Master 3IR
TP AIR #1
Introduction au développement d'applications Web en JAVA EE.

Prérequis :

- HTML (vue en cours)
- CSS (vue en cours)
- JAVA/JEE (vue en cours)
- Base de données et langage SQL (vue en cours)

Nouvelles notions abordées :

- Serveur d'application TOMCAT
- Serveur bdd PostgesSQL (langage SQL)

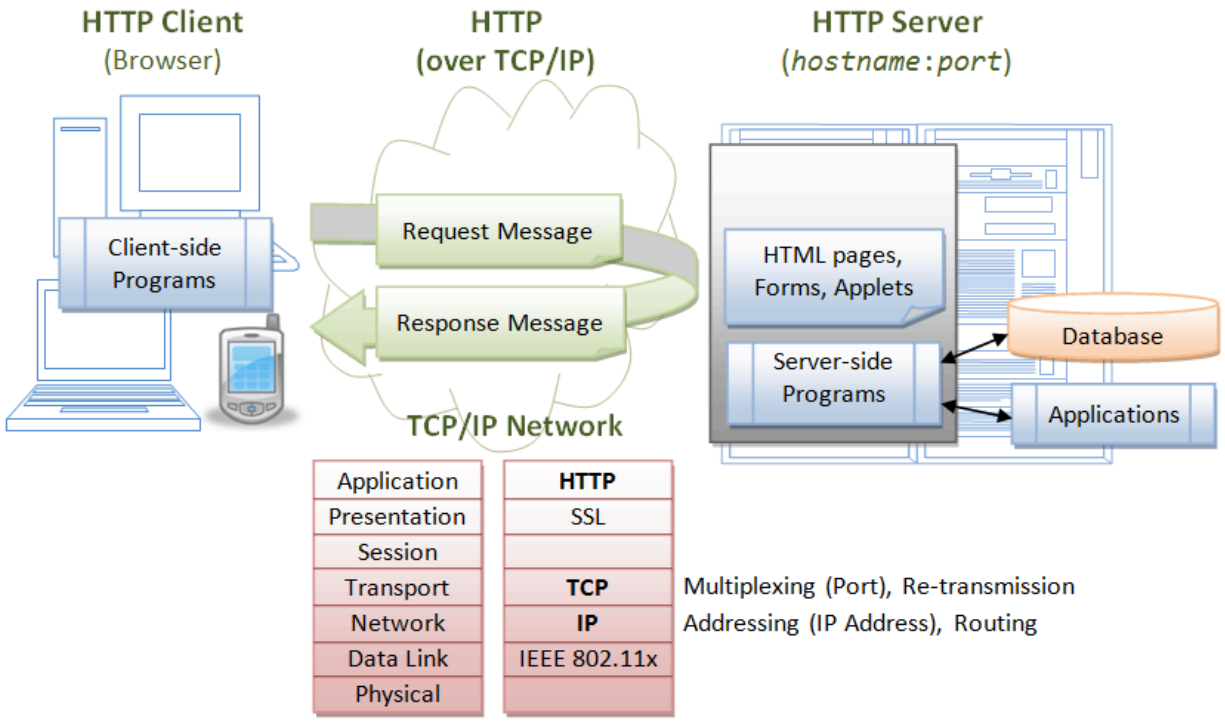


Plate-forme de développement :

a. Eclipse : https://eclipse.org/downloads
b. Tomcat : http://tomcat.apache.org
c. PostgreSQL : http://www.postgresql.org/download/

d. Maven : https://maven.apache.org/download.cgi

Exercice 1 : Environnement de développement

1. Installation et configuration de Tomcat
2. Installation et configuration de Maven
3. Installation et configuration de PostgreSQL

L'ensemble des applications suivantes devront être "**packager**" avec « maven ».

Exercice 2 : Prise en main de JAVA EE : Servlet & JSP (HELLO THE WORLD)

