



Rapport du projet Médiathèque JavaEE

Membre du groupe

Victor TRUONG 207

Albéric CUSIN 207

Installation

Nécessaire :

- MySQL(préférence) ou Oracle
- TomcatServer

Configuration base de donné :

- MySQL

username : root

password : ""

databasename : jee

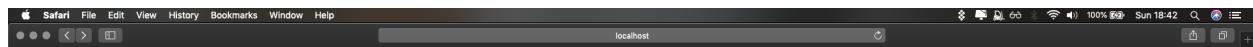
script : jee.sql

- TomcatServer

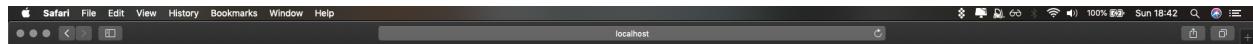
warfile : PROJET.war

Exemple d'utilisation de l'application

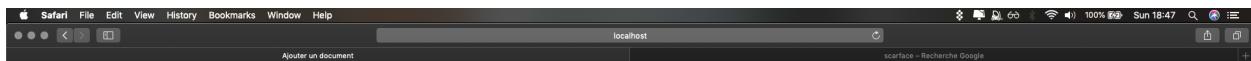
1) Connexion Bibliothécaire (username: oui, password: non)



2) Interface Bibliothécaire



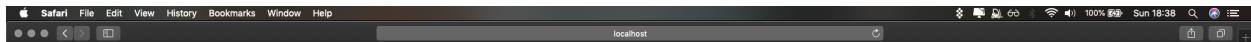
3) Ajouter un document (Scarface - Brian de Palma)



BIENVENUE OUI

Scarface
Brian de Palma
https://lh3.googleusercontent.com/pr...
✓ DVD Live
AJOUTER UN DOCUMENT

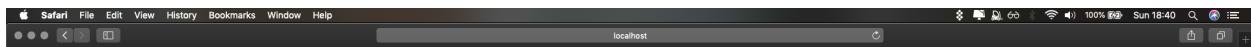
1) Connexion Abonné (username : cusin,password : cusin)



BIENVENUE À LA BIBLIOTHÈQUE

cusin
.....
SE CONNECTER

2) Interface Abonné



BIENVENUE CUSIN

Reserver un document

Emprunter un document

Rendre un document

3) Emprunter un document

Emprunter un Document

localhost

scarface - Recherche Google

The Social Network

PAS AVOIR DES MILLIONS D'AMIS
SAUF QUELQUES QUELQUES ENNEMIS

the social network

Titre : The Social Network par David Fincher

Type : DVD

Emprunter

Scarface

AL PACINO SCARFACE

ROBERT DE NIRO
MELISSA BRENNER
DONALD SUTHERLAND
ROBERT DUVALL
ROBERT DUVALL
JOHN GOODMAN
JAMES CAAN
JOHN GOODMAN
ROBERT DUVALL
JOHN GOODMAN

