Université Hassan 1er

Faculté des Sciences et Techniques Settat
Licence Sciences et Techniques : Génie Informatique

GESTION DE RÉSERVATION D'HÔTEL

Projet

Encadré par : Mr EZZATI Abdellah

Réalisé par : ISSAM Abderrahmane

BOUSLIKHANE Chaimae
MAFTOUH Jihane
NAMIR Manal

Année Universitaire :
2018/2019

Sommaire :

I-Introduction :

L'ordinateur, cette machine électrique qui depuis sa création en 1938 a révolutionné la vie humaine, réunie avec le réseau internet, ce duo est devenu le véritable secours de l'Homme lui permettant de vivre dans l'aisance la plus suprême. À priori les lourdes tâches jadis complexes et difficiles ou quasi impossible à réaliser sont devenues exécutables en un clin d'œil et si c'est à citer nul ne peut nier l'exactitude et la fiabilité de ses résultats. Une telle machine qui fournit un travail sans fatigue ni stresse ni oublie ou lenteur est le remède de nos problèmes humains sous sa forme la plus simple. Par ailleurs il est bien clair que l'ordinateur et l'informatique ont métamorphosé l'humanité moderne que ça soit sur le niveau personnel ou professionnel quant à ce dernier et dans la présence de plusieurs facteurs notamment la croissance des demandes et des exigences des clients ou de l'entreprise. La complexité des activités, la concurrence sur le marché, en termes de gestion Administrer ou gestionner une entreprise en pleine expansion peut être le sujet d'une tache très délicate, c'est pour cela que l'informatique se présente en tête de liste apportant grâce à ces logiciels et ses outils la meilleure solution qui puisse exister. Au font le sujet de notre travail portera sur la gestion d'un hôtel précisément l'inscription et la réservation des clients.

II-Définitions :

1-Servlet :

Une **servlet** est une **Classe Java** qui permet de créer dynamiquement des données au sein d'un **serveur Http** Ces données sont le plus généralement présentées au **format HTML**, mais elles peuvent également l'être au format **XML** ou tout autre format destiné aux **navigateurs web.** Les servlets utilisent l'API **Java Servlet** (***package javax.servlet***).

Une servlet s'exécute dynamiquement sur le serveur web et permet l'extension des fonctions de ce dernier, par exemple : l'accès à des **bases de données,** transactions de **commerce en ligne**, etc. Une servlet peut être chargé automatiquement lors du démarrage du serveur web ou lors de la première requête du client. Une fois chargés, les servlets restent actifs dans l'attente d'autres requêtes du client.

L'utilisation de servlets se fait par le biais d'un **conteneur de servlet** (framework) côté serveur. Celui-ci constitue l'environnement d'exécution de la servlet et lui permet de persister entre les requêtes des clients. L'API définit les relations entre le conteneur et la servlet. Le conteneur reçoit la requête du client, et sélectionne la servlet qui aura à la traiter. Le conteneur fournit également tout un ensemble de services standards pour simplifier la gestion des requêtes et des sessions.

2-JavaServer Pages :

Le **JavaServer Pages** ou **JSP** est une **technique** basée sur **Java** qui permet aux **développeurs** de créer dynamiquement du code **HTML**, **XML** ou tout autre type de **page web**. Cette technique permet au code Java et à certaines actions prédéfinies d'être ajoutés dans un contenu statique. Depuis la version 2.0 des spécifications, la syntaxe JSP est complètement conforme au standard **XML**.

La syntaxe du JSP ajoute des **balises** XML, appelées *actions JSP*, qui peuvent être utilisées pour appeler des **fonctions**. De plus, cette technique permet la création de **bibliothèques** de balises JSP (*taglib*) qui agit comme des extensions au HTML ou au XML. Les bibliothèques de balises offrent une méthode indépendante de la **plate-forme** pour étendre les fonctionnalités d'un **serveur HTTP**. Il existe aussi un langage de script particulier, appelé **Expression Language** (EL) destiné à réduire l'injection de code java au sein des pages JSP ainsi qu'à étendre les possibilités des taglibs, tel que la **JSTL**.

