Le Design Pattern MVC

P.Mathieu

IUT-A Lille
http://www.iut-a.univ-lille.fr
prenom.nom@univ-lille.fr

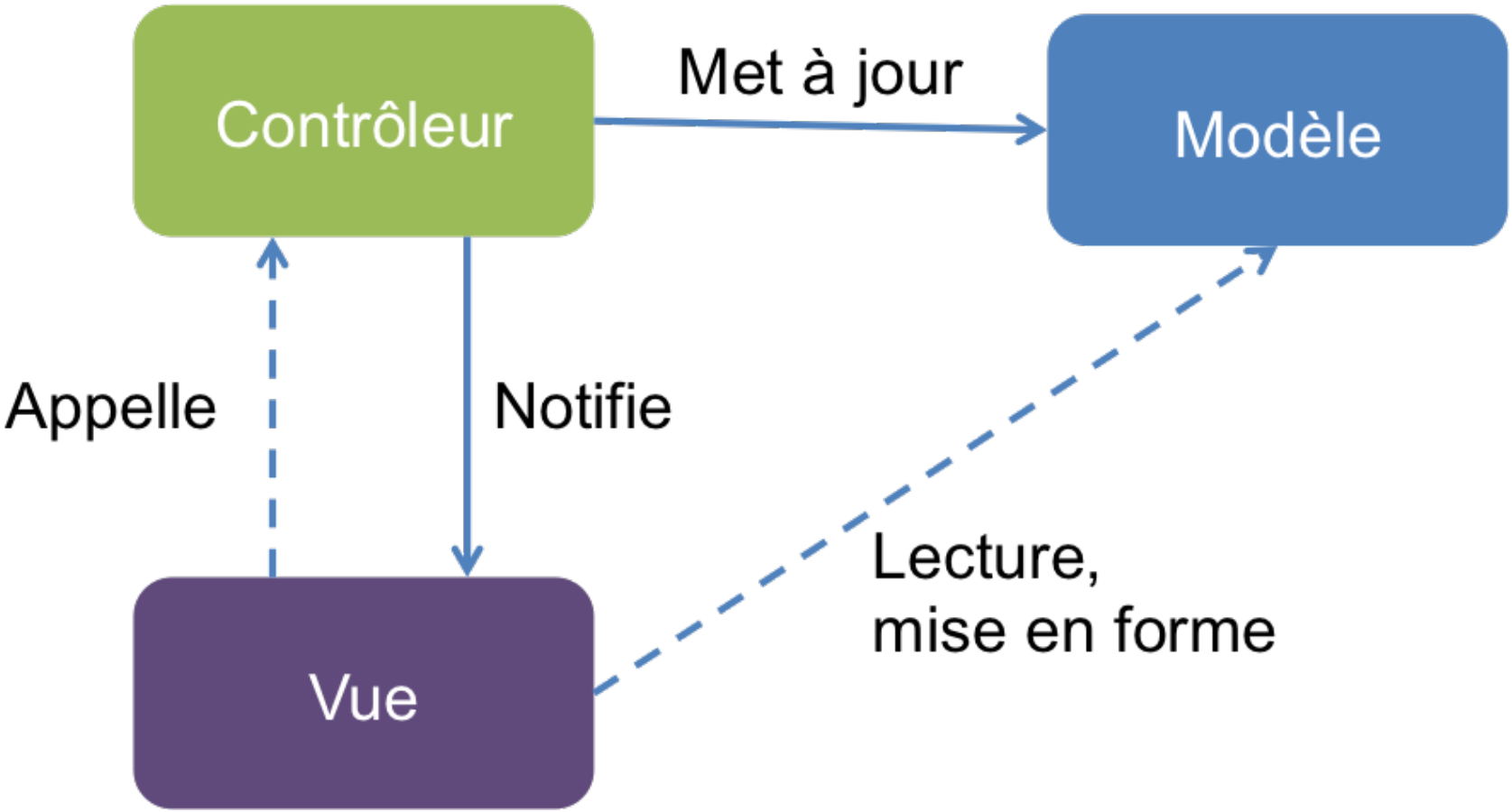
Principe

Modèle - Vue - Contrôleur (MVC)

