ORM / Spring Data JPA

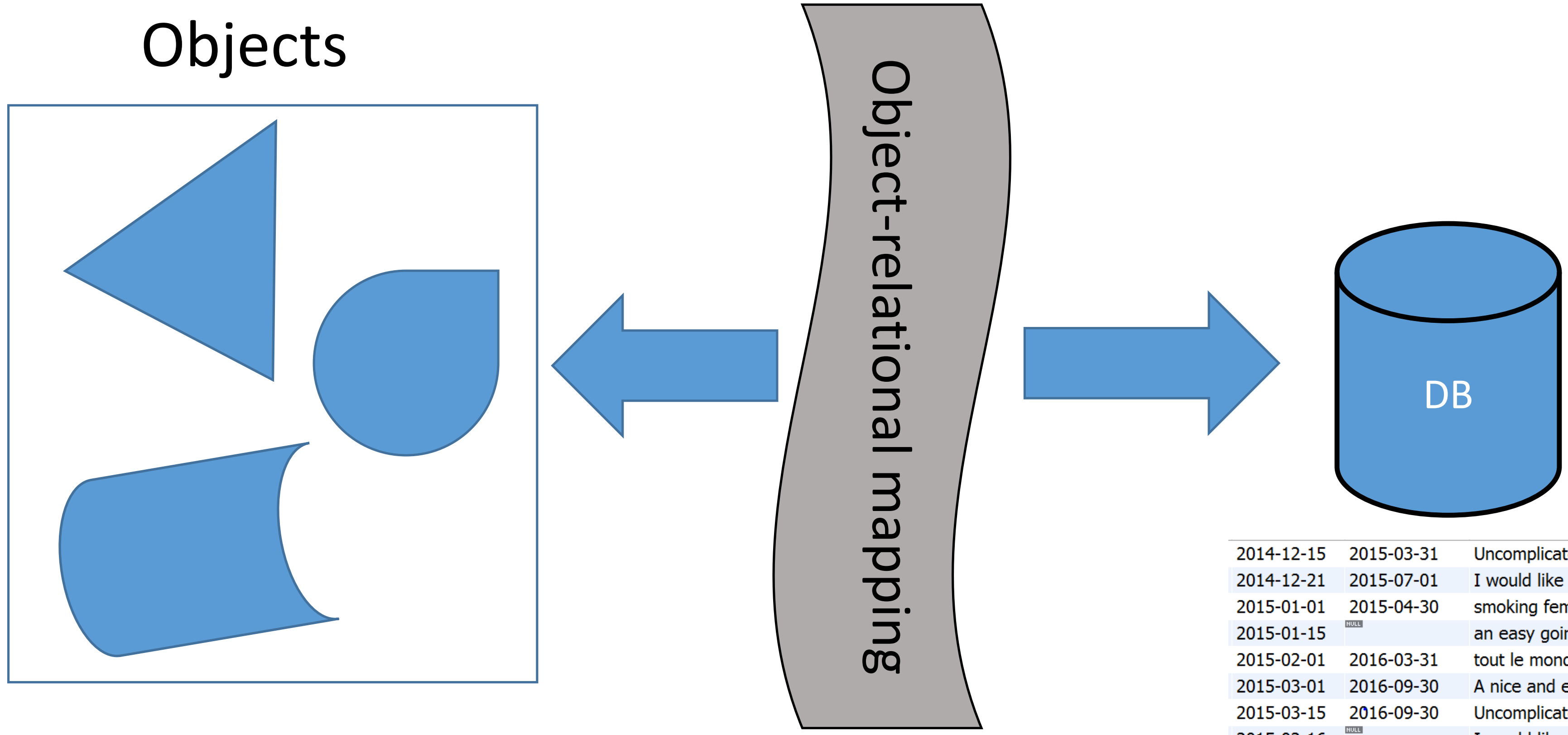
Claudio Corrodi

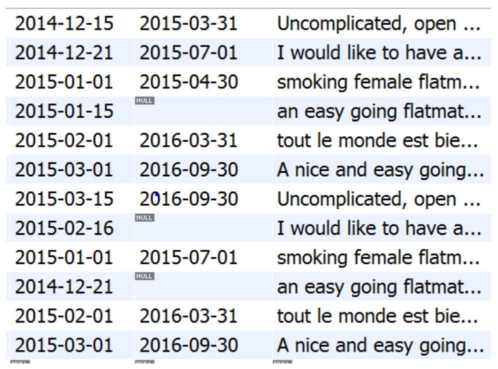
ESE 2016

Traditional DB connections

```java
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
        }
    }
}
```

