Base de données – Projet Fil Rouge (2)

AGENCE ESCAPADE

DUPAIN Olivier – BERNARD Bruno

27.05.18 - TD C

I | Check-in :

1. E1 : Le client souhaite réserver un séjour :

Message (*M1) XML, généré par le mobile, permettant l'envoi de la requête client :

```xml
<Client sexe="masculin">
  <Nom>DUPAIN</Nom>
  <Adresse arrondissement="16eme">ESILV, la defense</Adresse>
  <Date>visite de la Défense</Date>
</Client>
```

2. E2 : Le programme vérifie si le client existe :

Lecture du message *M1 par le programme qui utilise une requête XPath *P1 pour y trouver le nom du client :

```
string maRequeteXPath = "/Client/Nom";
XPathExpression expr = nav.Compile(maRequeteXPath);
```

Le programme vérifie ensuite l'existence de ce nom client dans la base de données par une requête SQL. Soit le client existe, soit il n'existe pas (les 2 cas sont traités).

Requête SQL pour interrogation de la BD : `command.CommandText = "select Nom, NumeroClient from Client";`

Extrait de code pour gérer les 2 cas possibles :

```
while (reader.Read())// Parcours l'ensemble des lignes des données (renvoie false quand une ligne est vide)
{
    nom = reader.GetString(0);  // récupération de la 1ère colonne
    if (nom == nomClient) //cas du client existant
    {
        numeroClient = reader.GetString(1); // on récupere le numero client
        Console.WriteLine("Le client existe déjà, son numéro client est" + numeroClient+".\n\n");
    }
}
reader.Close();
if (numeroClient == "") // cas du nouveau client
{
    MySqlCommand nouvellecommmande = connection.CreateCommand();
    numeroClient = "3344";
    nouvellecommmande.CommandText = "INSERT INTO `Client` (`NumeroClient`, `Nom`, `Prenom`, `Adresse`, `Numer
```

Explications : *Le nom et numéro client (string) sont instanciés à « ». Si le programme trouve une concordance avec le nom alors le nom et num client sont remplacés par leur valeur, s'il n'existe pas, le numeroClient reste « » et le programme entre dans le second if.*

3. E3 : Le programme sélectionne une voiture :

```
select Immatriculation , est_dispo from Voiture v, Stationne s , Parking p where p.CodePostal = '75016' and p.CodeParking = s.Parking_CodeParking and s.Voiture_immatriculation = v.immatriculation; "
```

Sélection d'une voiture disponible dans un parking défini a l'avance et récupération de son immatriculation par le programme.

Si la voiture est disponible dans l'arrondissement souhaité il l'a sélectionne directement, dans le cas contraire, il déplace une voiture d'un autre arrondissement : *Extrait de code :*

```
if (ImmatVoiture == "")
{
    MySqlCommand nouvellecommmande = connection.CreateCommand();
    nouvellecommmande.CommandText = "select Immatriculation from Voiture where est_dispo='1';";
    reader = nouvellecommmande.ExecuteReader();
    reader.Read();
    ImmatVoiture = reader.GetString(0);
    reader.Close();

    MySqlCommand nouvellecommmande2 = connection.CreateCommand();
    nouvellecommmande2.CommandText = "update Stationne set Parking_CodeParking='A16' where Voiture_Immatriculation='" + ImmatVoiture + "';";
    reader = nouvellecommmande2.ExecuteReader();
    reader.Close();
}
```

4. E5 : Sélection de l'appartement :

Pour sélectionner un appartement, le programme commence par désérialiser toutes les données de du JSON fourni par RNBP dans une nouvelle classe nommée appartement. Cette fonction permet d'avoir tous les appartements et leurs caractéristiques dans la classe appartement, mais aussi retourne la liste de l'ensemble des appartements.

```
StreamReader reader = new StreamReader("reponseRBNP.Json");
string JsonToString = reader.ReadToEnd();

List<Appartement> apparts = JsonConvert.DeserializeObject<List<Appartement>>(JsonToString);
```

