TP : Système de messagerie **exelib.net**

- **Fichiers properties**
- **PrepareStatement**

Une société souhaite mettre en place un système de messagerie entre ses employés.

Les travaux de l'équipe chargée de l'analyse et de la conception ont abouti au diagramme de classe suivant :

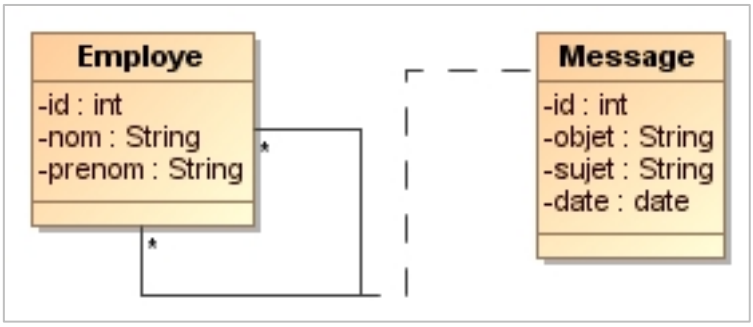


Diagramme de classe

1. Créer dans la racine de votre projet un fichier **properties** nommé **base** qui contient les informations de connexion à la base de données **messagerie**.

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost/messagerie
jdbc.username=root
jdbc.password=root
```

2. Créer une classe **Connexion** permettant d'ouvrir une connexion à la base de données dans le package « ma.projet.connexion » :

```
public class Connexion {

    private static Connection connection;

    static {
        try {
            FileInputStream f = new FileInputStream("base.properties");
            Properties p = new Properties();
            p.load(f);
            String url = p.getProperty("jdbc.url");
            String login = p.getProperty("jdbc.username");
            String password = p.getProperty("jdbc.password");
            String driver = p.getProperty("jdbc.driver");
            Class.forName(driver);
            connection = DriverManager.getConnection(url, login, password);
        } catch (Exception ex) {
            System.out.println(""+ex.getMessage());
        }
    }

    public static Connection getConnection() {
        return connection;
    }
}
```

3. Créer les classes **Employe** et **Message** dans le package « ma.projet.beans » :

Employe

```
public class Employe {

            private int id;
            private String nom;
            private String prenom;

            public Employe(String nom, String prenom) {
                this.nom = nom;
                this.prenom = prenom;
            }
            public Employe(int id, String nom, String prenom) {
                this.id = id;
                this.nom = nom;
                this.prenom = prenom;
            }
            public String getNom() {
                return nom;
            }
            public String getPrenom() {
                return prenom;
            }
            public int getId() {
                return id;
            }
            public void setNom(String nom) {
                this.nom = nom;
            }
            public void setPrenom(String prenom) {
                this.prenom = prenom;
            }
}
```

Message

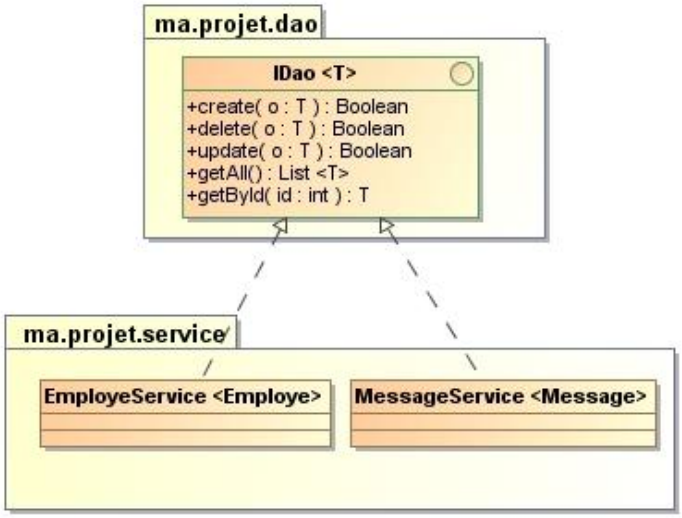
```
public class Message {
         private int id;
         private String object;
         private String sujet;
         private Date date;
         private Employe empEmetteur;
         private Employe empRecepteur;