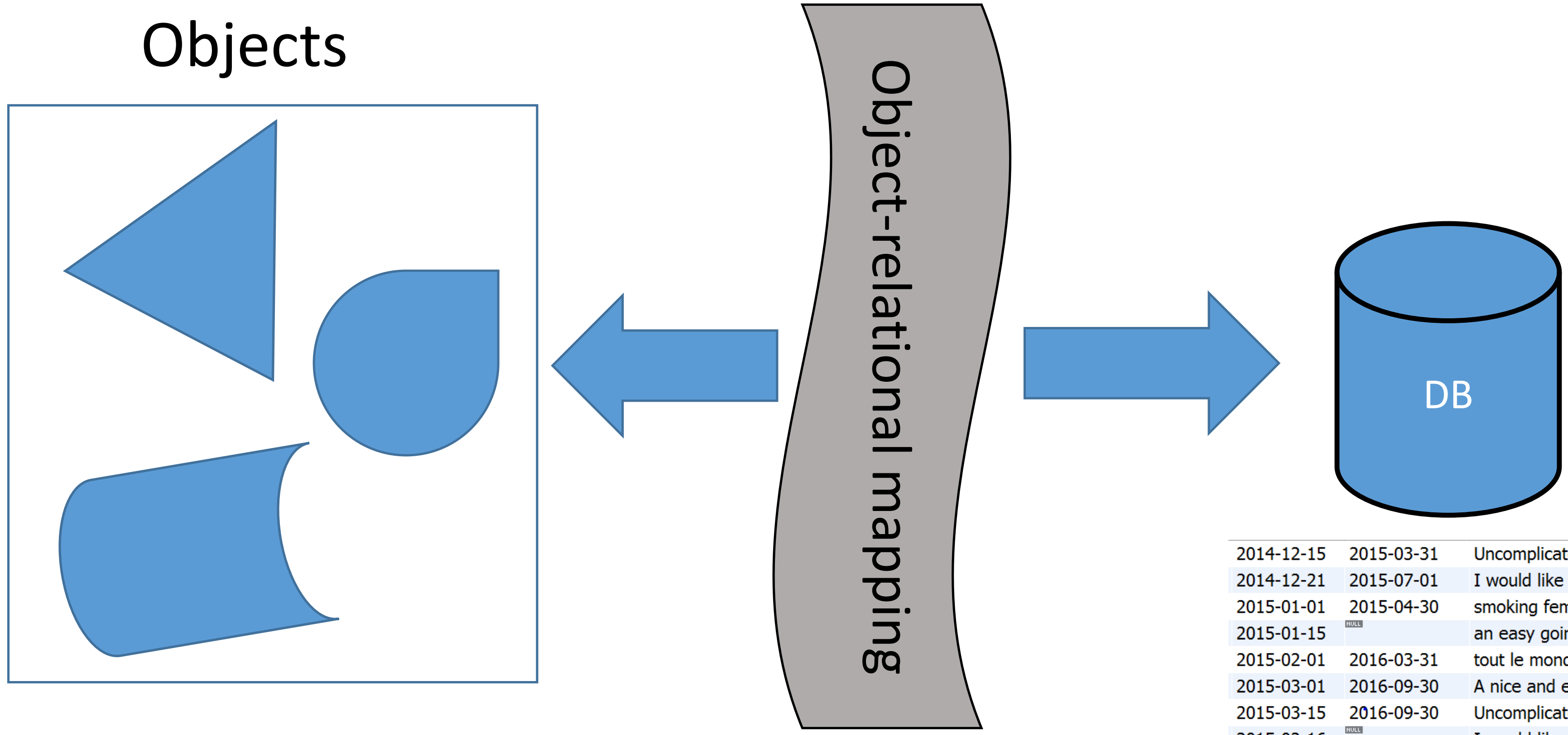
ORM: Object-relational mapping

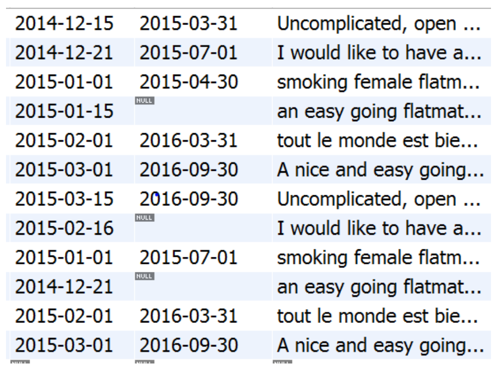




2014-12-15	2015-03-31	Uncomplicated, open ...
2014-12-21	2015-07-01	I would like to have a...
2015-01-01	2015-04-30	smoking female flatm...
2015-01-15	NULL	an easy going flatmat...
2015-02-01	2016-03-31	tout le monde est bie...
2015-03-01	2016-09-30	A nice and easy going...
2015-03-15	2016-09-30	Uncomplicated, open ...
2015-02-16	NULL	I would like to have a...
2015-01-01	2015-07-01	smoking female flatm...
2014-12-21	NULL	an easy going flatmat...
2015-02-01	2016-03-31	tout le monde est bie...
2015-03-01	2016-09-30	A nice and easy going...