- Séparer les préoccupations
  - la logique de traitement,
  - l'accès au données,
  - l'affichage
- Faciliter le travail des développeurs :
  - Découpage et répartition du travail,
  - Organisation du code pour la maintenance,
  - Indépendance des objets pour l'évolution de l'application

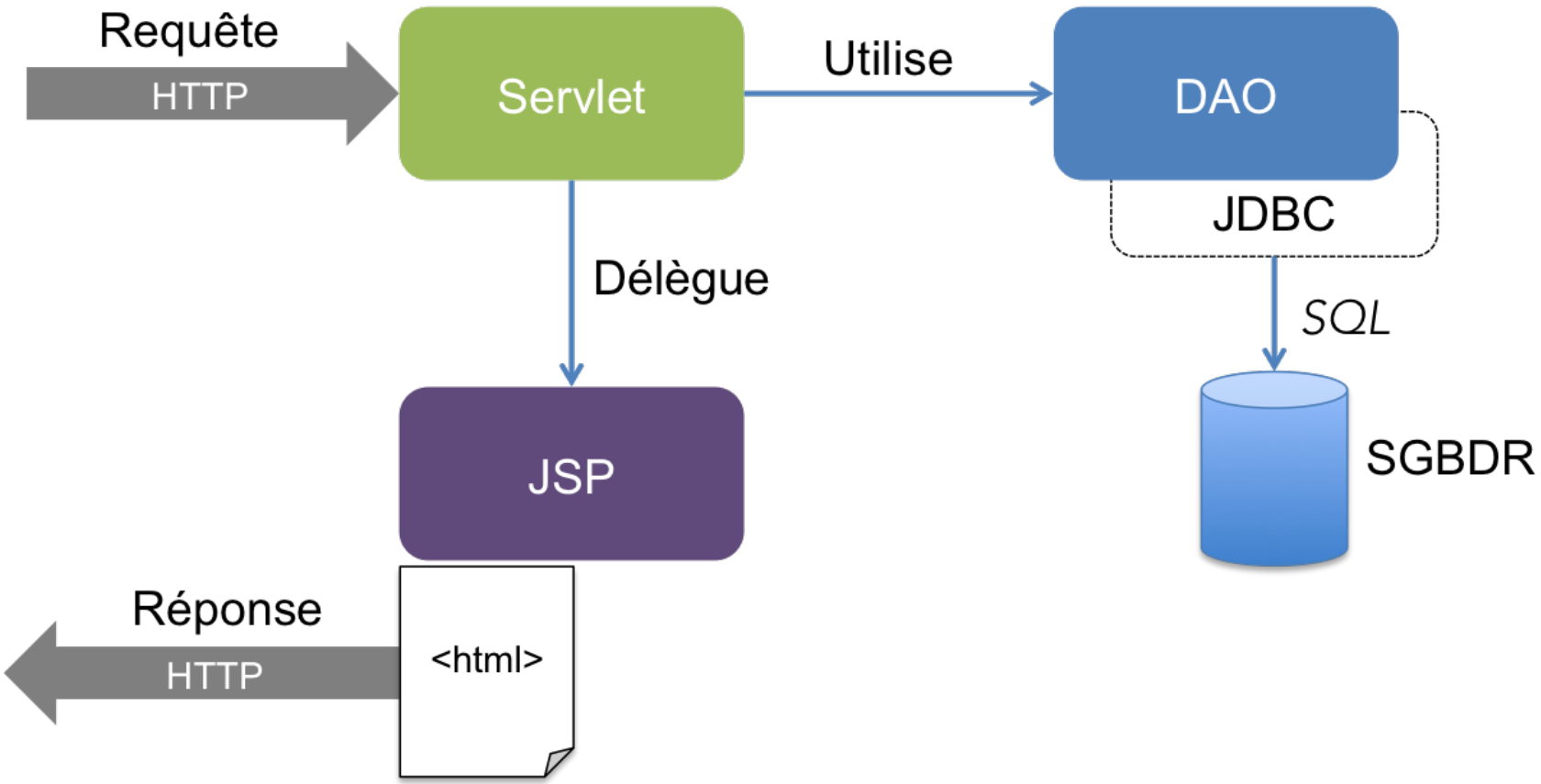
Principe

Pattern général MVC



Principe

Application du MVC au web JEE



Principe

Un exemple de mise en oeuvre

Actions :

