

# Ibn Battuta



**עבודת גמר**  
**בתכנון ותכנות מערכות**  
**במסלול שירותי רשת אינטרנט**  
**שנת הלימודים 2019 – 2020**

שם תלמיד : טארק שואהנה

בית ספר : ג'מאל טרביה סכנין

ת.ז. : 212680334

שם המורה : חנאן דראושה

**הקדשה/תודה :**

**מילת תודה והערכה למורה: על שנתיים נפלאות של  
השקעה והתמדה בתמיכתה ונכונתה אתנו בשיעורים, על  
התמיכה בי בפרויקט, היית לנו מורה למופת ונהדרת, ישר  
כח**

## תוכן עניינים

1	דף שער
2	הקדשה/תודה
3	תוכן עניינים
4	מבוא
4	מטרות האתר
4	קהל יעד
5	תיאור האתר
6	הטבלאות מתוך מסד הנתונים
11	קשרי הגומלין מתוך מסד הנתונים
12	Stored procedures (queries)
17	מבנה התיקיות
18	Business logic layers (Classes)
42	User interface (web forms)
59	Web service
67	Site map
68	מדריך משתמש

## מבוא :

مع تطور التكنولوجيا والوسائل التي تتيحها لنا، أصبحت حياة الإنسان المعاصر أسرع وأسهل بكثير من نمط الحياة قبل سنوات قليلة. فاليوم تتوفر لنا العديد من الخدمات والطرق للانكشاف إلى العالم والتعرف عليه. من هذه الخدمات المتوفرة هي طريقة تنظيم رحل عبر شبكة الإنترنت، وبما أن علينا أن نبني موقعاً في موضوع برمجة الحاسوب في الصف الثاني عشر، قررت أن ابني موقعاً ينظم رحل للمستخدم لعدة أماكن الذي يعرض للمستخدم وسائل مريحة وسهلة لمواكبة العصر الحديث وتسهيل طريقة معيشتة، حيث يستطيع الزبون أن ينظم عن طريق الإنترنت وأن يستخدم خدمات الموقع متى يشاء. بالإضافة لذلك هو وسيله لإيجاد عمل للمرشدين.

## מטרות האתר :

تسهيل إيجاد مناطق مناسبة لكل شخص لكل يخرج برحله اليها كما ويستطيع إيجاد مناطق جميله لم يكن يعلم بوجودها. بالإضافة لذلك الموقع يوفر إمكانية إيجاد عمل للمرشدين العاطلين عن العمل.

## קהל יעד :

كل إنسان يستطيع استخدام الموقع ولكن بما ان على المستخدم ادخال تفاصيل مهمه مثل معلومات ماليه (بطاقة الائتمان...) فيجب علينا وضع جيل أدنى

للاستخدام الموقع وخدماته، بما فيه الكفاية والجيل  
الملائم في هذه الحالة هو 15، لكي يكون المستخدم  
واعياً للتصرف بالمال.

#### **תיאור האתר :**

هذا الموقع عبارة عن مكتب سياحي، الذي يعرض  
للمستخدم وسائل مريحة وسهلة لمواكبة العصر الحديث  
وتسهيل طريقة معيشته، حيث يستطيع الزبون أن ينظم  
رحلات عن طريق الإنترنت وأن يستخدم خدمات الموقع  
متى يشاء.  
ولكي تكون إمكانية تنظيمهم للرحل المختلفة سهلة،  
مريحة وسريعة.