3-JavaBeans :

JavaBeans est une technologie de **composants logiciels** écrits en langage **Java**.

La spécification JavaBeans de **Oracle** définit les composants de type JavaBeans comme « des composants logiciels réutilisables manipulables visuellement dans un outil de conception ».

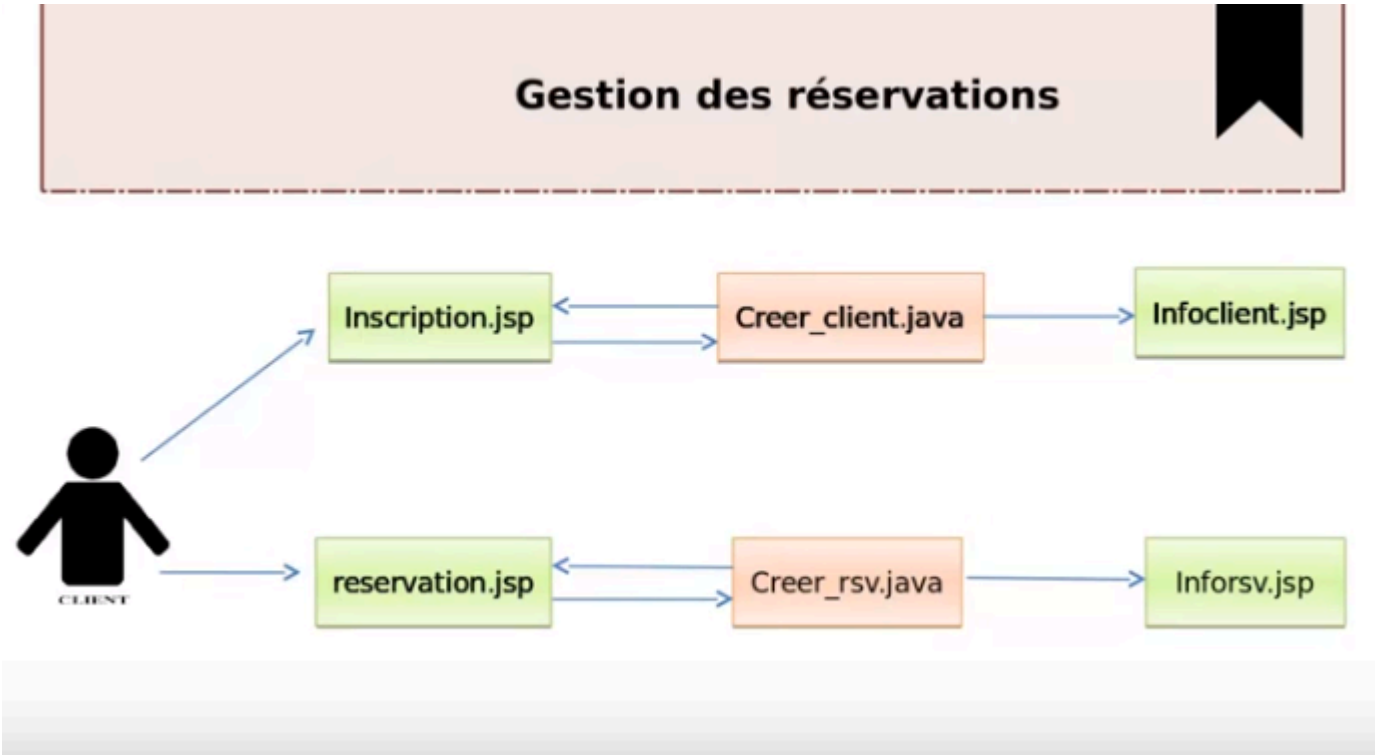
Ils sont utilisés pour encapsuler plusieurs objets dans un seul objet : le « **bean** » (ou haricot en français). Le « **bean** » regroupe alors tous les attributs des objets encapsulés, et peut définir d'autres attributs si besoin. Ainsi, il représente une entité plus globale que les objets encapsulés de manière à répondre à un besoin métier.