Une autre fonction chosirAppartement, crée une nouvelle liste d'appartement ne comprenant cette fois ci que les critères relatifs à notre recherche.

```
foreach(Appartement i in apparts)
{
    if(i.borough==16 && i.overall_satisfaction>=4.5 && i.bedrooms==1 && i.availability=="yes")
    {
        selection.Add(i);
    }
```

Enfin le programme de démonstration affiche les trois premiers éléments de la liste pour les proposer au client afin qu'il choisisse le bien qui lui correspond le plus.

```
for (int i = 0; i < 3;i++)
{
    Console.WriteLine(appartement_selection[i].ToString());
    Console.WriteLine("\n");
}
```

/!\ *Pour les besoins de la démonstration, nous simulons que le client réserve le premier appartement proposé par le programme de démonstration.*

Génération du message (*Jx) JSON en réponse à l'API :

```
//instanciation des "writer"
StreamWriter writer = new StreamWriter(monFichier);
JsonTextWriter jwriter = new JsonTextWriter(writer);

//debut du fichier Json
jwriter.WriteStartObject();

//debut du tableau Json
jwriter.WritePropertyName("Reponse_d_ESCAPADE");
jwriter.WriteStartArray();

jwriter.WriteStartObject();
jwriter.WritePropertyName("host_id");
jwriter.WriteValue(host_id);
jwriter.WritePropertyName("room_id");
jwriter.WriteValue(room_id);
jwriter.WritePropertyName("Week");
jwriter.WriteValue(week);
jwriter.WritePropertyName("availibility");
jwriter.WriteValue(availibility);
jwriter.WriteEndObject();

jwriter.WriteEndArray();
jwriter.WriteEndObject();

//femeture de "writer"
jwriter.Close();
writer.Close();
```

Résultat JSON :


```
{"Reponse_d_ESCAPADE":[{"host_id":"2626","room_id":"2893116","Week":"2018-14","availibility":"no"}]}
```


5. E6 : Le programme effectue la réservation en mode non confirmée :

Le programme récupère dans un premier temps les informations relatives à l'immatriculation sélectionnée :

```
"select p.nom, v.place from Voiture v, Parking p, Stationne s where v.immatriculation ='" + immat_voiture + "' and voiture_immatriculation=immatriculation and Parking_CodeParking = CodeParking"
```

Puis il crée un document Xml à destination de RBNP afin de finaliser la réservation en mode non confirmée : *Extrait de code :*

```
XmlDocument docXml = new XmlDocument();

docXml.CreateXmlDeclaration("1.0", "UTF-8", "no");
docXml.CreateComment("xml version='1.0' encoding='UTF-8'");

XmlElement racine_de_racine = docXml.CreateElement("Message_M2");
docXml.AppendChild(racine_de_racine);

XmlElement info_voyage = docXml.CreateElement("Detail_du_voyage");
racine_de_racine.AppendChild(info_voyage);

XmlElement reservation = docXml.CreateElement("Numero_reservation");
reservation.InnerText = "00555";
info_voyage.AppendChild(reservation);

XmlElement adherent = docXml.CreateElement("Adherant");
info_voyage.AppendChild(adherent);
```