ORM: Object-relational mapping



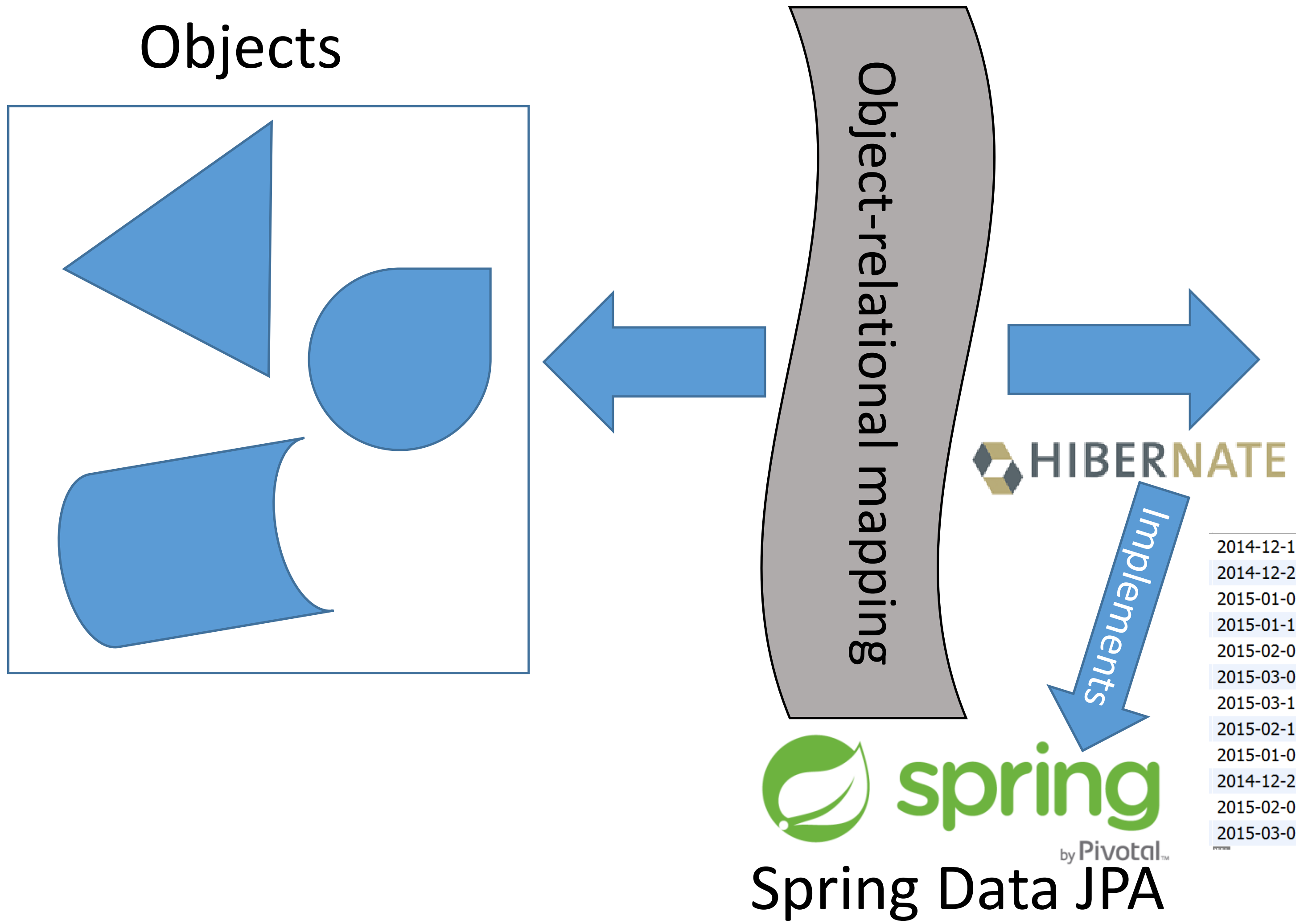


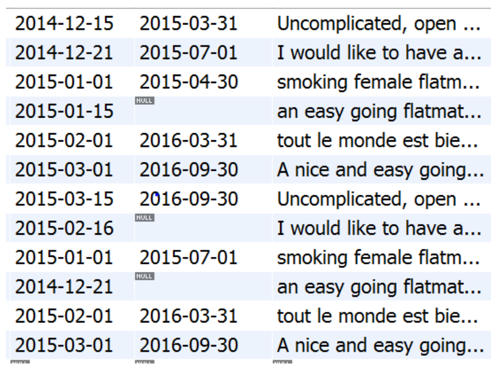
2014-12-15	2015-03-31	Uncomplicated, open ...
2014-12-21	2015-07-01	I would like to have a...
2015-01-01	2015-04-30	smoking female flatm...
2015-01-15	NULL	an easy going flatmat...
2015-02-01	2016-03-31	tout le monde est bie...
2015-03-01	2016-09-30	A nice and easy going...
2015-03-15	2016-09-30	Uncomplicated, open ...
2015-02-16	NULL	I would like to have a...
2015-01-01	2015-07-01	smoking female flatm...
2014-12-21	NULL	an easy going flatmat...
2015-02-01	2016-03-31	tout le monde est bie...
2015-03-01	2016-09-30	A nice and easy going...