4-HTML :

L'***HyperText Markup Language***, généralement abrégé **HTML**, est le ***langage de balisage*** conçu pour représenter les ***pages web***. C'est un langage permettant d'écrire de l'***hypertexte***, d'où son nom. HTML permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ***ressources multimédias*** dont des ***images***, des formulaires de saisie et des programmes informatiques. Il permet de créer des documents ***interopérables*** avec des équipements très variés de manière conforme aux exigences de l'***accessibilité du web***. Il est souvent utilisé conjointement avec le ***langage de programmation JavaScript*** et des ***feuilles de style en cascade*** (CSS).

III-Analyse et Conception :

1-Description de l'Application :



- La gestion hôtelière est une vitalité indispensable dans le déroulement des activités normale d'un hôtel et qui dit hôtel dit réservation, et afin d'améliorer l'expérience client tout en offrant aux agents hôtelier une gestion de réservation plus facile, nous proposons cette application qui leur sera d'une grande utilité conçue grâce à la plate-forme ***JEE (Java Platform, Enterprise Edition)***, ainsi que les pages ***JSP*** et les ***servlets***.

  D'abord en ce qui concerne l'inscription le client s'inscrit en remplissant tous les champs demandés dans la page JSP ***inscription.jsp*** (Nom, Prénom, Téléphone, E-mail) ensuite les informations seront envoyées à une servlet qui s'appelle ***creer_client*** ,cette dernière renvoie les données qu'elle a récupérés à ***inscription.jsp*** dans le cas où le client a laissé un champ vide en lui affichant un message d'erreur , dans le cas contraire la page ***infoclient.jsp*** affiche toute les information récupérée en confirmant l'inscription du client.

  Pour ce qui est de réservation , dans la page ***reservation.jsp*** le client aura un formulaire à remplir par des informations relatives au client lui-même ainsi que d'autre informations qui concernent la réservation notamment le type de réservation son prix … ces dites informations seront envoyées de même à la servlet ***creer_rsv*** qui va refaire le même traitement que ***creer_client*** , idem pour ***info_rsv*** qui affiche les informations du client et de la réservation.

2-Les paquetages et les méthodes :

- Le package javax.servlet définit plusieurs interfaces, classes et exceptions :

javax.servlet	Nom	Rôle
Les interfaces	RequestDispatcher	Définition d'un objet qui permet le renvoi d'une requête vers une autre ressource du serveur (une autre servlet, une JSP ...)
	Servlet	Définition de base d'une servlet
	ServletConfig	Définition d'un objet pour configurer la servlet
	ServletContext	Définition d'un objet pour obtenir des informations sur le contexte d'execution de la servlet
	ServletRequest	Définition d'un objet contenant la requête du client
	ServletResponse	Définition d'un objet qui contient la reponse renvoyée par la servlet
	SingleThreadModel	Permet de définir une servlet qui ne répondra qu'à une seule requête à la fois
Les classes	GenericServlet	Classe définissant une servlet indépendante de tout protocole
	ServletInputStream	Flux permet la lecture des donnes de la requête cliente
	ServletOutPutStream	Flux permettant l'envoie de la reponse de la servlet
Les exceptions	SevletException	Exception générale en cas de problème durant l'exécution de la servlet
	UnavailableException	Exception levée si la servlet n'est pas disponible

- Le package javax.servlet.http définit plusieurs interfaces et méthodes :

Javax.servlet	Nom	Rôle
Les interfaces	HttpServletRequest	Hérite de ServletRequest : définit un objet contenant une requête selon le protocole http
	HttpServletResponse	Hérite de ServletResponse : définit un objet contenant la reponse de la servlet selon le protocole http
	HttpSession	Définit un objet qui représente une session
Les classes	Cookie	Classe représentant un cookie (ensemble de données sauvegardées par le brower sur le poste client)
	HttpServlet	Hérite de GenericServlet : classe définissant une servlet utilisant le protocole http
	HttpUtils	Classe proposant des méthodes statiques utiles pour le développement de servlet http