         public Message(String object, String sujet, Date date, Employe empEmetteur,
             Employe empRecepteur) {
             this.object = object;
             this.sujet = sujet;
             this.date = date;
             this.empEmetteur = empEmetteur;
             this.empRecepteur = empRecepteur;
         }
         public Message(int id, String object, String sujet, Date date, Employe empEmetteur,
             Employe empRecepteur) {
             this.id = id;
             this.object = object;
             this.sujet = sujet;
             this.date = date;
             this.empEmetteur = empEmetteur;
             this.empRecepteur = empRecepteur;
         }
         public int getId() {
             return id;
         }
         public void setId(int id) {
             this.id = id;
         }
         public String getObject() {
             return object;
         }
         public void setObject(String object) {
             this.object = object;
         }
```

```
        public String getSujet() {
            return sujet;
        }
        public void setSujet(String sujet) {
            this.sujet = sujet;
        }
        public Date getDate() {
            return date;
        }
        public void setDate(Date date) {
            this.date = date;
        }
        public Employe getEmpEmetteur() {
            return empEmetteur;
        }
        public void setEmpEmetteur(Employe empEmetteur) {
            this.empEmetteur = empEmetteur;
        }
        public Employe getEmpRecepteur() {
            return empRecepteur;
        }
        public void setEmpRecepteur(Employe empRecepteur) {
            this.empRecepteur = empRecepteur;
        }
}
```

4. Créer l'interface **IDao** suivante dans le package « ma.projet.dao » :


```
public interface IDao<T> {
    boolean create(T o);
    boolean update(T o);
    boolean delete(T o);
    T getById(int id);
    List<T> getAll();
}
```

5. Créer les classes **Service** qui implémente l'interface **IDao** :



EmployeService

```
public class EmployeService implements IDao<Employe> {

    @Override
    public boolean create(Employe o) {
        try {
            String req = "insert into employe (nom, prenom) values(?,?)";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ps.setString(1, o.getNom());
            ps.setString(2, o.getPrenom());
            if (ps.executeUpdate() == 1) {
                return true;
            }
        } catch (SQLException ex) {
            Logger.getLogger(EmployeService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return false;

    }
    @Override
    public boolean update(Employe o) {
         try {
            String req = "update employe set nom = ? , prenom = ? where id = ?";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ps.setString(1, o.getNom());
            ps.setString(2, o.getPrenom());
            ps.setInt(3, o.getId());
            if (ps.executeUpdate() == 1) {
                return true;
            }
        } catch (SQLException ex) {
            Logger.getLogger(EmployeService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return false;
    }

    @Override
    public boolean delete(Employe o) {
        try {
            String req = "delete from employe where id = ?";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ps.setInt(1, o.getId());
            if (ps.executeUpdate() == 1) {
                return true;
            }
        } catch (SQLException ex) {
            Logger.getLogger(EmployeService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return false;
    }

    @Override
    public Employe getById(int id) {
        Employe employe = null;
        try {
            String req = "select * from employe where id = ?";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if(rs.next())
                employe = new Employe(rs.getInt("id"), rs.getString("nom"), rs.getString("prenom"));
        } catch (SQLException ex) {
            Logger.getLogger(EmployeService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return employe;
    }

    @Override
    public List<Employe> getAll() {
        List <Employe> employes = new ArrayList<>();
        try {
            String req = "select * from employe ";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ResultSet rs = ps.executeQuery();
            while(rs.next())
                employes.add(new Employe(rs.getInt("id"), rs.getString("nom"), rs.getString("prenom")));
        } catch (SQLException ex) {
            Logger.getLogger(EmployeService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return employes;
    }

}
```

MessageService

```
public class MessageService implements IDao<Message> {

    private EmployeService es;

    public MessageService() {
        es = new EmployeService();
    }

    @Override
    public boolean create(Message o) {
        try {
            String req = "insert into message (objet, sujet, date, idE, idR) values(?,?,?,?,?)";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ps.setString(1, o.getObject());
            ps.setString(2, o.getSujet());
            ps.setDate(3, new Date(o.getDate().getTime()));
            ps.setInt(4, o.getEmpEmetteur().getId());
            ps.setInt(5, o.getEmpRecepteur().getId());
            if (ps.executeUpdate() == 1) {
                return true;
            }
        } catch (SQLException ex) {
            Logger.getLogger(MessageService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return false;