Titre : Scarface par Brian de Palma

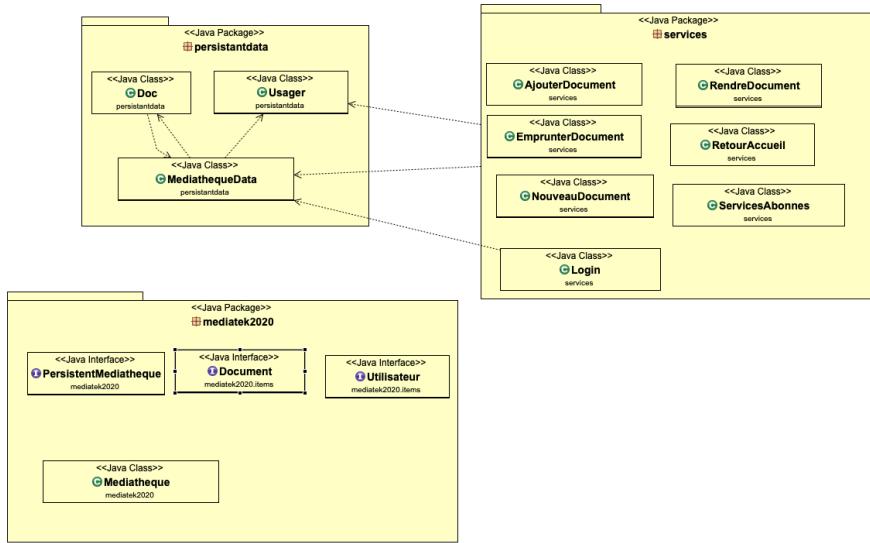
Type : DVD

Emprunter

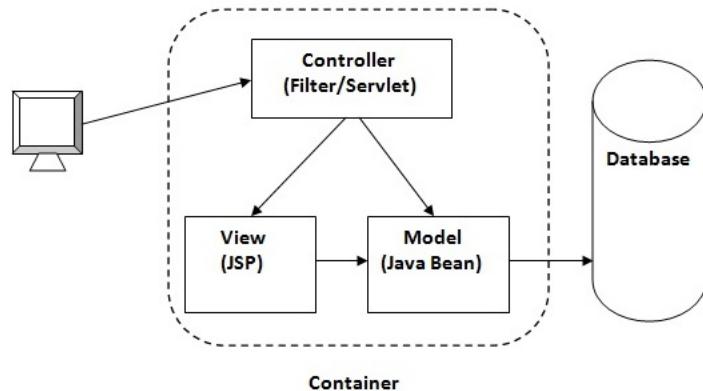
4) Rendre un document



Structuration de code, découplage et injection de dépendance



Notre application respecte une architecture MVC.



Model : se situe dans la package "persistantdata", où se trouve les objets, les classes, et MediathequeData qui permet de modifier la base de donnée.

View : se situe dans le package "webcontent", où se trouve nos JSP dont le contenu peut être modifié par les servlets.

Controller : nos servlets qui se situent dans le package "services".

Utilisation de Servlet et/ou JSP pour l'implémentation du module de service

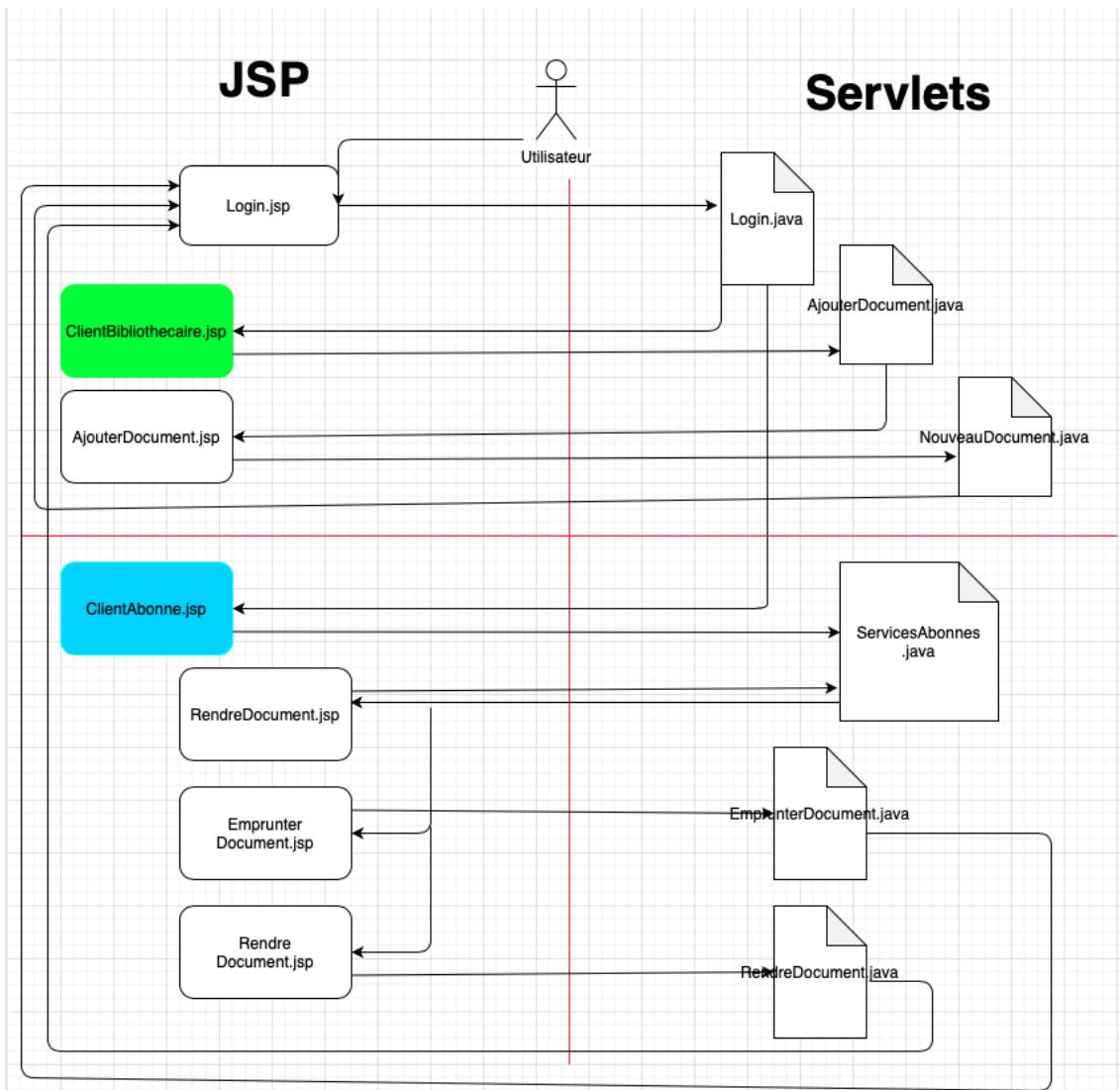
Notre WebApps utilise des Servlets, et des JSP.