spring by Pivotal

Spring Data JPA

ORM: Object-relational mapping





2014-12-15	2015-03-31	Uncomplicated, open ...
2014-12-21	2015-07-01	I would like to have a...
2015-01-01	2015-04-30	smoking female flatm...
2015-01-15	NULL	an easy going flatmat...
2015-02-01	2016-03-31	tout le monde est bie...
2015-03-01	2016-09-30	A nice and easy going...
2015-03-15	2016-09-30	Uncomplicated, open ...
2015-02-16	NULL	I would like to have a...
2015-01-01	2015-07-01	smoking female flatm...
2014-12-21	NULL	an easy going flatmat...
2015-02-01	2016-03-31	tout le monde est bie...
2015-03-01	2016-09-30	A nice and easy going...

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

```
/** 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

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

Keyword queries

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

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

Spring QueryDSL

Type-safe queries similar to SQL

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

userRepository.findAll(predicate);
```

More

@Query annotation

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

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>
```

More

@Query annotation

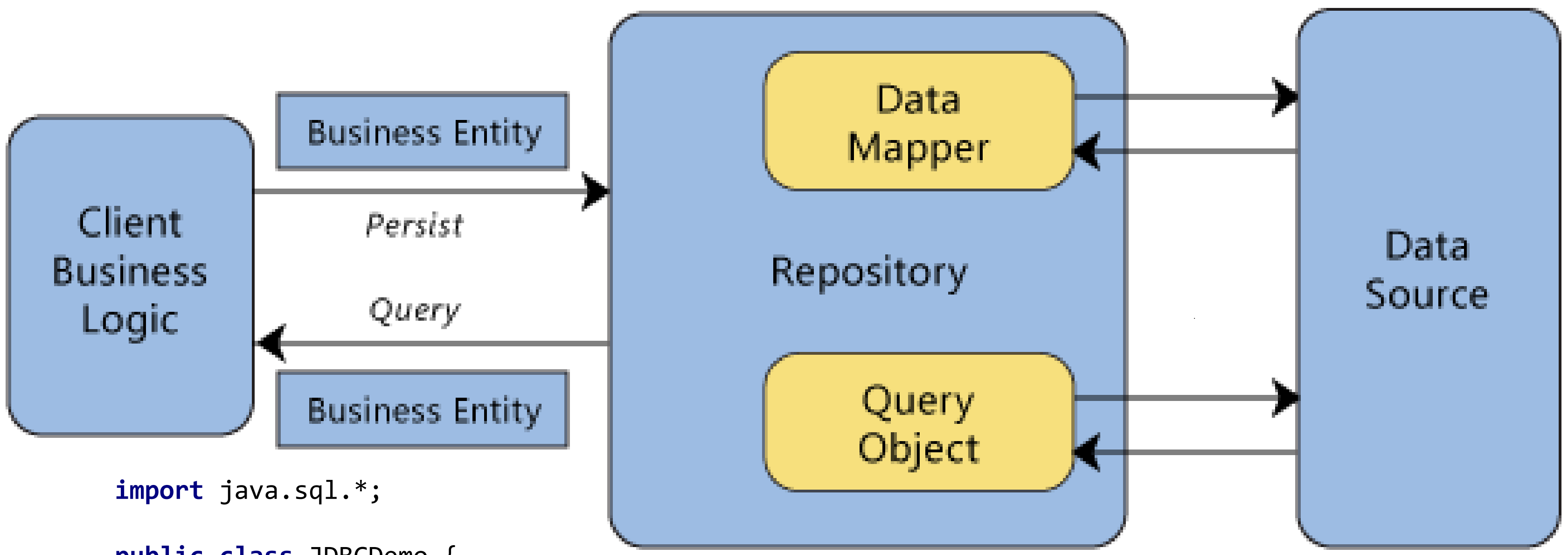
```
public interface UserRepository extends CrudRepository<User, Long> {
       @Query("select u from User u where u.emailAddress = ?1")
       User findByEmailAddress(String emailAddress);
}
```

Named queries through XML

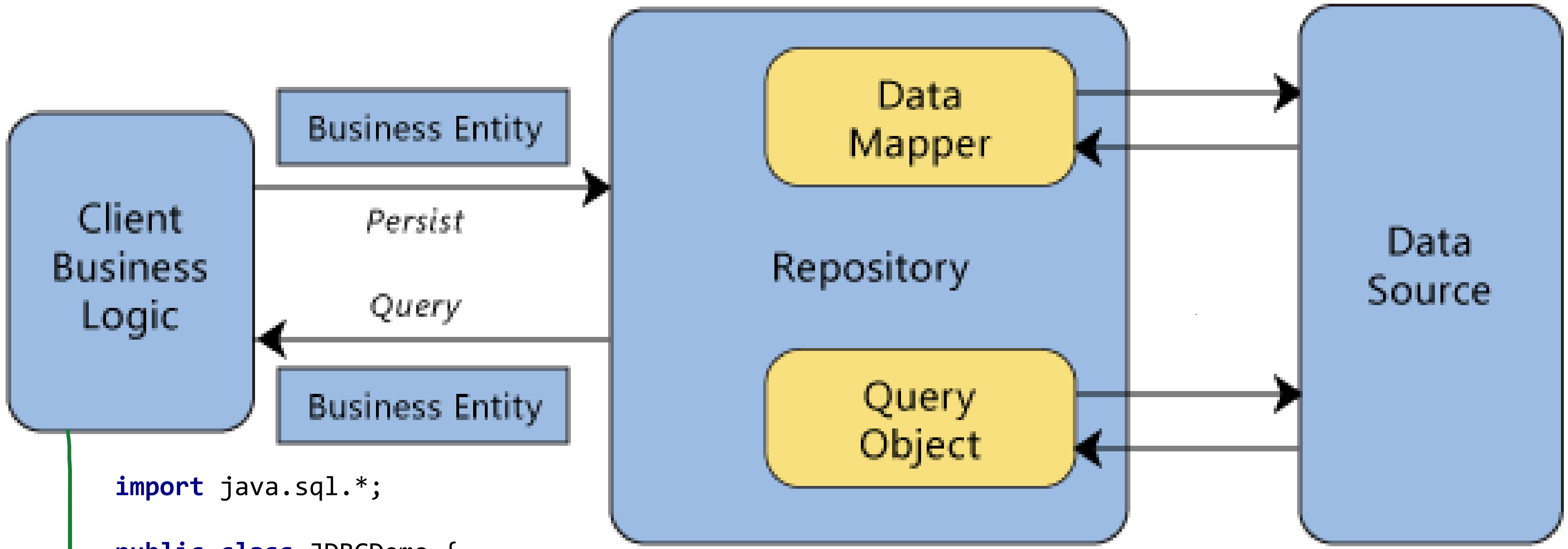
```
<named-query name="User.findByLastname">
       <query>
              select u from User u where u.lastname = ?1
       </query>
 </named-query>
```

…

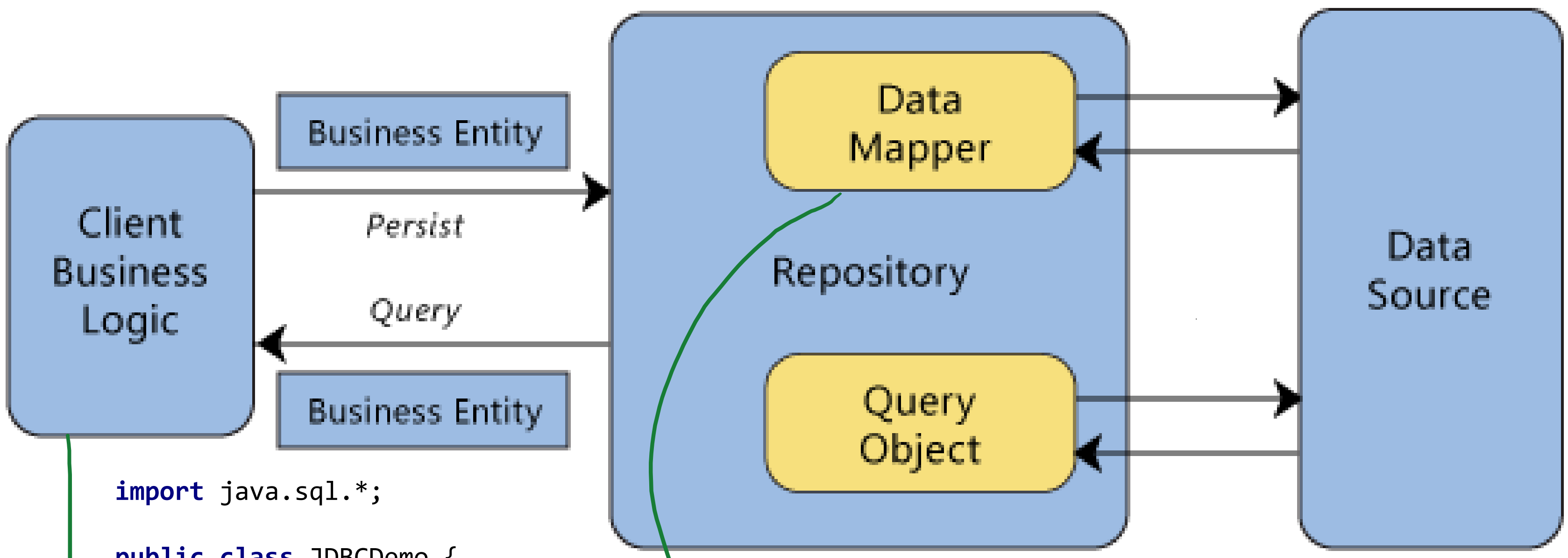
http://docs.spring.io/spring-data/jpa/docs/current/reference/html/



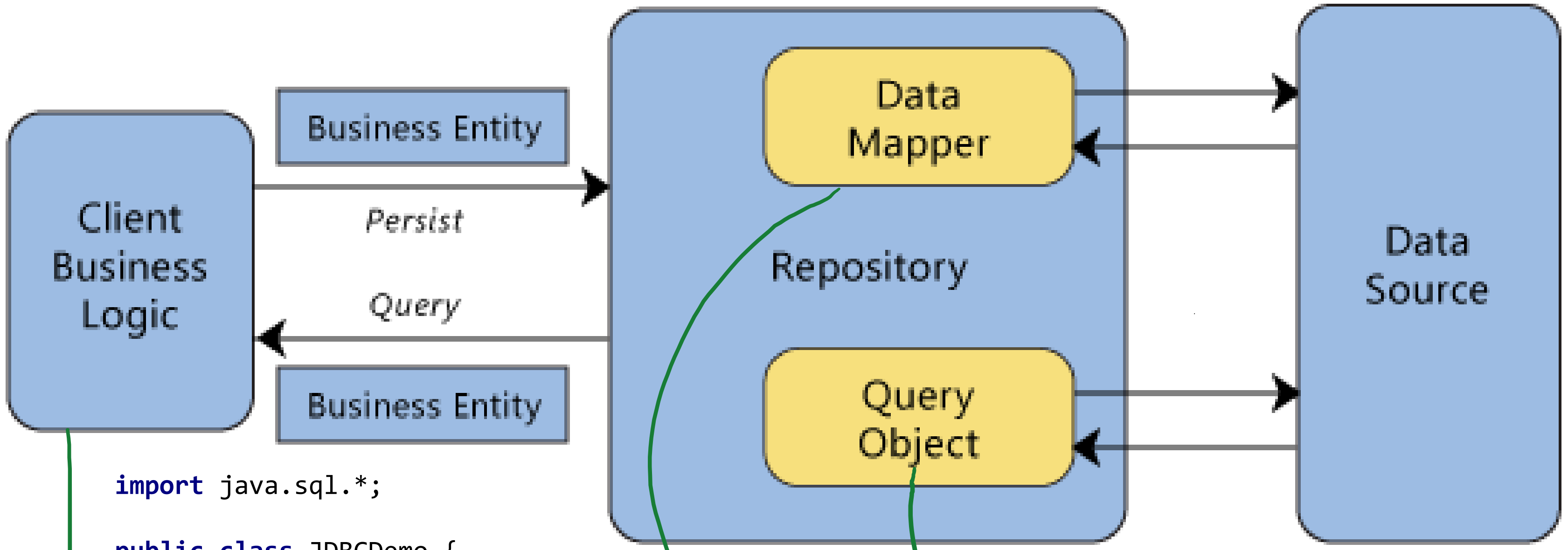
```
import java.sql.*;