## הטבלאות מתוך מסד הנתונים

tblAdmins:

tblAdmins		
שם שדה	סוג נתונים	
IDAdmin	מספור אוטומטי	🔑
Name	טקסט קצר	
Email	טקסט קצר	
UserName	טקסט קצר	
Password	טקסט קצר	

في هذا الجدول يخزن معلومات عن مدير الموقع (Admin)

### الصفات:

- IDAdmin: حقل عبارة عن الرقم التسلسلي للمدير
- Name: حقل عبارة عن اسم للمدير
- Email: حقل عبارة عن البريد الإلكتروني للمدير
- UserName: اسم المستخدم
- Password: الكلمة السرية للمستخدم

## tblPerson:

tblPerson		
שם שדה	סוג נתונים	
IDPerson	מספור אוטומטי	
TZ	טקסט קצר	
FirstName	טקסט קצר	
LastName	טקסט קצר	
Birthday	תאריך/שעה	
Address	טקסט קצר	
Phone	טקסט קצר	
Email	טקסט קצר	
UserName	טקסט קצר	
Password	טקסט קצר	

- هذا الجدول عبارة عن مجمع لمعلومات عن المستخدم الذي يسجل في الموقع.
- رقم تسلسلي (لا يدخله المستخدم)، رقم الهوية، البريد الإلكتروني، رقم الهاتف.
- أسم المستخدم والסיסמה لكي يستطيع ال دخول الى الحساب.

### الصفات:

- **IDPerson**: حقل عبارة عن رقم الهوية الخاص للمستخدم
- **TZ**: حقل عبارة عن الرقم التسلسلي للمستخدم
- **FirstName**: الحقل هذا هو الاسم الأول للمستخدم
- **LastName**: اسم العائلة للمستخدم
- **Birthday**: تاريخ ميلاد المستخدم
- **Email**: البريد الإلكتروني للمستخدم
- **Address**: عنوان بيت المستخدم
- **Phone**: رقم الهاتف للمستخدم
- **UserName**: اسم المستخدم
- **Password**: الكلمة السرية للمستخدم

## tblMorshd:

tblMorshd		
שם שדה	סוג נתונים	
IDMorshd	מספור אוטומטי	
Name	טקסט קצר	
Birthday	תאריך/שעה	
Picture	טקסט קצר	
UserName	טקסט קצר	
Password	טקסט קצר	

- هذا الجدول عبارة عن مجمع لمعلومات عن المرشد الذي يسجل في الموقع لإيجاد عمل.
- رقم تسلسلي (لا يدخله المستخدم)، رقم الهوية، البريد الإلكتروني، رقم الهاتف.
- أسم المستخدم والסיסמה لكي يستطيع ال دخول الى الحساب.
- **IDMorshd**: حقل عبارة عن الرقم التسلسلي للمرشد.

**Name** : الحقل هذا هو الاسم للمرشد  
**Birthday**: تاريخ ميلاد المرشد  
**Picture**: صورته للمرشد  
**UserName**: اسم المستخدم للمرشد  
**Password**: الكلمة السرية للمرشد



**tblTrip:**

tblTrip		
שם שדה	סוג נתונים	
IDTrip	מספור אוטומטי	
TripTypeCode	מספר	
TripName	טקסט קצר	
TripPrice	מספר	
TripDate	תאריך/שעה	
TripPlace	טקסט קצר	
MaxPlaces	מספר	
TripPictures	טקסט קצר	
IDMorshd	מספר	

في هذا الجدول يخزن معلومات عن كل الرحلات التي ممكن للمستخدم  
حجزها

**IDTrip:** رقم تسلسلي للرحلة

**TripTybeCode:** نوع الرحلة

**TripName:** اسم الرحلة

**TripPrice:** تكلفة الرحلة

**TripDate:** تاريخ الرحلة

**TripPlace:** مكان الرحلة

**MaxPlaces:** العدد الأقصى للمشاركين بالرحلة

**TripPictures:** صور للرحلة

**IDMorshd:** الرقم التسلسلي للمرشد المسؤول عن الرحلة

### tblTripType:

tblTripType	
שם שדה	סוג נתונים
TripTypeCode	מספור אוטומטי
TripType	טקסט קצר
TripTypeDescription	טקסט קצר

هذا الجدول يعرض نوع الرحلة (ترفيهيه، تعليميه، ثقافيه....)