- Les méthodes :

La méthode	Son rôle
getParameter()	Utilisé pour obtenir les informations dont vous avez besoin à partir de la page HTML du client. Nous utilisons ***request.getParameter()*** pour extraire les paramètres de requête (c'est-à-dire les données envoyées en postant un formulaire html).***request.getParameter()*** renvoie toujours la valeur ***String*** et les données proviennent du client.
getAttribute() setAttribute()	Utilisé pour obtenir les paramètres précédemment définis dans une autre page JSP ou Servlet. Nous utilisons ***request.getAttribute()*** pour obtenir un objet ajouté à la portée de la requête côté serveur, c'est-à-dire en utilisant ***request.setAttribute()*** . Vous pouvez ajouter n'importe quel type d'objet que vous aimez ici, ***Strings*** , objets personnalisés, en fait n'importe quel objet. Vous ajoutez l'attribut à la demande et transférez la demande à une autre ressource, le client ne le sait pas. Donc, tout le code traitant ceci serait typiquement dans JSP / servlets. Vous pouvez utiliser ***request.setAttribute()*** pour ajouter des informations supplémentaires et transférer / rediriger la requête en cours vers une autre ressource.

IV-Implémentation du programme :

1-La Servlet « creer_client.java » :

creer_client.java | creer_reservation.java | InfoReservation.jsp | Inscription.jsp | Reservation.jsp | InfoClient.jsp | Client.java

```java
package ma.rsv.servlet;

import java.io.IOException;

/**
 * Servlet implementation class creer_client
 */
@WebServlet("/creer_client")
public class creer_client extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public creer_client() {
        super();
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String nom=request.getParameter("nom");
        String prénom=request.getParameter("prénom");
        String télé=request.getParameter("télé");
        String email=request.getParameter("email");

        String message;

        if(nom.trim().isEmpty() ||prénom.trim().isEmpty()||télé.trim().isEmpty()||email.trim().isEmpty())
            { message="Vous devez remplir tous les champs SVP.....!!";
            request.setAttribute("message", message);
            this.getServletContext().getRequestDispatcher("/Inscription.jsp").forward(request,response);
            }else {
                message="Inscription avec succés....!!!";
                request.setAttribute("message", message);

                Client client=new Client();

                client.setNom(nom);
                client.setPrenom(prénom);
                client.setTelephone(télé);
```

creer_client.java | creer_reservation.java | InfoReservation.jsp | Inscription.jsp | Reservation.jsp | InfoClient.jsp | Client.java

```java
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String nom=request.getParameter("nom");
        String prénom=request.getParameter("prénom");
        String télé=request.getParameter("télé");
        String email=request.getParameter("email");

        String message;

        if(nom.trim().isEmpty() ||prénom.trim().isEmpty()||télé.trim().isEmpty()||email.trim().isEmpty())
            { message="Vous devez remplir tous les champs SVP.....!!";
            request.setAttribute("message", message);
            this.getServletContext().getRequestDispatcher("/Inscription.jsp").forward(request,response);
            }else {
                message="Inscription avec succés....!!!";
                request.setAttribute("message", message);

                Client client=new Client();

                client.setNom(nom);
                client.setPrenom(prénom);
                client.setTelephone(télé);
                client.setEmail(email);

                request.setAttribute("client", client);
```

2-Page JSP « incription.jsp » :

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp | Client.java | Res

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div>
<h3>inscription</h3>
<form action="client" method="post">
<table>
<tr>
<td>Nom :</td>
<td><input type="text" name="nom" ></td>
</tr>

<tr>
<td>Prénom :</td>
<td><input type="text" name="prenom" ></td>
</tr>

<tr>
<td>T                :</td>
<td><                pe="text" name="tel" ></td>
</tr>