Le message M2 présente de nombreuses informations :
-le détail du voyage (numéro de résa, client, détail de réservation, détail parking)
-le listing des 3 appartements et de leurs caractéristiques

```xml
<Message_M2>
  <Detail_du_voyage>
    <Numero_reservation>00555</Numero_reservation>
    <Adherant>
      <Nom>DUPAIN</Nom>
      <Numero_Adherent>3344</Numero_Adherent>
    </Adherant>
    <Detail_reservation>
      <Nom_theme>16_ieme_arrondissement</Nom_theme>
      <Date_sejour>Semaine_14</Date_sejour>
      <etat_resa>sejour_valide</etat_resa>
    </Detail_reservation>
    <Detail_voiture>
      <Nom_parking>VICTOR HUGO</Nom_parking>
      <Numero_place>A1</Numero_place>
      <Immatriculation_du_vehicule>18NVA00</Immatriculation_du_vehicule>
    </Detail_voiture>
  </Detail_du_voyage>
  <Appartement_propose>
    <Reference_hebergement>2893116</Reference_hebergement>
    <Type_de_chambre>Entire home/apt</Type_de_chambre>
    <Pres_de>Porte-Dauphine</Pres_de>
    <Theme>16</Theme>
    <Satisfaction_generale>4,5</Satisfaction_generale>
    <Nombre_de_chambre>1</Nombre_de_chambre>
    <Prix>114</Prix>
    <Disponibilite>no</Disponibilite>
  </Appartement_propose>
  <Appartement_propose>
    <Reference_hebergement>330030</Reference_hebergement>
    <Type_de_chambre>Private room</Type_de_chambre>
    <Pres_de>Auteuil</Pres_de>
    <Theme>16</Theme>
    <Satisfaction_generale>4,5</Satisfaction_generale>
    <Nombre_de_chambre>1</Nombre_de_chambre>
    <Prix>123</Prix>
    <Disponibilite>yes</Disponibilite>
  </Appartement_propose>
  <Appartement_propose>
    <Reference_hebergement>645572</Reference_hebergement>
    <Type_de_chambre>Entire home/apt</Type_de_chambre>
    <Pres_de>Chaillot</Pres_de>
    <Theme>16</Theme>
    <Satisfaction_generale>4,5</Satisfaction_generale>
    <Nombre_de_chambre>1</Nombre_de_chambre>
    <Prix>145</Prix>
    <Disponibilite>yes</Disponibilite>
  </Appartement_propose>
</Message_M2>
```

6. E7 : Validation du séjour

Le message M3 de confirmation a été écrit à la main, il envoyé par le portable du client à destination de l'agence :

```
<Confirmation>
    <Numero_Client>3344</Numero_Client>
    <Nom>DUPAIN</Nom>
    <Numero_sejour>00555</Numero_sejour>
</Confirmation>
```

Le programme va ensuite extraire les informations de la réponse XML par des requêtes XPATH :

```
string maRequeteXPath = "/Confirmation/*";
```

Puis pour simplifier l'utilisation des données, nous avons décidé de traiter les données reçues du Xml par un dictionnaire sur c# :

```
Dictionary<string, string> info_resa = new Dictionary<string, string>();

while (nodes.MoveNext())
{
    text += nodes.Current.Name + " : " + nodes.Current.Value + "\n";
    info_resa.Add(nodes.Current.Name,nodes.Current.Value);
}
```

Puis le programme va ajouter la réservation dans la base de donnée :

```
INSERT INTO `ESCAPADE`.`Reserve` (`NumeroRésa`, `Client_NumeroClient`, `Séjour_idSejour`, `Voiture_Immatriculation`) VALUES (4,'"+info_resa["Numero_Client"]+"','"+info_resa["Numero_sejour"]+"','"+numero_immat+"');";
```

On utilise également un try-catch, dans le cas ou la réservation existe déjà, cela permet au programme de ne pas cesser de fonctionne si tel est le cas.

On effectue enfin la confirmation de la réservation dans la BD :

```
MySqlCommand commande2 = connection.CreateCommand();
commande2.CommandText = "update séjour set estConfirme = 1 where idSejour='"+info_resa["Numero_sejour"]+"';";
```

/!\ *Notre schéma a été pensé de manière à ce qu'un séjour ne corresponde qu'à une seule réservation et un seul client. Un même séjour ne peut donc pas correspondre à différentes réservations.*

II | Check-out :

1. Enregistrement de la position de la voiture rendue :

La requête SQL permettant de récupérer la place de la voiture :

```
"select Place from Voiture natural join Reserve natural join Client where NumeroClient='" + numeroClient + "' and est_rendu=1;";
```

La méthode retourne ensuite la nouvelle place de la voiture et informe le client dans la démo.

2. Contrôle de la voiture par les vérificateurs :

Le programme démarre par mettre la voiture indisponible :

```
command1.CommandText = "update Voiture set est_verif=0, est_dispo=0 where Immatriculation='" + immatVoiture + "';";
```

Ensuite ajoute un entretien à la voiture sélectionnée :

```
ommand2.CommandText = "INSERT INTO `Entretien` (`idEntretien`, `Motif`, `Date`, `Voiture_Immatriculation`, `Controleur_idControleur`) VALUES (0004, 'Netoyage', '2018-01-24', '" + immatVoiture + "', 001
```

Utilisation d'un try catch si l'entretien existe déjà afin de ne pas perturber le bon déroulement du programme en cas d'erreur de saisi sur l'id entretien et information dans le programme de démo de ce que le programme a effectué.