TripTypeCod: رقم تسلسلي لنوع الرحلة

TripTybe: نوع الرحلة

TripTybeDescription: وصف لنوع الرحلة

### tblOrderTrip:

tblOrderTrip	
שם שדה	סוג נתונים
IDOrderTrip	מספור אוטומטי
IDPerson	מספר
IDTrip	מספר
OrderQuantity	טקסט קצר

جدول يصف الرحل التي طلبها المستخدم

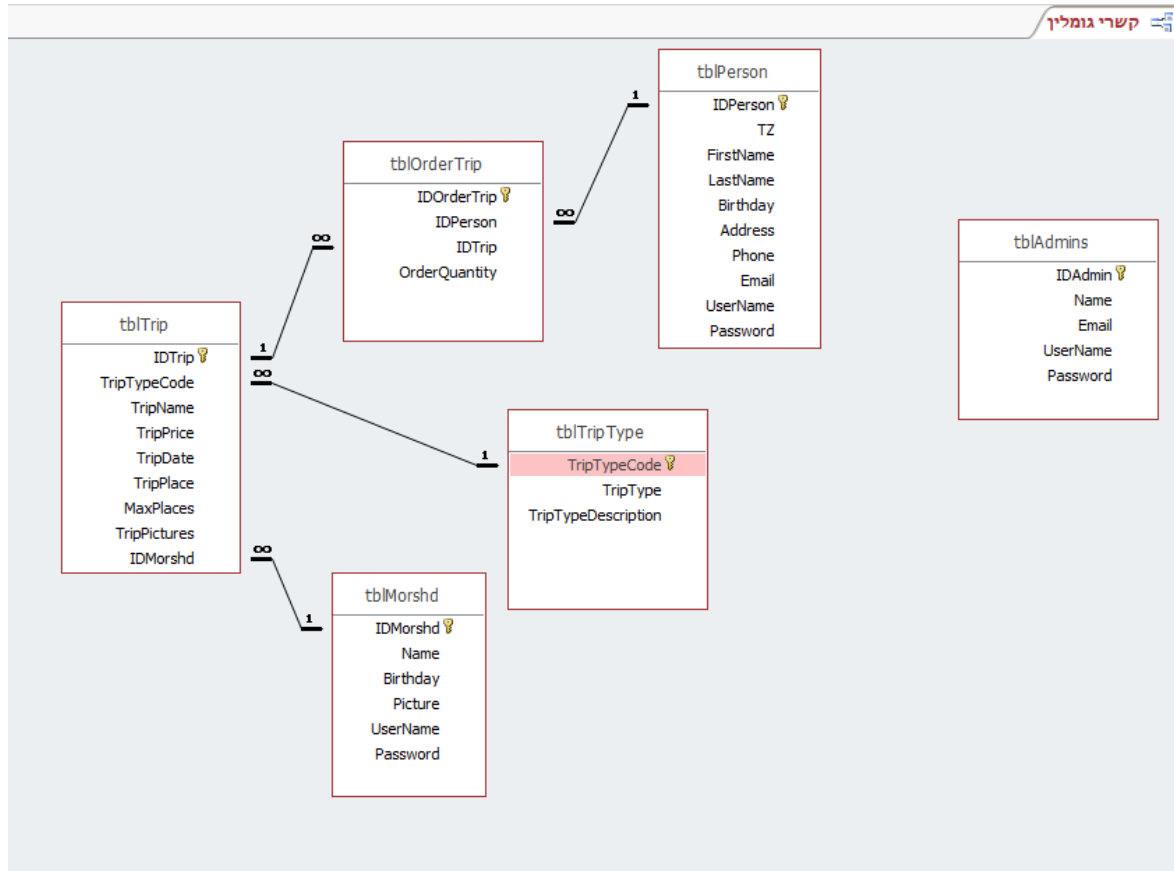
IDOrderTrip: الرقم التسلسلي للطلبية

IDPerson: الرقم التسلسلي للمستخدم

IDTrip: الرقم التسلسلي للرحلة

OrderQuantity: عدد الطلبيات

## מבנה מסד הנתונים


















העلاقة بين **tblTrip** و **tblMorshed** هي واحد لكثير بحيث يستطيع المرشد الواحد ان يكون مسؤول عن عدة رحل .

