user=root&password=&serverTimezone=MET");
        statement = connection.prepareStatement("INSERT INTO ad "
                    + "(title, street, city, price, isStudio) VALUES (?, ?, ?, ?, ?);");
        statement.setString(1, title);
        statement.setString(2, street);
        statement.setString(3, city);
        statement.setDouble(4, price);
        statement.setBoolean(5, isStudio);
        statement.execute();
        connection.close();
    } catch(SQLException ex){
        ex.printStackTrace();
    }
}
```

Getting an Ad by ID from the DB

Traditional Approach

```
public Ad getAdByID(long id){
    Ad ad = null;

    Connection connection;
    PreparedStatement statement;
    ResultSet resultSet;

    try{
        connection = DriverManager.getConnection("jdbc:mysql://localhost/project8joke" +
                "?user=root&password=&serverTimezone=MET");
        statement = connection.prepareStatement("SELECT * FROM ad WHERE id=?");
        statement.setLong(1, id);
        resultSet = statement.executeQuery();
        resultSet.next();

        // fill result data into new Ad object
        ad = new Ad(
                resultSet.getString("title"),
                resultSet.getString("street"),
                resultSet.getString("city"),
                resultSet.getDouble("price"),
                resultSet.getBoolean("isStudio"));

        connection.close();
    } catch (SQLException ex){
        ex.printStackTrace();
    }
    return ad;
}
```

Getting an Ad by ID from the DB

```
public interface AdDao extends CrudRepository<Ad, Long> {

    public Ad findById(long id);

    …

}
```

Preparation

ORM approach

Query

```
@Autowired
AdDao adDao;

private Ad getAdById(long id){
    return adDao.findById(id);
}
```

Spring QueryDSL

Type-safe queries similar to SQL

```
Predicate predicate =
    user
    .firstname.equalsIgnoreCase("dave")
    .and(user.lastname.startsWithIgnoreCase("mathews"));

userRepository.findAll(predicate);
```

Learn more about QueryDSL here: https://spring.io/blog/2011/04/26/advanced-spring-data-jpa-specifications-and-querydsl/

More

@Query annotation

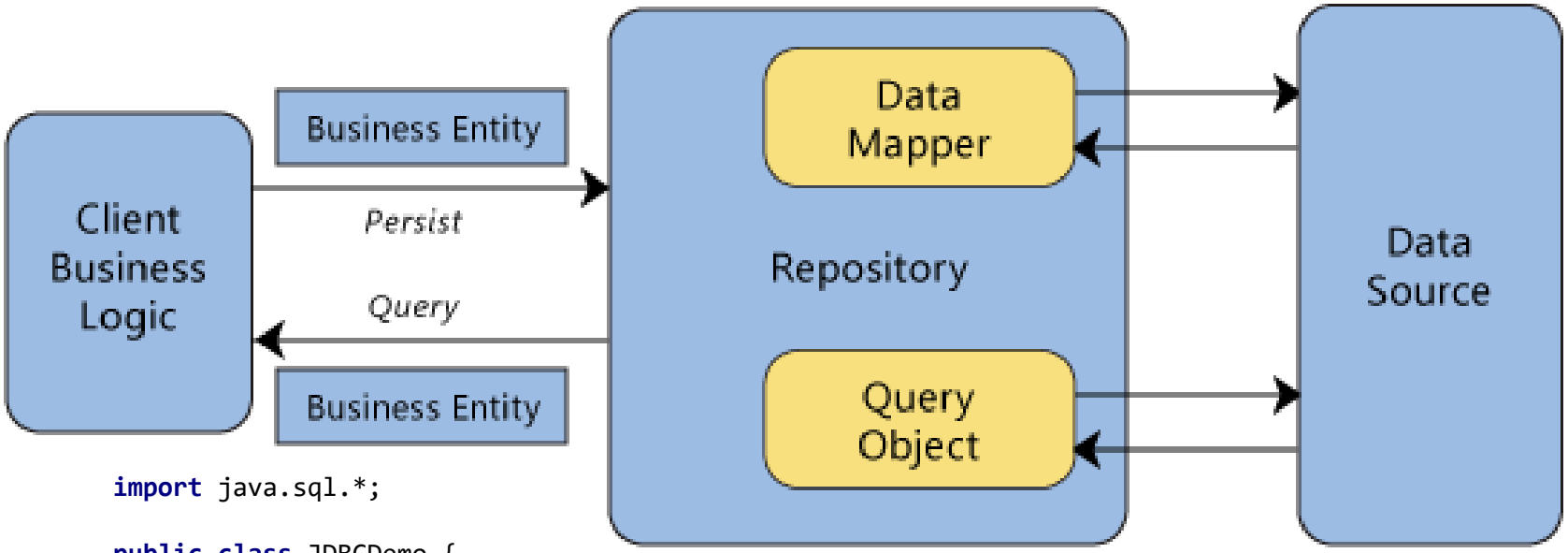
```
public interface UserRepository extends CrudRepository<User, Long> {
	@Query("select u from User u where u.emailAddress = ?1")
	User findByEmailAddress(String emailAddress);
}
```

Named queries through XML

