

Desenvolupament d'Aplicacions Web

Mòdul 7 – Desenvolupament web en entorn servidor

EAC4

(Curs 2021–22 / 1r semestre)

Solució

En aquest document apareixen les respostes de les preguntes de resposta oberta. Les respostes de les tipus text ja apareixen al qüestionari. L'ordre pot ser diferent en cada cas.

Exercici 1:


Teniu la següent interfície que defineix un Repository:

```
public interface CotxesRepository {  
    Cotxe getCotxeByName(String name); // Retorna el Cotxe que coincideix amb el nom  
    name  
    List<Cotxe> getAllCotxes(); // Retorna un llistat amb tots els Cotxes a la BD  
    void changeCotxeStock(Cotxe cotxe, int numUnits); // Canvia l'estoc del cotxe,  
    substituint numUnitatsDisp per el valor de numUnits  
}
```

amb el model:

```
public class Cotxe {  
    private String nom;  
    private Double preu;  
    private int numUnitatsDisp;  
    // Getters i setters  
}
```

Desenvolueu un DAO utilitzant JDBC. Podeu assumir que teniu la connexió establerta i disponible. Es valorarà que refactoritzeu el codi per reutilitzar parts comunes.

	Codi: I71	Exercici d'avaluació continuada 4	Pàgina 1 de 8
	Versió: 03	DAW_M07B2_EAC4_Solucio_2122S1	Lliurament: 19/10/2021

```
package cat.xtec.ioc.repository.impl;

import cat.xtec.ioc.domain.Cotxe;
import cat.xtec.ioc.repository.CotxesRepository;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Repository;

@Repository
public class CotxesDAO implements CotxesRepository {
    private Connection connection;
    @Override
    public Cotxe getCotxeByName(String name) {
        String qry = "select * from cotxes where name = ?";
        PreparedStatement preparedStatement;
        try {
            preparedStatement = connection.prepareStatement(qry);
            preparedStatement.setString(1, name);
            return executeQuery(preparedStatement).get(0);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return null;
    }

    @Override
    public List<Cotxe> getAllCotxes() {
        String qry = "select * from cotxes";
        PreparedStatement preparedStatement;
        try {
            preparedStatement = connection.prepareStatement(qry);
            List<Cotxe> cotxes = executeQuery(preparedStatement);
            return cotxes;
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return null;
    }
}
```

```
private List<Cotxe> executeQuery(PreparedStatement preparedStatement) {
    List<Cotxe> cotxes = new ArrayList<>();

    try {
        ResultSet rs = preparedStatement.executeQuery() {
            while (rs.next()) {
                Cotxe cotxe = buildCotxeFromResultSet(rs);
                cotxes.add(cotxe);
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return cotxes;
    }

@Override
public void changeCotxeStock(Cotxe cotxe, int numUnits) {
    String qry = "UPDATE cotxes SET numUnitatsDisp = ? WHERE nom = ?";
    PreparedStatement preparedStatement;
    try {
        preparedStatement = connection.prepareStatement(qry);
        preparedStatement.setInt(1, numUnits);
        preparedStatement.setString(2, cotxe.getNom());
        preparedStatement.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}


private Cotxe buildCotxeFromResultSet(ResultSet rs) throws SQLException {
    String nom = rs.getString("nom");
    Double preu = rs.getDouble("preu");
    int numUnitatsDisp = rs.getInt("numUnitatsDisp");
    Cotxe cotxe = new Cotxe(nom, preu, numUnitatsDisp);
    return cotxe;
}
}
```

Exercici 2:

Quin és el motiu pel qual els tests unitaris són acumulatius? és a dir, un cop un test passa no es treu del conjunt de test, si no que es deixa?

Feu un raonament curt però tan precís com pugueu.

A l'implementar noves funcionalitats podem fer que alguna part ja desenvolupada deixi de funcionar, per això cal passar tots els tests cada vegada que fem build, per assegurar que tot continua funcionant de forma correcta.

	Codi: I71	Exercici d'avaluació continuada 4	Pàgina 3 de 8
	Versió: 03	DAW_M07B2_EAC4_Solucio_2122S1	Lliurament: 19/10/2021

Exercici 3:

Es disposa del següent mètode per desactivar un usuari:

```
public void deleteUser(User user) throws Exception {
    String qry = "UPDATE users SET active = false where user_id = ?";
    PreparedStatement preparedStatement = getPreparedStatement(qry);
    preparedStatement.setInt(1, user.getUserId());
    createOrUpdateUser(user.getUsername(), preparedStatement);
}
```

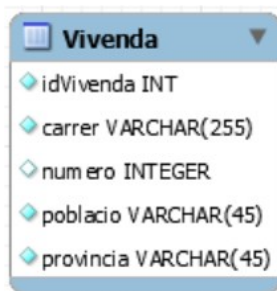
Creeu un test unitari per comprovar-ne el funcionament correcte. Podeu assumir-ne el DAO fet.

```
@Test
public void deleteUser() throws Exception {
    String username = "testUser";
    String name = "Pete Test";
    String email = "pete@email.com";
    User createdUser = userDao.createUser(username, name, email);
    Assert.assertNotNull(createdUser);
    userDao.deleteUser(createdUser);
    User deletedUser = userDao.findUserByUsername(username);
    Assert.assertNotNull(deletedUser);
    Assert.assertFalse(deletedUser.isActive());
}
```

Exercici 4:

Ens encarreguen el desenvolupament d'una app web per al lloguer de vivendes.