העلاقة بين **tblTrip** و **tblTripCode** هي واحد لكثير بحيث يمكن لعدة رحل ان تكون من نفس النوع

העلاقة بين **tblOrderTrip** و **tblTrip** هي واحد لكثير بحيث ان في الطلبيه الواحده ممكن ان يكون هناك عدة رحل

העلاقة بين **tblPerson** و **tblOrderTrip** هي واحد لكثير بحيث ان الشخص الواحد بامكانه ان يطلب عدة رحل

## שאלות ( Stored procedures )

שאלות	
spAllOptions	
spAllUserNamesAnd...	
spExistAdmin	
spExistMorshd	
spExistOrder	
spExistPerson	
spExistTrip	
spNotFullTrips	
spOrder	
spPossibleTrips	
spAddAdmin	
spAddMorshd	
spAddOrder	
spAddPerson	
spAddTrip	

### spAllOptions :

هذا الاستعلام يعرض كل الرحل الممكن التسجيل بها حسب اذا كان عدد المسجلين لها اقل من عدد الأماكن الاقصى

```
SELECT spPossibleTrips.IDTrip, spPossibleTrips.TripTypeCode, spPossibleTrips.TripName,
spPossibleTrips.TripPrice, spPossibleTrips.TripDate, spPossibleTrips.TripPlace,
spPossibleTrips.MaxPlaces, spPossibleTrips.TripPictures, spPossibleTrips.IDMorshd,
tblPerson.IDPerson

FROM spPossibleTrips, tblPerson

WHERE (((spPossibleTrips.IDTrip) Not In (Select IDTrip From tblOrderTrip Where
tblPerson.IDPerson=tblOrderTrip.IDPerson))));
```

### spAllUserNamesAndPasswords :

هذا الاستعلام يظهر جميع اسماء المستخدمين وكلمات المرور في الموقع.

```
SELECT tblPerson.UserName, tblPerson.Password

FROM tblPerson;
```

### spPossibleTrips :

هذا الاستعلام يعرض الرحل الممكن التسجيل بها حسب اذا كان عدد المسجلين لها اقل من عدد الأماكن الاقصى

```
SELECT tblTrip.IDTrip, tblTrip.TripTypeCode, tblTrip.TripName, tblTrip.TripPrice, tblTrip.TripDate,
tblTrip.TripPlace, tblTrip.MaxPlaces, tblTrip.TripPictures, tblTrip.IDMorshd

FROM tblTrip

WHERE (((tblTrip.IDTrip) In (Select IDTrip From spNotFullTrips)) AND ((tblTrip.TripDate)>=Now()));
```

### spBestTrips :

هذا الاستعلام يعرض افضل الرحل حسب تقييمها

```
SELECT tblTrip.TripPictures, tblTrip.TripName, tblTrip.TripRating

FROM tblTrip

WHERE (((tblTrip.TripRating)>=8));
```

### spNotFullTrips :

هذا الاستعلام يعرض الرحل الغير ممتلئه أي ما زال هناك امكانيه لتسجيل بها

```
SELECT tblTrip.IDTrip
FROM tblTrip LEFT JOIN tblOrderTrip ON tblTrip.IDTrip=tblOrderTrip.IDTrip
GROUP BY tblTrip.IDTrip
HAVING Count(*)<Max(MaxPlaces);
```

### SpOrder :

```
SELECT spAllOptions.IDTrip, spAllOptions.TripTypeCode, spAllOptions.TripName,
spAllOptions.TripPrice, spAllOptions.TripDate, spAllOptions.TripPlace, spAllOptions.MaxPlaces,
spAllOptions.TripPictures, spAllOptions.IDPerson, tblMorshd.Name
FROM spAllOptions INNER JOIN tblMorshd ON spAllOptions.IDMorshd=tblMorshd.IDMorshd;
```