Etapes du déclenchement des Servlets et des JSP :

- 1) Les JSP ont des formulaires HTML qui appellent des servlets "services" par une méthode POST.
- 2) Les servlets récuperent les paramètres d'entrées de l'utilisateur par le "DoPost" et modifient les variables de sessions, ou request, et peuvent aussi rediriger l'utilisateur vers une autre page.
- 3) Les JSP une fois l'action des servlets effectués, peuvent récupérer les variables sessions ou request et peuvent modifier le DOM HTML.

Shéma des JSP et Servlets :

les flèches de JSP vers SERVLET sont des requêtes DOPOST
et les flèches de SERVLET vers JSP sont des getRequestDispatcher



Transformation objet-relationnel : comment passe-t-on des objets de l'application Document et Utilisateur aux tables relationnelles de la base de données ?

Nous passons de l'objet de l'application Document et Utilisateur aux tables relationnelles de la base de données grâce à la distinction entre 'MédiathèqueData' et 'Médiathèque'.

Médiathèque ⇒ objet de l'application

MédiathèqueData ⇒ base de données

persistence.MediathequeData s'auto-déclare à mediatheque.PersistantMediatheque par injection de dépendance en appelant setData dans son bloc static.

Nous allons prendre l'exemple de la création d'un nouveau document :

```
//La Servlet NouveauDocument.java
```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String titre = request.getParameter("titre");
    String auteur = request.getParameter("auteur");
    String couverture = request.getParameter("couverture");
    String statut = request.getParameter("statut");

    Mediatheque m = Mediatheque.getInstance();
    m.nouveauDocument(Integer.parseInt(statut), titre, auteur, couverture);
    this.getServletContext().getRequestDispatcher("/ClientBibliotheque.jsp").forward(request, response);

}

```

L'objet Médiathèque à l'appel de nouveauDocument() exécute cette méthode dans Médiathèque et par délégation aussi à MédiathèqueData.

```

//La Class MediathequeData.java
@Override
public void nouveauDocument(int type, Object... args) {
    Connection co = connection();
    String titre = "";
    String auteur = "";
    String couverture = "";
    String reqUser = "";
    PreparedStatement state;
    switch (type) {
        case TYPE_DVD:
            titre = (String) args[0];
            auteur = (String) args[1];
            couverture = (String) args[2];
            reqUser = "Insert into document(Titre,Auteur,Disponible,Couverture,statut,pseudo) values (?, ?,?, ?, 'DVD', 'default')";
            try {
                state = co.prepareStatement(reqUser);
                state.setString(1, titre);
                state.setString(2, auteur);
                state.setString(3, couverture);
                state.executeUpdate();

            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            break;
        case TYPE_LIVRE:
            titre = (String) args[0];
            auteur = (String) args[1];
            couverture = (String) args[2];
            reqUser = "Insert into document(Id,Titre,Auteur,Disponible,Couverture,statut,pseudo) values (SEQ_DOCUMENT.nextVal,?, ?,?, ?, 'Li";
            try {
                state = co.prepareStatement(reqUser);
                state.setString(1, titre);
                state.setString(2, auteur);
                state.setString(3, couverture);
                state.executeQuery();

            } catch (SQLException e) {
                e.printStackTrace();
            }
            break;
    }
    try {
        co.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

Variables sessions : ouverture, maintien et fermeture de la session de travail des utilisateurs ?

