MODULE 14

QUERIES

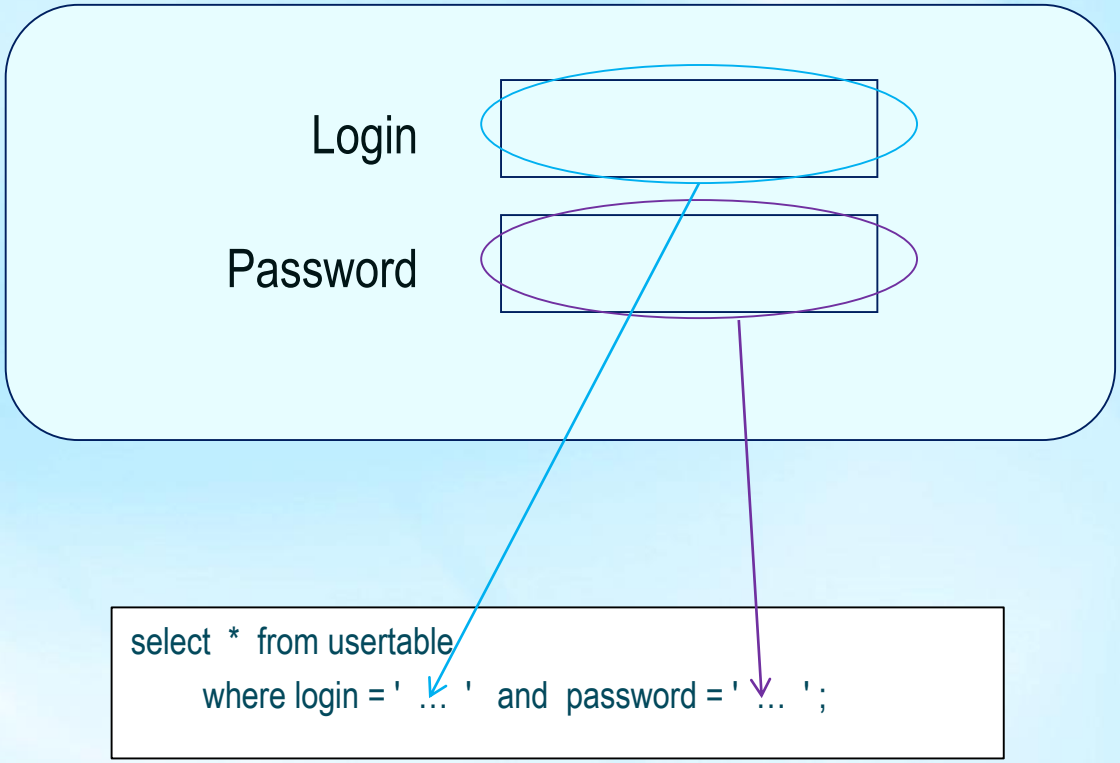
Françoise Dubisy

TABLE OF CONTENT

- SQL Injection
- Query
- Built-in Query from Repository
- Customized Query
- Named Query