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
        }
    }
}
```



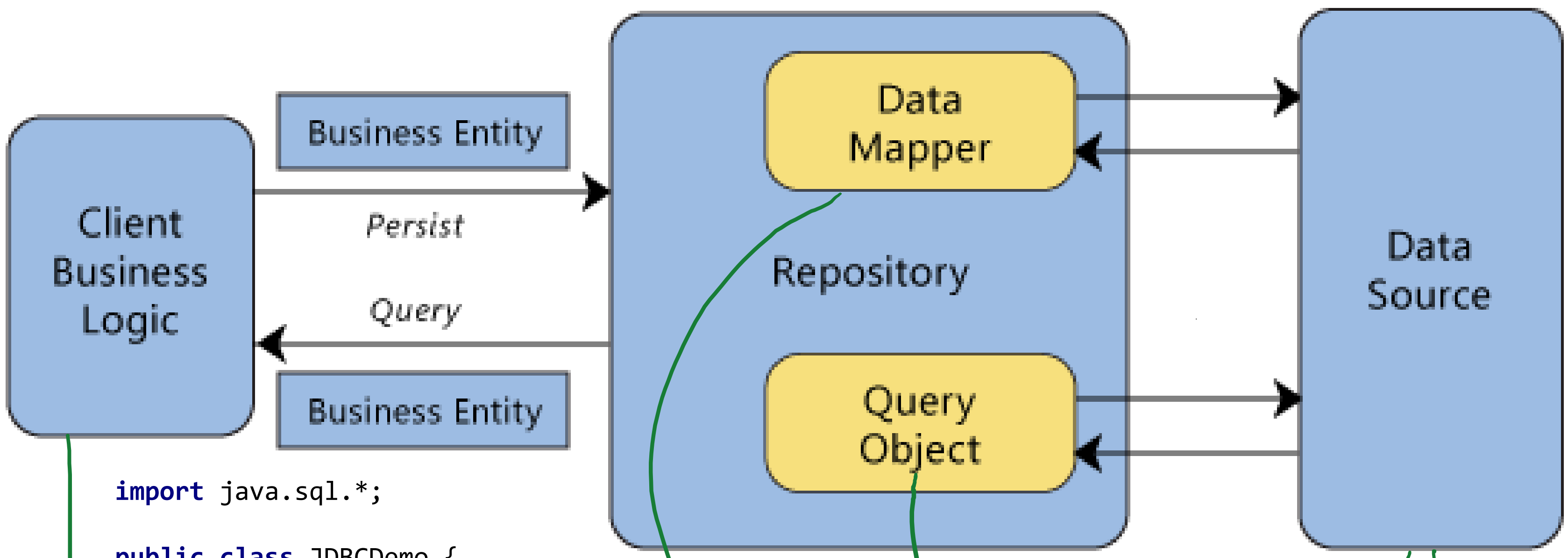
```
import java.sql.*;