Créer une application qui affiche « Hello the world » à l'écran

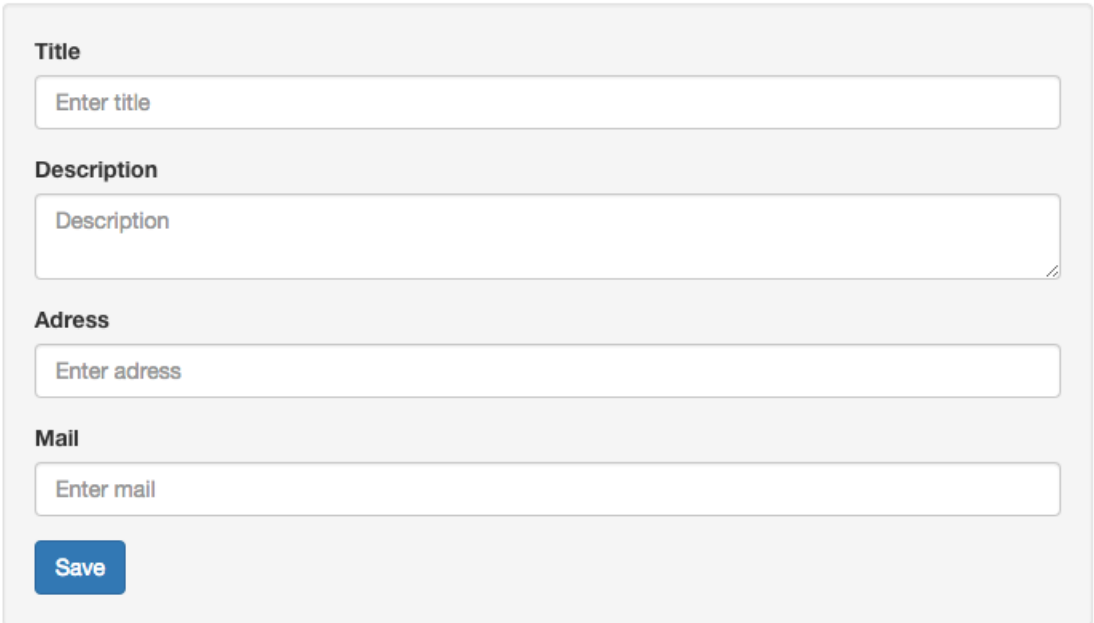
Exercice 3 : Passage de paramètre

Ajouter un formulaire permettant la saisie de votre nom. L'application devra afficher « Hello the World » suivit de votre nom.

Exercice 4 : Connexion à une base de données

1. Créer une base de données "**MasterAnnonce**"
2. Créer la table "**annonce**" avec les colonnes
   a. title (varchar[64])
   b. description (varchar[256])
   c. adress (varchar[64])
   d. mail (varchar[64])
   e. date (timestamp)
3. Utiliser les instructions « INSERT, UPDATE, DELETE » pour insérer et manipuler des données dans la table.

Exercice 5 : Création d'une application



Formulaire de saisie

a. Dans le répertoire “WebContent”, écrire une première JSP “AnnonceAdd.jsp” affichant un formulaire de saisie avec les champs suivants
    - title [text]
    - description [textarea]
    - adress [text]
    - mail [mail]

   Le formulaire utilise la méthode “POST” `<form method="post" action="">`
b. Ecrire une première Servlet “AnnonceAdd.java” affichant la JSP précédente

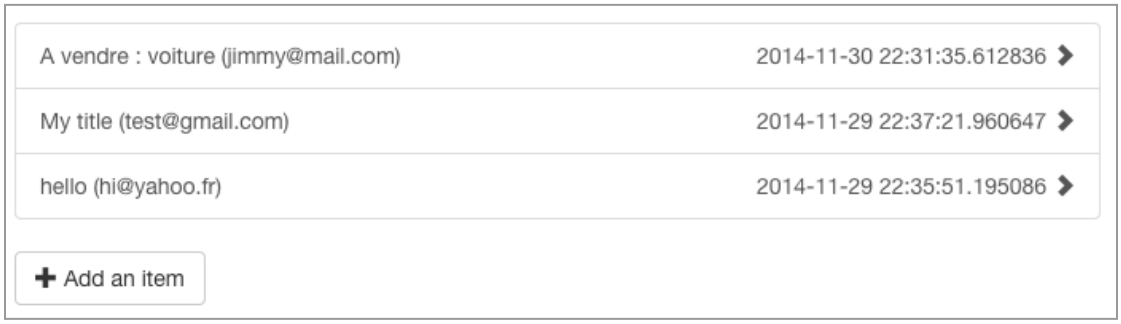
4. DAO/SQL

a. Dans le répertoire “Java Resources/src”, écrire la classe de connection à la base de données “ConnectionDB.java” disponible en annexe (page 4)
b. Écrire la DAO permettant de gérer le CRUD d’une annonce
    - DAO.java de type **abstract** définissant les méthodes à implémenter par AnnonceDAO.java
    - AnnonceDAO.java **étendant** la classe DAO.java

5. Saisie d’une annonce

a. Dans la servlet “AnnonceAdd.java” permettant d’afficher le formulaire de saisie
    - La méthode **doGet()** permet d’afficher la JSP “AnnonceAdd.jsp”
b. Ecrire la méthode permettant d'enregistrer l’annonce soumise par le formulaire
    - Écrire dans la méthode **doPost()** le code validant le formulaire (tous les champs sont obligatoires) et le code insérant une annonce en base de données

6. Afficher la liste des annonces disponibles



Liste des annonces

a. La Servlet “AnnonceList.java” recherche en base toutes les annonces
b. et affiche la JSP “AnnonceList.jsp”

7. Mise à jour d’une annonce

a. Servlet “AnnonceUpdate.java” avec récupération du parametre “id” de l’URL
    - exemple : http://localhost:8080/MasterAnnonce/AnnonceUpdate?id=123

b. JSP “AnnonceUpdate.jsp” affichant un formulaire valorisé avec les détails de l’annonce

8. **Suppression d’une annonce**
    a. Servlet “AnnonceDelete.java” avec récupération du parametre “id” de l’URL
        - exemple : http://localhost:8080/MasterAnnonce/AnnonceUpdate?id=123
    b. JSP “AnnonceUpdate.jsp” affichant un formulaire valorisé avec les détails de

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionDB {

	private String url = "jdbc:postgresql://localhost:5433/MasterAnnonce";
	private String user = "postgres";
	private String passwd = "password13";
	/**
	 * Objet Connection
	 */
	private static Connection connect;

	/**
	 * Constructeur privé
	 * @throws ClassNotFoundException
	 */
	private ConnectionDB() throws ClassNotFoundException{
		try {
			Class.forName("org.postgresql.Driver");
			connect = DriverManager.getConnection(url, user, passwd);
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	/**
	 * Methode qui va nous retourner notre instance
	 * et la creer si elle n'existe pas...
	 * @return
	 * @throws ClassNotFoundException
	 */
	public static Connection getInstance() throws ClassNotFoundException{
		if(connect == null){
			new ConnectionDB();
		}
		return connect;
	}
}
```

ConnectionDB.java