```
<named-query name="User.findByLastname">
	<query>
		select u from User u where u.lastname = ?1
	</query>
 </named-query>
```

...

http://docs.spring.io/spring-data/jpa/docs/current/reference/html/

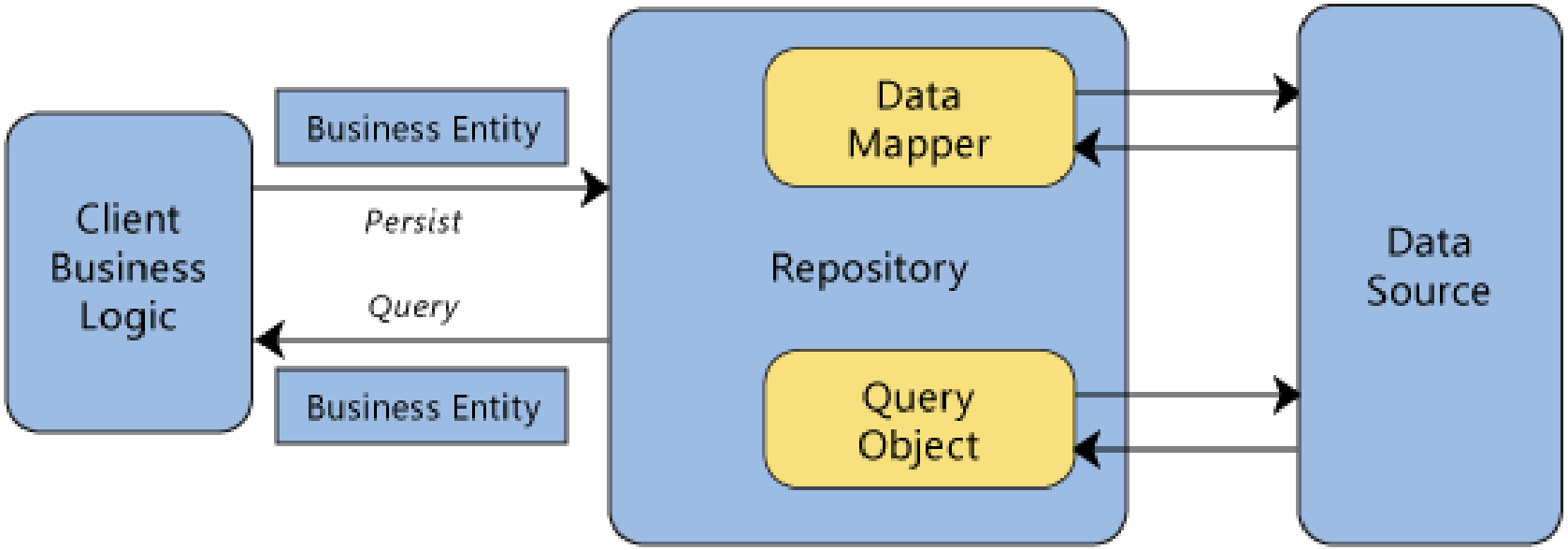



```
import java.sql.*;

public class JDBCDemo {
    public static void main(String[] args) {

        Connection connection;
        PreparedStatement statement;
        ResultSet resultSet;

        try {
            connection = DriverManager.getConnection("jdbc:mysql://localhost/team1?" +
                    "user=root&password=");
            statement = connection.prepareStatement(
                        "select city from ad where prizePerMonth > ?;");
            statement.setInt(1, 750);
            resultSet = statement.executeQuery();
            while (resultSet.next()) {
                System.out.printf("City: %s\n", resultSet.getString(1));
            }
        } catch (SQLException e) {
            // handle exception
        }
    }
}
```




```java
/** Describes an
advertisement that users can
place and search for. */
@Entity
public class Ad {

    @Id
    @GeneratedValue
    private long id;

    @Column(nullable = false)
    private String title;

    @Column(nullable = false)
    private String street;
    /* ... */
}
```

```java
public interface AdDao extend
        CrudRepository<Ad, Long> {
    /* … */
}
```

```xml
<bean id="mainDataSource" class="com.jolbox.bonecp.BoneCPDataSource"
destroy-method="close">
    <property name="driverClass" value="com.mysql.jdbc.Driver" />
    <property name="jdbcUrl"
value="jdbc:mysql://localhost/team1?autoReconnect=true&amp;
                createDatabaseIfNotExist=true&amp;useUnicode=true
&amp;characterEncoding=utf-8" />
    <property name="username" value="root"/>
    <property name="password" value=""/>
    <property name="idleConnectionTestPeriodInMinutes" value="60"/>
    <property name="idleMaxAgeInMinutes" value="240"/>
    <property name="maxConnectionsPerPartition" value="30"/>
    <property name="minConnectionsPerPartition" value="10"/>
    <property name="partitionCount" value="3"/>
    <property name="acquireIncrement" value="5"/>
    <property name="statementsCacheSize" value="100"/>
    <property name="releaseHelperThreads" value="3"/>
</bean>
```

Useful links

- Spring data JPA quick start
  https://spring.io/guides/gs/accessing-data-jpa/

- Spring data JPA reference documentation
  http://docs.spring.io/spring-data/jpa/docs/current/reference/html/

- Spring data guides
  https://spring.io/guides