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
        }
    }
}
```



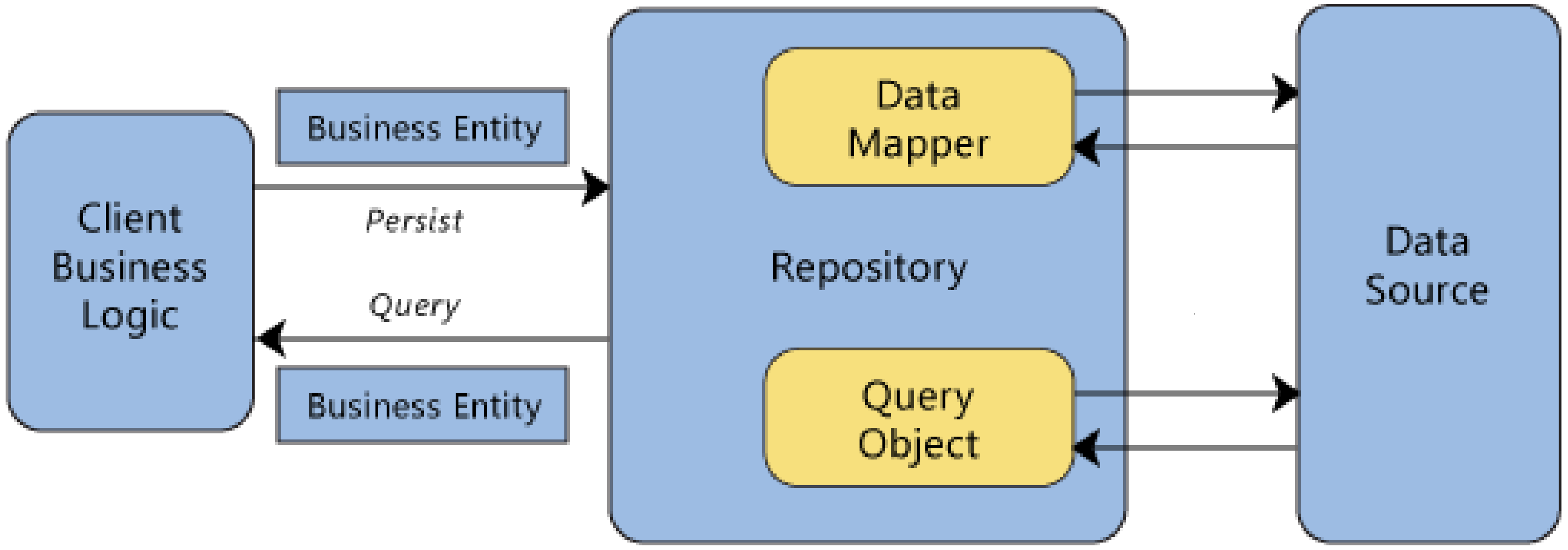
```java
import java.sql.*;