    }

    @Override
    public boolean update(Message o) {
        return false;
    }

    @Override
    public boolean delete(Message o) {
        return false;
    }

    @Override
    public Message getById(int id) {
        Message employe = null;
        try {
            String req = "select * from message where id = ?";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                employe = new Message(rs.getInt("id"), rs.getString("objet"), rs.getString("sujet"),
      rs.getDate("date"), es.getById(rs.getInt("idE")), es.getById(rs.getInt("idR")));
            }
        } catch (SQLException ex) {
            Logger.getLogger(MessageService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return employe;
    }

    @Override
    public List<Message> getAll() {
        List<Message> employes = new ArrayList<>();
        try {
            String req = "select * from message ";
            PreparedStatement ps = Connexion.getConnection().prepareStatement(req);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                employes.add(new Message(rs.getInt("id"), rs.getString("objet"), rs.getString("sujet"),
      rs.getDate("date"), es.getById(rs.getInt("idE")), es.getById(rs.getInt("idR"))));
            }
        } catch (SQLException ex) {
            Logger.getLogger(MessageService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return employes;
    }
```

6. Créer une classe **Messagerie** pour tester les **CRUD** dans le package « ma.projet.test », dans cette classe créer 3 employés :
   - Le 1er employé envoie un message aux deux autres ;
   - Le 2ème employé envoie un message aux deux autres ;
   - Afficher les messages reçus par le 3ème employé.

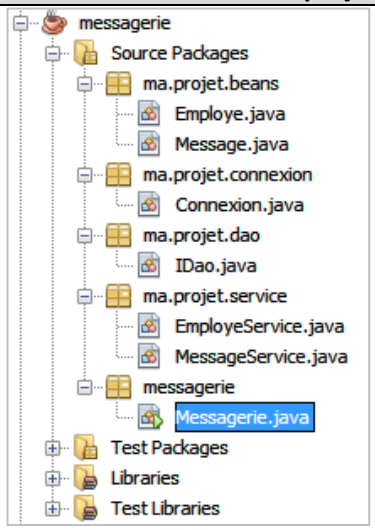
Teste Employe

```
EmployeService es  = new EmployeService();
//Création des employes
es.create(new Employe("LACHGAR", "Mohamed"));
es.create(new Employe("RAMI", "ALI"));
es.create(new Employe("SAFI", "Fatima"));

//Modification d'un employe
Employe e = es.getById(3);
if (e != null) {
       e.setNom("ALAOUI");
       e.setPrenom("Manale");
       es.update(e);
}

//Suppression d'un employe
es.delete(es.getById(4));

//Liste des employes
for(Employe emp : es.getAll())
      System.out.println(""+emp.getNom());
```

Teste Message

```
EmployeService es = new EmployeService();
MessageService ms = new MessageService();

ms.create(new Message("Réunion", "Réunion de fin d'année", new Date(), es.getById(1), es.getById(2)));
ms.create(new Message("Réunion", "Réunion de fin d'année", new Date(), es.getById(1), es.getById(3)));

ms.create(new Message("Stage", "Stage à Marrakech", new Date(), es.getById(2), es.getById(1)));
ms.create(new Message("Stage", "Stage à Marrakech", new Date(), es.getById(2), es.getById(3)));

//Les message reçus par l'employé 3
for (Message m : ms.getAll()) {
     if(m.getEmpRecepteur().getId() == 3)
           System.out.println(""+m.getSujet());

}
```

Structure finale de votre projet



Script de la base de données

```
CREATE TABLE IF NOT EXISTS `employe` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nom` varchar(50) NOT NULL,
  `prenom` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1
AUTO_INCREMENT=5 ;


CREATE TABLE IF NOT EXISTS `message` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `objet` varchar(50) NOT NULL,
  `sujet` varchar(500) NOT NULL,
  `date` date NOT NULL,
  `idE` int(11) NOT NULL,
  `idR` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idE` (`idE`,`idR`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1
AUTO_INCREMENT=18 ;
```