Notre seul est unique variable de session est "username".

Nous ouvrons notre session dès lorsque l'utilisateur a réussi à s'identifier et est reconnue par la librairie.

```
//servlet login
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    String identifiant = request.getParameter("identifiant");
    String motDePasse = request.getParameter("motDePasse");

    try {
        Class.forName(MediathequeData.class.getName());
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Mediatheque m = Mediatheque.getInstance();
    Utilisateur u = m.getUser(identifiant, motDePasse);
    boolean isBiblio;
    if(u == null) {
        this.getServletContext().getRequestDispatcher("/LoginInvalide.jsp").forward(request, response);
    }else {
        //ouverture de session //
        HttpSession session = request.getSession(true);
        //ouverture de session //

        isBiblio = u.isBibliothecaire();

        //attribution d'une variable username //
        session.setAttribute("username", identifiant);
        //attribution d'une variable username //

        if(isBiblio) {

            this.getServletContext().getRequestDispatcher("/ClientBibliothecaire.jsp").forward(request, response);
        } else {

            this.getServletContext().getRequestDispatcher("/ClientAbonne.jsp").forward(request, response);

        }
    }
}
```

Notre variable de session "username" va servir à relier les emprunts avec le nom des utilisateurs.

La session sera fermé une fois que l'utilisateur aura effectué son emprunt ou son retour.

```
//Servlet EmprunterDocument.java

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    HttpSession session = request.getSession(true);
    Mediatheque m = Mediatheque.getInstance();

    try {
        m.emprunter(m.getDocument(Integer.parseInt(request.getParameter("iddoc"))), (Utilisateur)session.getAttribute("userConnecte"))
        session.invalidate();
    } catch (NumberFormatException | EmpruntException e) {
        //cloture de la session //
        session.invalidate();
        //cloture de la session //
        e.printStackTrace();
    }
    this.getServletContext().getRequestDispatcher("/Login.jsp").forward(request, response);
}
```

Concurrence : identification des ressources critiques et la gestion java de la concurrence ?

La ressource critique sont les documents. Il ne faudrait pas que deux usagers puissent emprunter le même documents en même temps.

C'est pour cela qui faut utilisé la méthode "synchronized()" pour ne pas avoir des problèmes de concurrence.

```
//Class doc dans le package persstantdata
public class Doc implements Document {

    ...

    @Override
    public void emprunter(Utilisateur user) throws EmpruntException {
        synchronized(this) {
            if (disponible && this.pseudo.equals("default")) {
                this.pseudo = user.name();
                MediathequeData.emprunter(this, user);
                this.disponible = false;
            } else {
                if (user.name().equals(this.pseudo)) {
                    this.disponible = false;
                    MediathequeData.emprunter(this, user);
                }
                else
                    throw new EmpruntException();
            }
        }
    }

    @Override
    public void rendre(Utilisateur user) throws RetourException {
        synchronized(this) {
            if (user.name().equals(this.pseudo) && !this.disponible) {
                this.pseudo = "default";
                this.disponible = true;
                MediathequeData.rendre(this, user);
            }
            else
                throw new RetourException();
        }
    }

    @Override
    public void reserver(Utilisateur user) throws ReservationException {
        synchronized(this) {
            if(disponible && pseudo.isEmpty())
                this.pseudo = user.name();
            else
                throw new ReservationException();
        }
    }
}
```

Efficacité des requêtes d'accès à la base de données : gestion de la connexion à la base de données et de la précompilation des requêtes

Nous nous connectons à une base de donnée MySQL :

```
public class MediathequeData implements PersistentMediatheque {
    ...

    private static Connection connection() {
        try {
            try {
                Class.forName("com.mysql.jdbc.Driver");
```

```

} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
Connection co = DriverManager.getConnection("jdbc:mysql://localhost:3306/jee", "root", "12345678");
return co;
} catch (SQLException e) {
    e.printStackTrace();
}
return null;
}
...
}