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
        }
    }
}
```



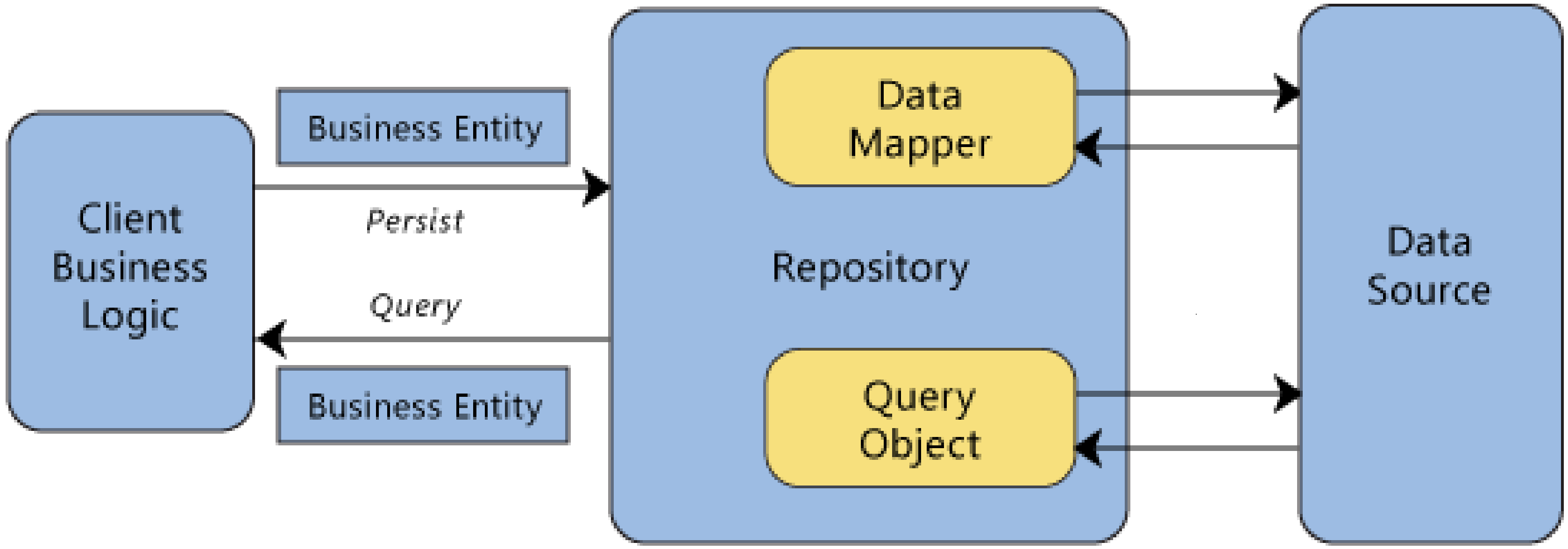
```java
import java.sql.*;

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
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
        }
    }
}
```

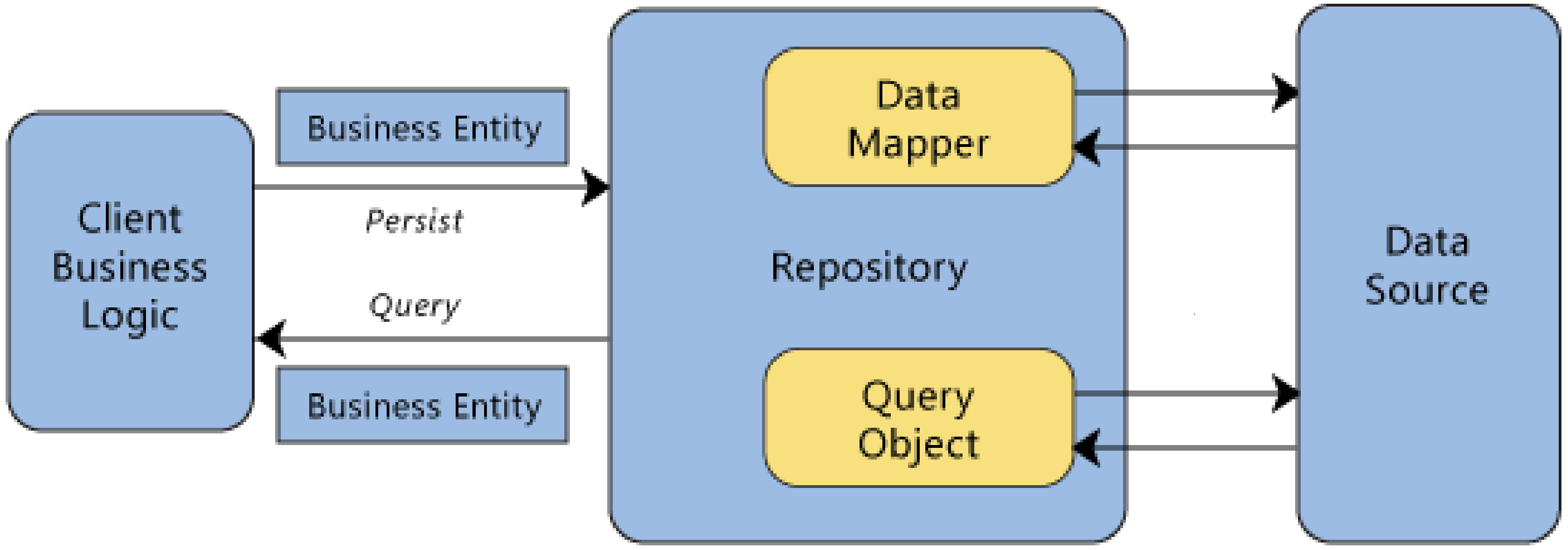


```
/** Describes an
advertisement that users can
place and search for. */
@Entity
public class Ad {

   @Id
   @GeneratedValue
   private long id;

   @Column(nullable = false)
   private String title;

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



```java
/** 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> {
   /* … */
}
```



```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