<tr>
```

3-Javabeans « Client.java » :

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp | Client.java | Reservation.java | web.xml

```
1  package com.rsv.beans;
2
3  public class Client {
4
5
6      private String nom;
7      private String prenom;
8      private String telephone;
9      private String email;
10     private Reservation reservation;
11
12     public Client(){
13
14     }
15
16
17     public String getNom() {
18         return nom;
19     }
20
21     publi  23  om(String nom) {
22         th    om;
23     }
24
25     public String getPrenom() {
26         return prenom;
```

4-Page JSP « infoClient.jsp » :

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp | Client.java | Reservation.java | web.xml

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ page import="com.rsv.beans.Client" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<b style="color : green;">
<%
String message =(String) request.getAttribute("message");
if(message != null)
{
    out.print(message);
}
%>
</b>
<%
Client client =(Client)request.getAttribute("client");
%>
<table border="1">
<tr>
<td>Nom : </td>
<td><%=client.getNom()%></td>
</tr>
<tr>
<td>Prenom : </td>
<td><%=client.getPrenom()%></td>
</tr>
<tr>
<td>Telephone : </td>
```

5-La servlet « créer_reservation.java » :

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp | Client.java

```
1  package ma.rsv.servlet;
2
3  import java.io.IOException;
10 /**
11  * Servlet implementation class creer_reservation
12  */
13 @WebServlet("/creer_reservation")
14 public class creer_reservation extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public creer_reservation() {
21         super();
22         // TODO Auto-generated constructor stub
23     }
24
25     /
26                 ttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
27
28     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29         String nom= request.getParameter("nom");
30         String prenom= request.getParameter("prenom");
```

23

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp | Client.java | Reservation.java

```
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String nom= request.getParameter("nom");
        String prenom= request.getParameter("prenom");
        String tel= request.getParameter("tel");
        String email= request.getParameter("email");
        String type = request.getParameter("type");
        double prix;
        try {
            prix = Double.parseDouble(request.getParameter("prix"));
        }
        catch(NumberFormatException e)
        {
            prix=0.5;
        }

        String message;

        if(nom.trim().isEmpty() || prenom.trim().isEmpty() || tel.trim().isEmpty() || email.trim().isEmpty() || type.trim().isEmpty()
                || type.trim().isEmpty() || prix==0.5)
        {
            message="vous devez remplir tous les champs!";
            request.setAttribute("message", message);
            getServletContext().getRequestDispatcher("/reservation.jsp").forward(request,response);

        }
        else
        {
```

23

6-Page JSP « reservation.jsp » :

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div>
<h3>réservation</h3>
<form action="reservation" method="post">
<table>
<tr>
<td>Nom :</td>
<td><input type="text" name="nom" ></td>
</tr>

<tr>
<td>Prénom :</td>
<td><input type="text" name="prenom" ></td>
</tr>

<tr>
<td>Téléphone :</td>
<td><input type="text" name="tel" ></td>
</tr>
```

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.j

```
17  </tr>
18
19  <tr>
20  <td>Prénom :</td>
21  <td><input type="text" name="prenom" ></td>
22  </tr>
23
24  <tr>
25  <td>Téléphone :</td>
26  <td><input type="text" name="tel" ></td>
27  </tr>
28
29  <tr>
30  <td>Email :</td>
31  <td><input type="text" name="email" ></td>
32  </tr>
33
34  <tr>
35  <td>Type :</td>
36  <td>
37  <select name="type">
38
39  <option value="1 chambre">1 chambre</option>
40  <option value="2 chambre">2 chambre</option>
41  <option value="3 chambre">3 chambre</option>
42
```

7-JavaBeans « Reservation.java » :

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp

```
1  package com.rsv.beans;
2
3  public class Reservation {
4
5      private String type;
6      private double prix;
7
8      public Reservation()
9      {
10
11     }
12
13     public String getType() {
14         return type;
15     }
16
17     public void setType(String type) {
18         this.type = type;
19     }
20
21     double getPrix() {
22         turn prix;
23     }
24
25     public void setPrix(double prix) {
```

8-Page JSP « info_reservation » :

creer_client.java | creer_reservation.java | inscription.jsp | reservation.jsp | info_client.jsp | info_reservation.jsp | Clie

```
2     pageEncoding="ISO-8859-1"%>
3     <%@ page import="com.rsv.beans.Client" %>
4  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6  <head>
7  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8  <title>Insert title here</title>
9  </head>
10 <body>
11 <b style="color : green;">
12 <%
13 String message =(String) request.getAttribute("message");
14 if(message != null)
15 {
16     out.print(message);
17 }
18 %>
19 </b>
20 <br>
21 <%
22 Client client =(Client)request.getAttribute("client");
23 %>
24 <b>Bi  23  Mr <%=client.getNom() %></b>
25 <tabl  r="1">
26 <tr>
27 <td>Nom : </td>
28 <td><%= client.getNom()%></td>
```

9-web.xml :

creer_client.java | creer_reservation.java | InfoReservation.jsp | Inscription.jsp | Reservation.jsp | InfoClient.jsp

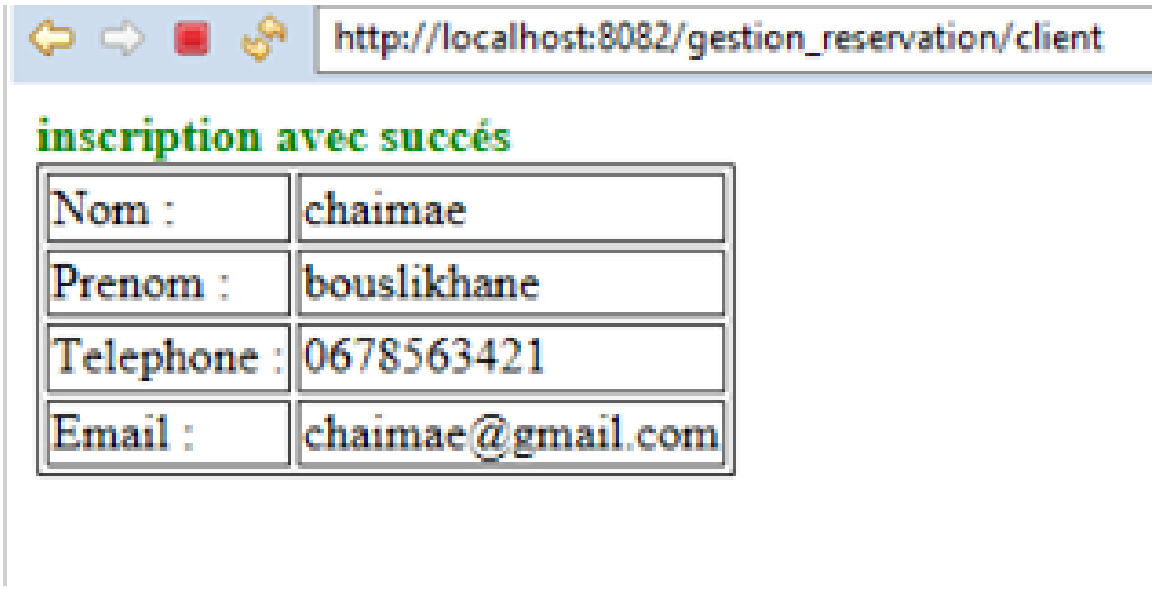
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
id="WebApp_ID" version="4.0">
  <display-name>Gestion_reservation</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>creer_client</servlet-name>
  <servlet-class>ma.rsv.servlet.creer_client</servlet-class>
  </servlet>
   <servlet-mapping>
  <s 23 -name>creer_client</servlet-name>
  <u tern>/client</url-pattern>
  </servlet-mapping>
    <servlet>
```

V-Exemple d'exécution :