```

Nous précompilons évidemment nos requêtes pour éviter toutes injections sql :

```

public class MediathequeData implements PersistentMediatheque {
    ...
    // renvoie la liste de tous les documents de la bibliothèque
    @Override
    public List<Document> tousLesDocuments() {
        try {
            Connection co = connection();
            String requUser = "Select * from document";
            PreparedStatement state = co.prepareStatement(requUser);
            ResultSet user = state.executeQuery();
            int id;
            String titre;
            String auteur;
            boolean disponible;
            String couverture;
            String statut;
            String pseudo;
            List<Document> documents = new ArrayList<>();
            while (user.next()) {
                id = user.getInt(1);
                titre = user.getString(2);
                auteur = user.getString(3);
                disponible = (user.getInt("Disponible") == 0 ? true : false);
                couverture = user.getString(5);
                statut = user.getString(6);
                pseudo = user.getString(7);

                documents.add(new Doc(id, titre, auteur, disponible, couverture, statut, pseudo));
            }
            state.close();
            user.close();
            co.close();
            return documents;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }

    // va récupérer le User dans la BD et le renvoie
    // si pas trouvé, renvoie null
    @Override
    public Utilisateur getUser(String login, String password) {
        try {
            Connection co = connection();
            String requUser = "Select * from utilisateur where pseudo = ? and motdepasse = ?";

            PreparedStatement state = co.prepareStatement(requUser);
            state.setString(1, login);
            state.setString(2, password);
            ResultSet user = state.executeQuery();
            String log = "";
            String pass = "";
            String statut = "";
            while (user.next()) {
                log = user.getString("pseudo");
                pass = user.getString("motdepasse");
                statut = user.getString("statut");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }
}

```

```

        }
        if (statut.equals("default"))
            statut = "";
        co.close();
        if (log.isEmpty() || pass.isEmpty() || statut.isEmpty())
            return null;
        return new Usager(log, pass, statut);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}

// va recuperer le document de numero numDocument dans la BD
// et le renvoie
// si pas trouve, renvoie null
@Override
public Document getDocument(int numDocument) {
    try {
        Connection co = connection();
        String reqUser = "Select * from document where id = ?";
        PreparedStatement state = co.prepareStatement(reqUser);
        state.setInt(1, numDocument);
        ResultSet user = state.executeQuery();

        int id = -1;
        String titre = "";
        String auteur = "";
        boolean disponible = false;
        String couverture = "";
        String statut = "";
        String pseudo = "";
        while (user.next()) {
            id = user.getInt(1);
            titre = user.getString(2);
            auteur = user.getString(3);
            disponible = (user.getInt(4) == 0 ? true : false);
            couverture = user.getString(5);
            statut = user.getString(6);
            pseudo = user.getString(7);

        }
        state.close();
        user.close();
        co.close();
        return new Doc(id, titre, auteur, disponible, couverture, statut, pseudo);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}

@Override
public void nouveauDocument(int type, Object... args) {
    Connection co = connection();
    String titre = "";
    String auteur = "";
    String couverture = "";
    String reqUser = "";
    PreparedStatement state;
    switch (type) {
    case TYPE_DVD:
        titre = (String) args[0];
        auteur = (String) args[1];
        couverture = (String) args[2];
        reqUser = "Insert into document(Titre,Auteur,Disponible,Couverture,statut,pseudo) values (?, ?,?, ?, 'DVD', 'default')";
        try {
            state = co.prepareStatement(reqUser);
            state.setString(1, titre);
            state.setString(2, auteur);
            state.setString(3, couverture);
            state.executeUpdate();

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

```

        break;
    case TYPE_LIVRE:
        titre = (String) args[0];
        auteur = (String) args[1];
        couverture = (String) args[2];
        reqUser = "Insert into document(Id,Titre,Auteur,Disponible,Couverture,statut,pseudo) values (SEQ_DOCUMENT.nextVal,?, ?,0,?, 'Li";
        try {
            state = co.prepareStatement(reqUser);
            state.setString(1, titre);
            state.setString(2, auteur);
            state.setString(3, couverture);
            state.executeQuery();
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
        break;
    }
    try {
        co.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public static void emprunter(Document d, Utilisateur u) {
    Connection co = connection();
    String req = "Update Document SET Pseudo = ?, Disponible = 1 Where Id = ? ";
    PreparedStatement state;
    try {
        state = co.prepareStatement(req);
        state.setString(1, u.name());
        state.setInt(2,(int) d.data()[0]);
        state.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void rendre(Document d, Utilisateur u) {
    Connection co = connection();
    String req = "Update Document SET Pseudo = 'default', Disponible = 0 Where Id = ? ";
    PreparedStatement state;
    try {
        state = co.prepareStatement(req);
        state.setInt(1,(int) d.data()[0]);
        state.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

Merci d'avoir examiné notre dossier ! Nous vous souhaitons une bonne santé et un bon confinement