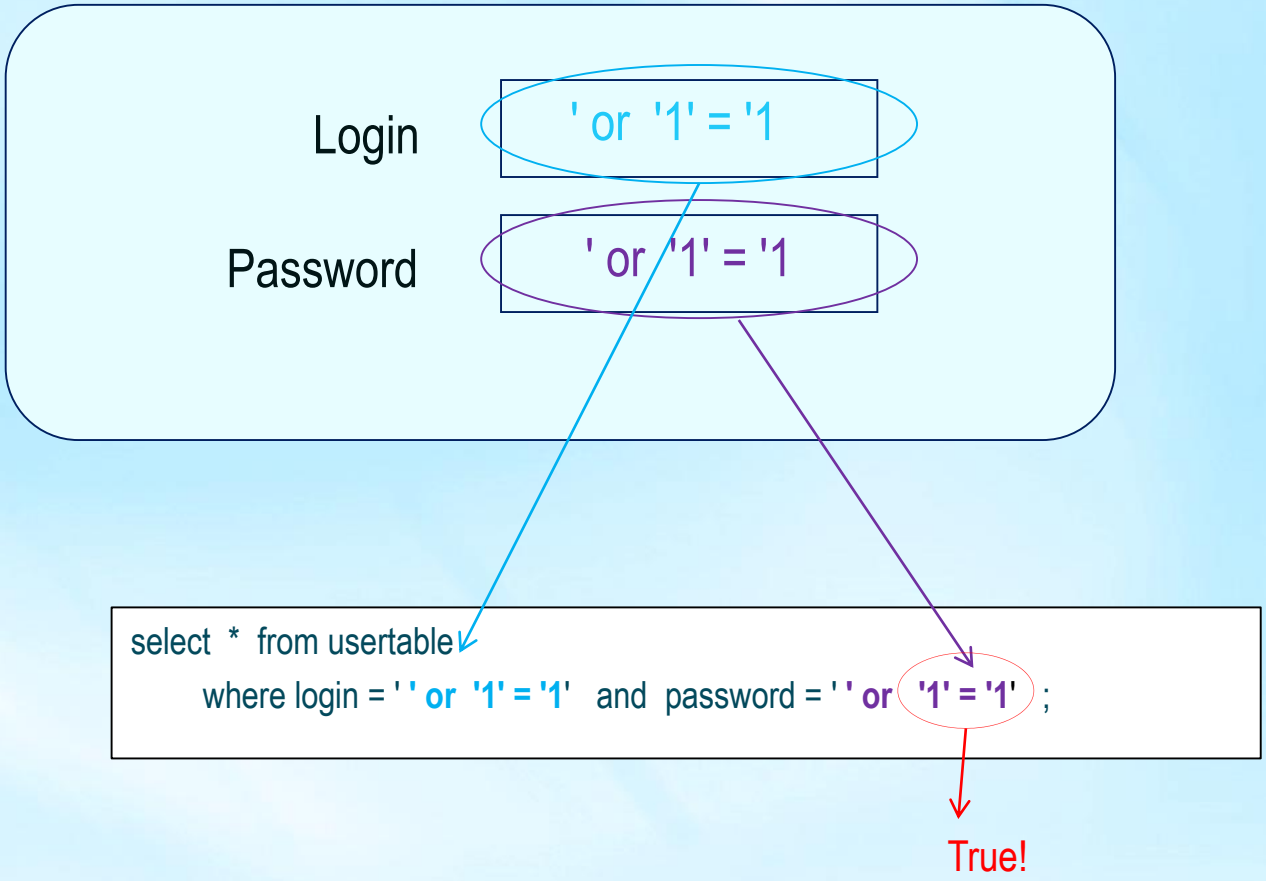
SQL Injection

- To bypass the authentification step



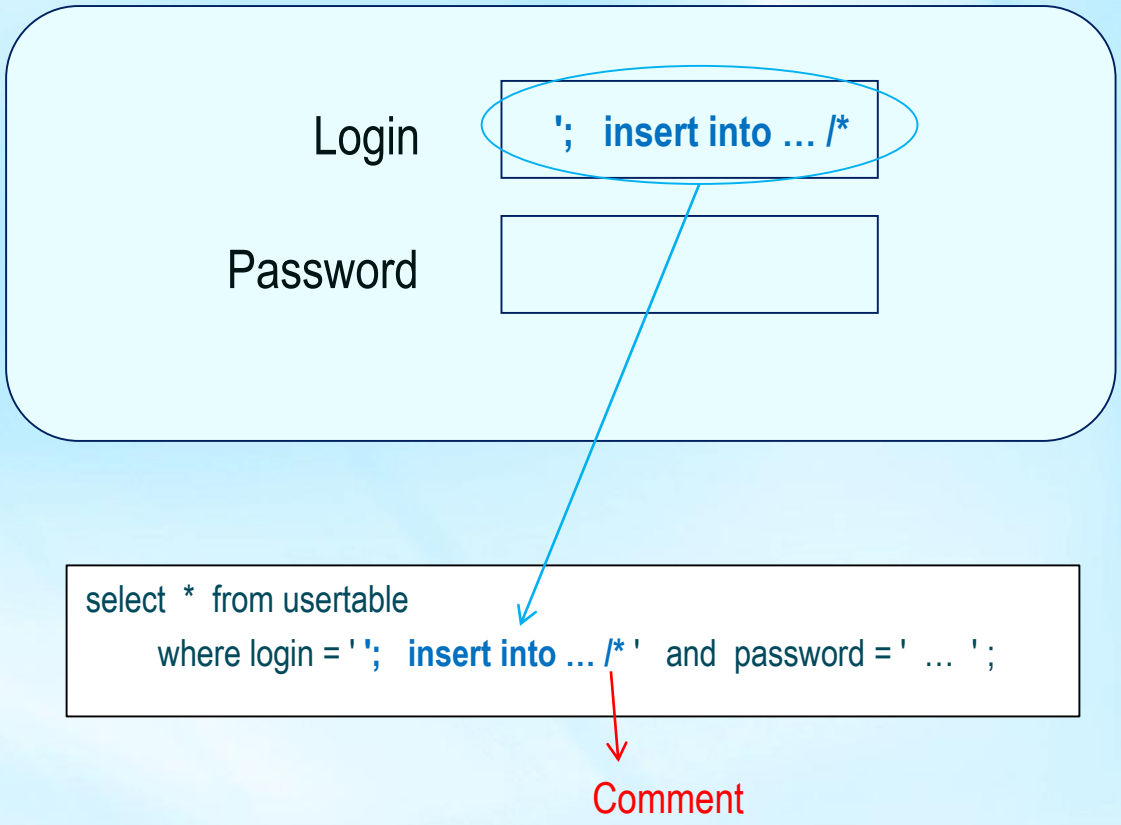
SQL Injection

- E.g,



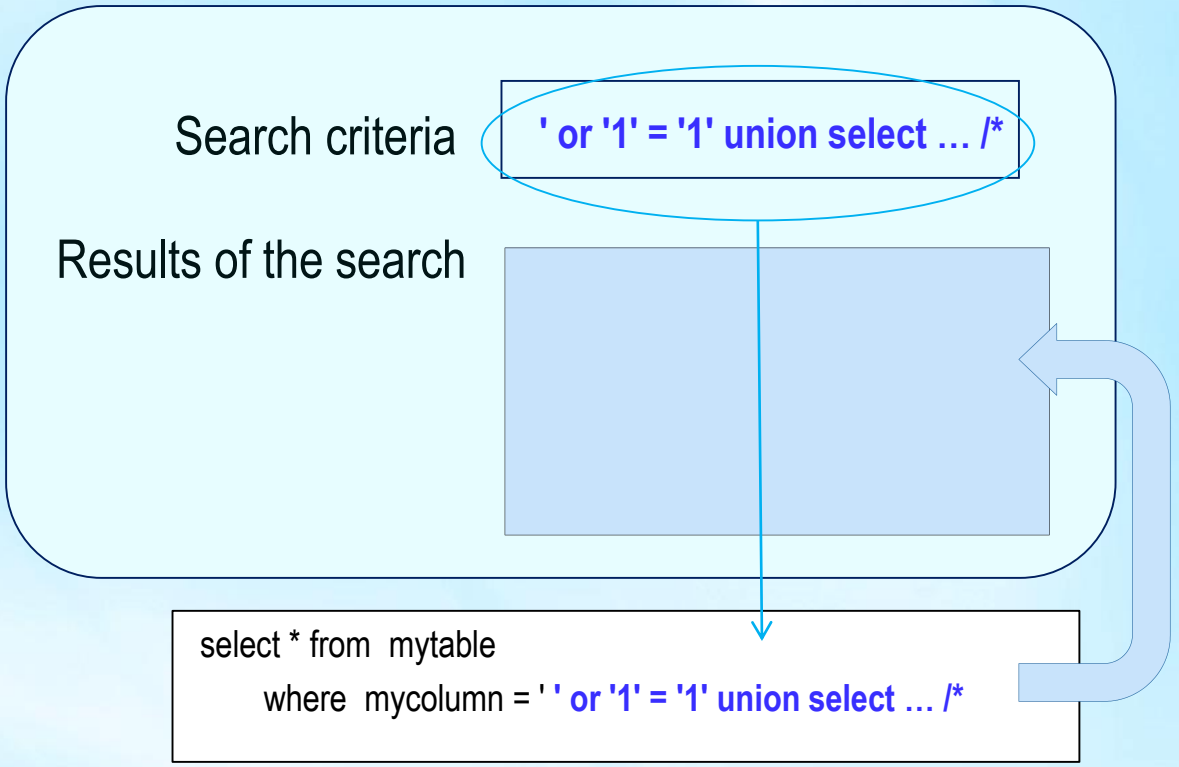
SQL Injection

- To inject data in database
- E.g,



SQL Injection

- To read data from a database
- E.g,



Query

- To access the database in lecture
- Several ways to create query
  - Built-in query from Repository
  - Customized query added to the repository
  - Named Query

Built-in Query from Repository

- Available queries from ***JpaRepository***
- No need to implement!
- *See Entity Beans Module*
- E.g,
  - T **findOne**(ID primaryKey)
  - List<T> **findAll**()
  - Long **count**()
  - boolean **exists**(ID primaryKey)

Customized Query

- Additional queries can be generated in repository interface
  - By combining logical keywords and attribute names

Logical keyword	Keyword expressions
AND	And
OR	Or
AFTER	After, IsAfter
BEFORE	Before, IsBefore
CONTAINING	Containing, IsContaining, Contains
BETWEEN	Between, IsBetween
ENDING_WITH	EndingWith, IsEndingWith, EndsWith
EXISTS	Exists
FALSE	False, IsFalse
GREATER_THAN	GreaterThan, IsGreaterThan
GREATER_THAN_EQUALS	GreaterThanEqual, IsGreaterThanEqual
IN	In, IsIn
IS	Is, Equals, (or no keyword)

http://docs.spring.io/spring-data/jpa/docs/1.4.1.RELEASE/reference/html/jpa.repositories.html