- Ajouter un joueur
- Modifier un joueur existant
- Supprimer un joueur
- Afficher la liste des joueurs par club

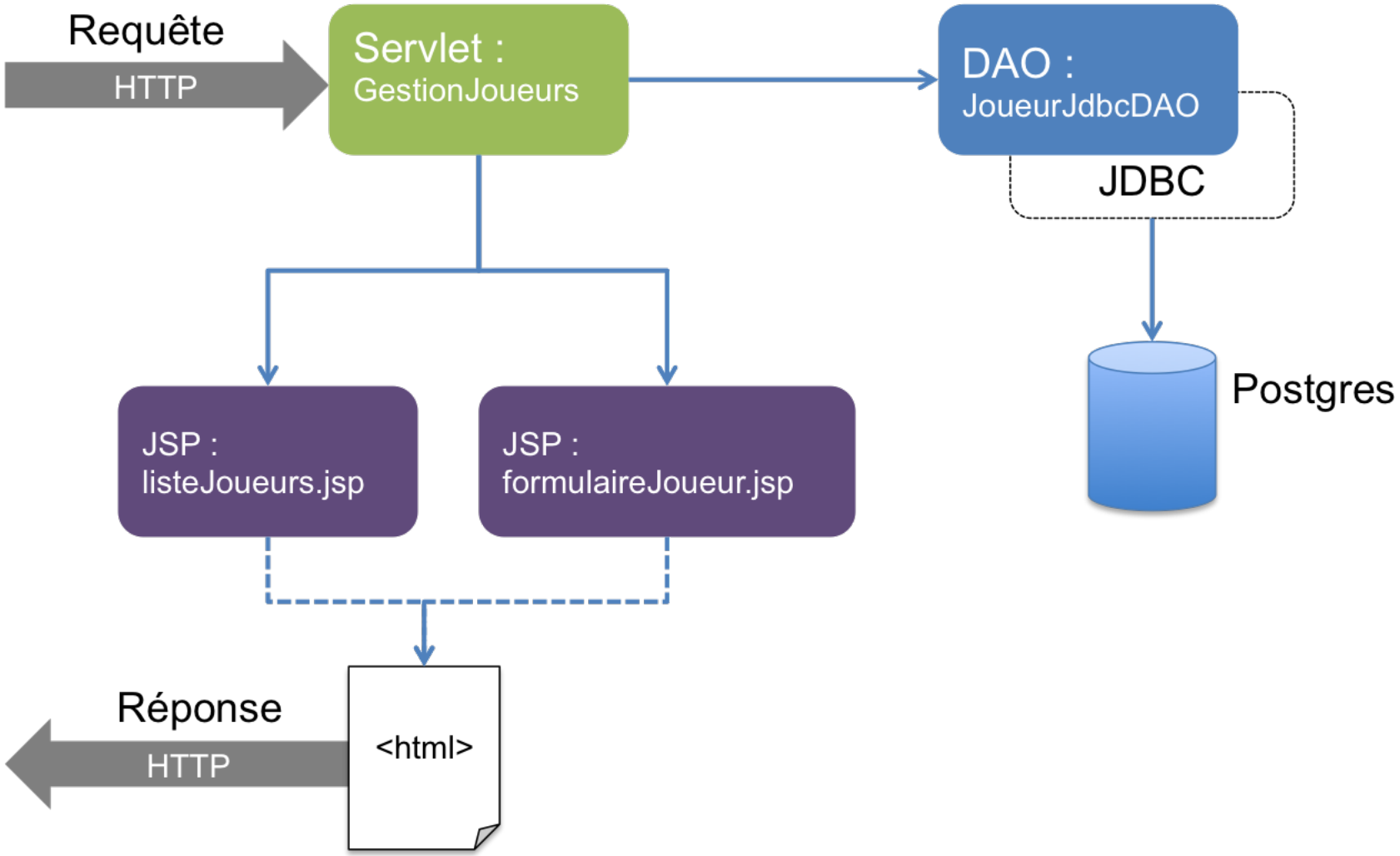
Définissent les entrées du Contrôleur :

`afficher`, `modifier`,`lister` ,..