Implementeu la següent entitat, mitjançant un POJO JPA i amb les restriccions adients.



```
package cat.xtec.ioc.domain;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
```

	Codi: I71	Exercici d'avaluació continuada 4	Pàgina 4 de 8
	Versió: 03	DAW_M07B2_EAC4_Solucio_2122S1	Lliurament: 19/10/2021

@Entity

```
public class Vivenda implements Serializable {  
    private static final long serialVersionUID = 1L;  
  
    @Id  
    @NotNull  
    @Column(name = "idVivenda")  
    private Integer idVivenda;  
  
    @NotNull  
    @Size(max = 255)  
    @Column(name = "carrer")  
    private String carrer;  
  
    @Column(name = "numero")  
    private Integer numero;  
  
    @NotNull  
    @Size(max = 45)  
    @Column(name = "poblacio")  
    private String poblacio;  
  
    @NotNull  
    @Size(max = 45)  
    @Column(name = "provincia")  
    private String provincia;  
  
    public Vivenda() {}  
  
    public Integer getIdVivenda() {  
        return idVivenda;  
    }  
  
    public void setIdVivenda(Integer idVivenda) {  
        this.idVivenda = idVivenda;  
    }  
  
    public String getCarrer() {  
        return carrer;  
    }  
  
    public void setCarrer(String carrer) {  
        this.carrer = carrer;  
    }  
  
    public Integer getNumero() {  
        return numero;  
    }  
}
```

```
public void setNumero(Integer numero) {
    this.numero = numero;
}

public String getPoblacio() {
    return poblacio;
}

public void setPoblacio(String poblacio) {
    this.poblacio = poblacio;
}

public String getProvincia() {
    return provincia;
}

public void setProvincia(String provincia) {
    this.provincia = provincia;
}
```

Exercici 5:


Tenim la següent interface:

```
public interface VideojocRepository {
    List<Videojoc> getAllVideojocs();
}
```

Escriu el mètode del DAO en el supòsit que treballem amb JDBC (assumint la connexió feta) i si treballem amb JPA (EntityManager llest per utilitzar)

JDBC:

```
public List <Videojoc> getAllVideojocs ( ) throws SQLException {
    String qry = " select * from Videojoc " ;
    List <Videojoc> videojocs = new ArrayList < > ( ) ;
    try (
        Statement stmt = conn.createStatement ( ) ;
        ResultSet rs = stmt.executeQuery ( qry ) ; ) {
        while (rs.next ( ) ) {
            String nomVideojoc = rs.getString ( "nomVideojoc");
            double preu = rs.getDouble ( "preu" ) ;
            String genere = rs.getString ( "genere" ) ;
            String pegi = rs.getString ( "pegi" ) ;
            int unitatsVenudes = rs.getInt ( "unitatsVenudes" ) ;
            Videojoc videojoc = new Videojoc(nomVideojoc, preu,
            genere, pegi, unitatsVenudes) ;
        }
    }
}
```

	Codi: I71	Exercici d'avaluació continuada 4	Pàgina 6 de 8
	Versió: 03	DAW_M07B2_EAC4_Solucio_2122S1	Lliurament: 19/10/2021

```
        if ( videojoc != null ) {  
            videojocs.add( videojoc ) ;  
        }  
    }  
    } catch ( SQLException | IOException e ) {  
        e.printStackTrace ( ) ;  
    }  
    return videojocs;  
}
```

JPA:

```
public List <Videojoc> getAllVideojocs ( ) {  
    return (List <Videojoc> entityManager.createQuery ( " Select * from  
Videojoc").getResultList( );  
}
```

Exercici 6:


Tenim la següent interfície:

```
public interface VolsRepository {  
    Vol getVolByDesti(String desti);  
    List<Vol> getAllVols();  
    List<Vol> getVolByCompany(String company);  
    void addVol(Vol vol);  
    void updateVol(Vol vol);  
}
```

Escriu el DAO Hibernate que l'implementi

@Repository

```
public class VolHibernate implements VolsRepository {  
  
    @Override  
    public Vol getVolByDesti(String desti) {  
        return getSession().get(Vol.class, desti);  
    }  
  
    @Override  
    public List<Vol> getAllVols() {  
        Criteria criteria = createEntityCriteria();  
        return (List<Vol>) criteria.list();  
    }  
  
    @Override  
    public List<Vol> getVolByComany(String company) {  
        return (List<Vol>) getSession().get(Vol.class, company)  
    }  
}
```

	Codi: I71	Exercici d'avaluació continuada 4	Pàgina 7 de 8
	Versió: 03	DAW_M07B2_EAC4_Solucio_2122S1	Lliurament: 19/10/2021

```

@Override
public void addVol(Vol vol) {
    getSession().saveOrUpdate(vol);
}

@Override
public void updateVol(Vol vol) {
    getSession().merge(vol);
}

protected Session getSession() {
    return sessionFactory.getCurrentSession();
}

private Criteria createEntityCriteria() {
    return getSession().createCriteria(Vol.class);
}
}

```

Exercici 7:

Com puc modificar el pom.xml per no haver de modificar moltes vegades el valor de la versió de les dependències de Spring?

Caldrà afegir una propietat:

```

<properties>
    <Spring.version>4.3.4.RELEASE</Spring.version>
</properties>

```

i modificar les dependències utilitzant la property creada:

```

(...)
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>${Spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${Spring.version}</version>
    <scope>test</scope>
</dependency>
(...)

```