- Pour faire l'inscription le client doit remplir le formulaire suivant :

- Au cas où le client laisse un champ vide un message d'erreur va s'afficher :

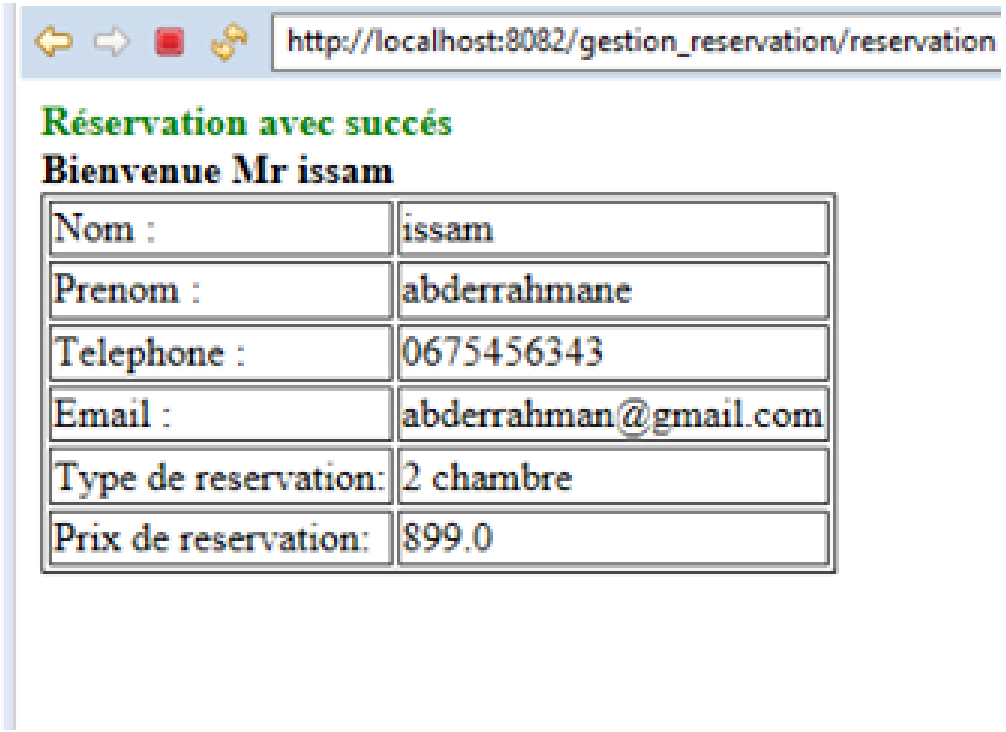
- Lorsque le client remplit tous les champs et clique sur le bouton s'inscrire :



- Pour faire la réservation d'une chambre ou plus le client doit remplir le formulaire suivant :

- Au cas où le client laisse un champ vide un message d'erreur va s'afficher :

- Lorsque le client remplit tous les champs et clique sur le bouton envoyer :



VI-Conclusion :

- Nous voici au terme de notre travail qui a porté sur la gestion de réservation d'hôtel. Pour réaliser ledit projet nous nous sommes servis de certains logiciels parmi lesquels ***Apache-tomcat*** et ***Eclipse***, hormis l'introduction générale et la conclusion, notre travail est subdivisé en quatre partie la première intitulée ***Définitions*** où on a indiqué et définit toutes les méthodes utilisées, en suite ***Analyse et conception*** illustrant l'application, les paquetages et les méthodes,

le code source de l'application qui est présenté dans la partie ***Implémentation du programme***, et en fin l'exemplification et illustration de l'exécution de l'application mise en évidence dans la partie ***Exemple d'exécution.*** En perspective, notre application peut être améliorée en ajoutant d'autres fonctionnalités, cependant elle pourra bien être de grande utilité en permettant aux clients de faire une réservation à distance (en ligne) chose qui va bel et bien améliorer l'expérience client tout en facilitant la tâche aux personnels de l'hôtel.