Customized Query

- Continue

IS_NOT_NULL	NotNull, IsNotNull
IS_NULL	Null, IsNull
LESS_THAN	LessThan, IsLessThan
LESS_THAN_EQUAL	LessThanEqual, IsLessThanEqual
LIKE	Like, IsLike
NEAR	Near, IsNear
NOT	Not, IsNot
NOT_IN	NotIn, IsNotIn
NOT_LIKE	NotLike, IsNotLike
REGEX	Regex, MatchesRegex, Matches
STARTING_WITH	StartingWith, IsStartingWith, StartsWith
TRUE	True, IsTrue
WITHIN	Within, IsWithin

http://docs.spring.io/spring-data/jpa/docs/1.4.1.RELEASE/reference/html/jpa.repositories.html

Customized Query

- E.g,

And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull	findByAgeIsNull	… where x.age is null
IsNotNull,NotNull	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1

http://docs.spring.io/spring-data/jpa/docs/1.4.1.RELEASE/reference/html/jpa.repositories.html

Customized Query

- E.g (continue),

OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> age)	… where x.age not in ?1
True	findByActiveTrue()	… where x.active = true
False	findByActiveFalse()	… where x.active = false

http://docs.spring.io/spring-data/jpa/docs/1.4.1.RELEASE/reference/html/jpa.repositories.html

Customized Query

- E.g (continue),

StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)

http://docs.spring.io/spring-data/jpa/docs/1.4.1.RELEASE/reference/html/jpa.repositories.html

Customized Query

- Examples of customized queries

```
@Repository
@Transactional
public interface BookRepository extends JpaRepository<BookEntity, String>{
    public List<BookEntity> findByNbPagesAndTitle(Integer nbPages, String title);
    public List<BookEntity> findByIsbnOrTitle(String isbn, String title);
    public List<BookEntity> findByEditionDateIsNull();
    public List<BookEntity> findByTitleLike(String keyword);
    public List<BookEntity> findByNbPagesIn(Collection<Integer> collectionNbPages);
}
```

Customized Query

- Examples of call of customized queries
  - E.g, in BookDAO

```
bookRepository.findByNbPagesAndTitle(111,"My data and me");
```

```
bookRepository.findByIsbnOrTitle("isbn154962","My data and me");
```

```
bookRepository.findByEditionDateIsNull();
```

```
bookRepository.findByTitleLike("%data%");
```

```
ArrayList<Integer> nbPagesCollection = new ArrayList<Integer>();
nbPagesCollection.add(111);
nbPagesCollection.add(569);
nbPagesCollection.add(444);
List <BookEntity> bookEntities = bookRepository.findByNbPagesIn(nbPagesCollection);
```

Named Query

- Avoid SQL injection by using named queries
- **Declare named queries in entity class**
- Using annotations
  - @NamedQueries
  - @NamedQuery
    - **name** attribute : name of the query
    - **query** attribute : syntax of the query

Named Query

- Syntax for named query
  - ***from*** EntityClass ***entityObject***
  - ***Where***
    - Conditions on properties of the ***entityObject***
    - Names of parameter preceded by **:**

Named Query

- E.g,

Parameter

```
import javax.persistence.*;

@Entity
@Table(name="book")
@NamedQueries({
    @NamedQuery(
          name = "findBookByMoreThanXPages",
          query = "from BookEntity b where b.nbPages > :nbPages"
                ),
    @NamedQuery(
          name = "findBookByPagesAndData",
          query = "from BookEntity b where b.nbPages > :nbPages and b.title like '%data%'"
                ),
    })
public class BookEntity {

    @Id
    @Column(name="isbn")
    private String isbn;

    @Column(name="title")
    private String title;

    @Column(name="nbpages")
    private Integer nbPages;

    @Column(name="editiondate")
    private java.util.Date editionDate;
```

Named Query

- Call named queries in DAO classes
  - Call ***session.getNamedQuery(…)***
    - By using the name of the query
    - By setting values for parameters
      - Using the right setter (e.g, setString, setInteger …)
  - ***getNamedQuery*** returns an instance of ***org.hibernate.Query***
    - Call ***list()*** on this query object
      - Returns a List<Object>

Named Query

- E.g,

```
@Service
@Transactional
public class BookDAO {
    @Autowired
    private SessionFactory sessionFactory;
    @Autowired
    private BookRepository bookRepository;
    @Autowired
    private ProviderConverter providerConverter;

    public ArrayList<Book> findBookByMoreThan500Pages()
    {   Session session = sessionFactory.getCurrentSession();
        session.beginTransaction();
        Query query = session.getNamedQuery("findBookByMoreThanXPages").setInteger("nbPages", 500);
        List <BookEntity> bookEntities = query.list();
        ArrayList <Book> books = new ArrayList<>();
        for (BookEntity entity : bookEntities)
         {Book book= providerConverter.bookEntityToBookModel(entity);
         books.add(book);}
        session.getTransaction().commit();
        return books;
    }
```

To set value of parameters in the named query