Joueur
id
nom
prenom
poste
club
dateNaiss

Principe

Implémentation MVC correspondante



Principe

Contrôleur : La redirection

Université de Lille

- Problématique : le controleur récupère le modèle et le passe à la vue.
- Un outil privilégié : le `RequestDispatcher`
- Deux méthodes de redirection
  - Via le client
    `response.sendRedirect(url)`
    Crée un nouvel objet `HttpServletRequest`
  - Interne au serveur

```
RequestDispatcher rd=request.getRequestDispatcher("chemin");
rd.forward(request, res);
```

    Conserve le même objet `HttpServletRequest`
  - l'objet `HttpServletRequest` contient une hashtable accessible avec `getAttribute` et `setAttribute` !

Principe

Contrôleur : La structure

Université de Lille

```
public class MonControleur extends HTTPServlet
{
  public void service(....)
  {
    action = req.getParameter("action");
    String vue;
    switch(action)
    {
      case "a1" :
      // lecture des éventuels paramètres complémentaires
      // Interrogation du modèle (DAO) en fonction des params
      // Affectation d'une partie du modèle dans les attributs de la req
      req.setAttribute("cle",valeur);
      // requestDispatcher vers la vue
      vue="lavue.jsp"
      break;
      case "a2" :
      ......
      default :
    }
    req.getRequestDispatcher(vue).forward(req,res);
  }
}
```

Ecriture du DAO

Université de Lille

```
class JoueurJdbcDao
{
  JoueurJdbcDao(DS ds) { ... }
  public void create( Joueur joueur ) { ... }
  public void update( Joueur joueur ) { ... }
  public void delete( int idJoueur ) { ... }
  public Joueur find( int idJoueur ) { ... }
  public List<Joueur> findByClub( String club ) { ... }
}
```

Ecriture du contrôleur

```
@WebServlet("/GestionJoueurs")
public class GestionJoueurs extends HTTPServlet
{
  JoueurJdbcDao daoJoueur;

  public void service(...) {
    String action = req.getPamameter("act");
    String vue = "WEB-INF/formulaireJoueur.jsp";
    switch( action ) {
      case "lister":
          listerParClub(req); break;
          vue = "WEB-INF/listeJoueurs.jsp";
      case "editer": editer(req); break;
      case "modifier":
          if ( modifier(req) )
          { res.sendRedirect("GestionJoueurs?act=lister"); return;}
          break;
      default:
          res.sendError(404,"Action non supportée" ); return;
      }
      req.getRequestDispatcher(vue).forward(req,res); }
     // ...
 }
```

Détail du contrôleur

```
private void listerParClub( HttpServletRequest req )
{ String club = req.getParameter("club");
  if ( club == null ) club = "Lille";
  List<Joueur> joueurs = daoJoueur.findByClub(club);
  req.setAttribute( "joueurs", joueurs );
}

private boolean modifier(HttpServletRequest req )
{ try
  {
    Joueur j = joueurFromRequest( req );
    if ( j.id == null ) daoJoueur.create(club);
    else daoJoueur.update(club);
  }
  catch(Exception e )
  { req.setAttribute( "error", e );
    req.setAttribute( "joueur", j );
    return false;
  }
  return true;
}
```

Ecriture de la vue

```
<% Joueur j = (Joueur) request.getAttribute("joueur"); %>
<h1><%= (j == null ? "Ajouter" : "Modifier") %> un joueur.</h1>
<form action=GestionJoueurs method=POST>
  <input type='hidden' name='act' value='modifier'>
  <% if ( j != null ) { %>
        Id : <%= j.id %>
        <input type='hidden' name='id' value='<%= j.id %>'/> <br/>
  <% } %>
  Nom    :  <input type='text' name='nom'    value='<%= j.nom %>' />
  Prenom :  <input type='text' name='prenom' value='<%= j.prenom %>' />
  // ...
  <input type='submit' value='Enregistrer' />
</form>
```

En résumé

- MVC est un patron de conception couramment utilisé pour faciliter le développement d'applications.
- Sa mise en oeuvre web peut être implémenté en JEE par :
  - Une servlet qui orchestre un ensemble de traitements
  - Un ou plusieurs DAO pour manipuler le modèle, généralement via JDBC.
  - Des JSP qui mettent en forme le modèle et permettent à l'utilisateur d'effectuer des actions.