3. Nettoyage et remise en service du véhicule

Le programme de démo permet à l'utilisateur de nettoyer le véhicule.
Ensuite le véhicule ciblé est remis en service :

```
"update Voiture set est_verif=1, est_dispo=1 where Immatriculation='" + immatVoiture + "';";
```

III | Tableau de bord ESCAPADE :

On cherche tout d'abord à donner l'historique pour un véhicule donné (utilisé auparavant).
Pour cela, on récupère tout d'abord les informations sur tout le véhicule grâce à une requête :

```
"select idEntretien , Motif , Date , Controleur_idControleur  from Entretien where Voiture_Immatriculation ='" + immat + "';";
```

Le programme affiche ensuite l'historique de toutes les interventions par le code :

```
while (reader.Read())
{
    int numInter = Convert.ToInt32(reader.GetString(0));
    Console.WriteLine("Intervention numero :" + numInter);


    string motif = reader.GetString(1);
    string date = Convert.ToString(reader.GetString(2));
    int numCon = Convert.ToInt32(reader.GetString(3));

    Console.WriteLine("");
    Console.WriteLine("motif : " + motif);
    Console.WriteLine("date : " + date);
    Console.WriteLine("numero du controleur : " + numCon);
    Console.WriteLine("\n\n");

}
```

On s'intéresse maintenant à l'historique concernant le client traité précédemment.
Voici la requête sql utilisée :

```
select idSejour , Date , Theme , `Hebergement (par API)`  from Séjour a, Client b, Reserve c where b.NumeroClient = '" + numC + "' and b.NumeroClient=c.Client_NumeroClient and c.Séjour_idSejour=a.idSejour;"
```

Affichage requête :

```
while (reader.Read())
{
    string numSejour = reader.GetString(0);
    Console.WriteLine("Sejour numero :" + numSejour);


    string date = reader.GetString(1);
    string theme = Convert.ToString(reader.GetString(2));
    string hebergement = reader.GetString(3);


    Console.WriteLine("");
    Console.WriteLine("Date : " + date);
    Console.WriteLine("Theme : " + theme);
    Console.WriteLine("Hebergement : " + hebergement);
    Console.WriteLine("\n\n");

}
```

Enfin, on s'est intéressé à la rentabilité des séjours, en faisant chercher au programme le séjour le plus rentable.
Grâce à la requête suivante,

```
"select Séjour_idSejour from Reserve";
```

le programme récupère l'ensemble des séjours réservés. Ensuite, ces séjours sont comptés et le programme sait de fait, quel séjour a été le plus apprécié par les clients. Il l'annonce dans la démonstration de la manière suivante :

```
while (reader.Read())
{
    if (reader.GetString(0) == "0001") S1++;
    if (reader.GetString(0) == "0002") S2++;
    if (reader.GetString(0) == "0003") S3++;
}
int monMax = max(S1, S2, S3);
if (monMax == 1) Console.WriteLine("Le sejour dans le 1er arrodissement est le sejour le plus rentable, il a ete choisi a " + S1 + " reprises.");
if (monMax == 2) Console.WriteLine("Le sejour dans le 18ieme arrondissement est le sejour le plus rentable, il a ete choisi a " + S2 + " reprises.");
if (monMax == 3) Console.WriteLine("Le sejour dans le 12ieme arrondissement est le sejour le plus rentable, il a ete choisi a " + S3 + " reprises.");
```

NB : Dans toute la démonstration, pour des raisons de lisibilité et de fluidité dans la démonstration, nous avons choisi de ne pas afficher les documents xml et json dans la console, car leur lecture n'était pas suffisamment intuitive et fluide, nous avons donc préféré d'utiliser la commande Process.Start() qui permet d'ouvrir automatiquement l'ouverture des fichiers au moment désiré, afin de vous permettre de les consulter au fur et à mesure de la démonstration.