### spExistAdmin :

هذا الاستعلام يتلقى رقم اسم المدير ويفحص عن طريقه فيما اذا كان المدير موجود في قاعدة البيانات.

```
SELECT Count(*) AS Expr1
FROM tblAdmins
WHERE (((tblAdmins.Name)=[@Name]));
```

### spExistMorshd :

هذا الاستعلام يتلقى اسم المرشد ويفحص عن طريقه فيما اذا كان المرشد موجود في قاعدة البيانات.

```
SELECT Count(*) AS Expr1
FROM tblMorshd
WHERE (((tblMorshd.Name)=[@Name]) AND ((tblMorshd.Birthday)=[@Birthday]));
```

### spExistTrip:

هذا الاستعلام يتلقى تفاصيل رحله ويفحص عن طريقها فيما اذا كانت الرحله موجوده في قاعدة البيانات.

```
SELECT Count(*) AS Expr1
FROM tblTrip
WHERE (((tblTrip.TripName)=[@Name]) AND ((tblTrip.TripDate)=[@Date]) AND
((tblTrip.TripPlace)=[@Place]) AND ((tblTrip.IDMorshd)=[@IDMorshd]));
```

### spExistPerson:

هذا الاستعلام يتلقى رقم هوية الشخص ويفحص عن طريقها فيما اذا كان الشخص موجود في قاعدة البيانات.

```
SELECT Count(*) AS Expr1
FROM tblPerson
WHERE (((tblPerson.TZ)=[@TZ]));
```

### spExistOrder:

هذا الاستعلام يتلقى رقم تسلسلي للرحلة والرقم التسلسلي للشخص ويفحص عن طريقهم فيما إذا كانت الطلبية موجودة في قاعدة البيانات.

```
SELECT count(*)
FROM tblOrder
WHERE IDTrip=[@IDTrip] And IDPerson=[@IDPerson];
```

### spAddAdmin :

هذا الاستعلام يقوم بإضافة مدير جديد الى قاعدة البيانات.

```
INSERT INTO tblAdmins ( Name, Email, UserName, [Password] )
SELECT [@Name] AS Expr1, [@Email] AS Expr2, [@UserName] AS Expr3, [@Password] AS Expr4;
```

### spAddMorshd:

هذا الاستعلام يقوم بإضافة مرشد جديد الى قاعدة البيانات.

```
INSERT INTO tblMorshd ( Name, Birthday, Picture, UserName, [Password] )
```

```
SELECT [@Name] AS Expr1, [@Birthday] AS Expr2, [@Picture] AS Expr3, [@UserName] AS Expr4,  
[@Password] AS Expr5;
```

### spAddTrip:

هذا الاستعلام يقوم بإضافة رحلة جديد الى قاعدة البيانات.

```
INSERT INTO tblTrip ( TripTypeCode, TripName, TripPrice, TripDate, TripPlace, TripPictures,  
IDMorshd, MaxPlaces )
```

```
SELECT [@TripTypeCode] AS Expr7, [@TripName] AS Expr6, [@TripPrice] AS Expr5, [@TripDate] AS  
Expr4, [@TripPlace] AS Expr3, [@TripPictures] AS Expr2, [@IDMorshd] AS Expr1, [@MaxPlaces] AS  
Expr8;
```

### spAddPerson:

هذا الاستعلام يقوم بإضافة شخص جديد الى قاعدة البيانات.

```
INSERT INTO tblPerson ( TZ, FirstName, LastName, Birthday, Address, Phone, Email, Username,  
[Password] )
```

```
VALUES ([@TZ], [@FirstName], [@LastName], [@Birthday], [@Address], [@Phone], [@Email],  
[@Username], [@Password]);
```

### spAddOrder:

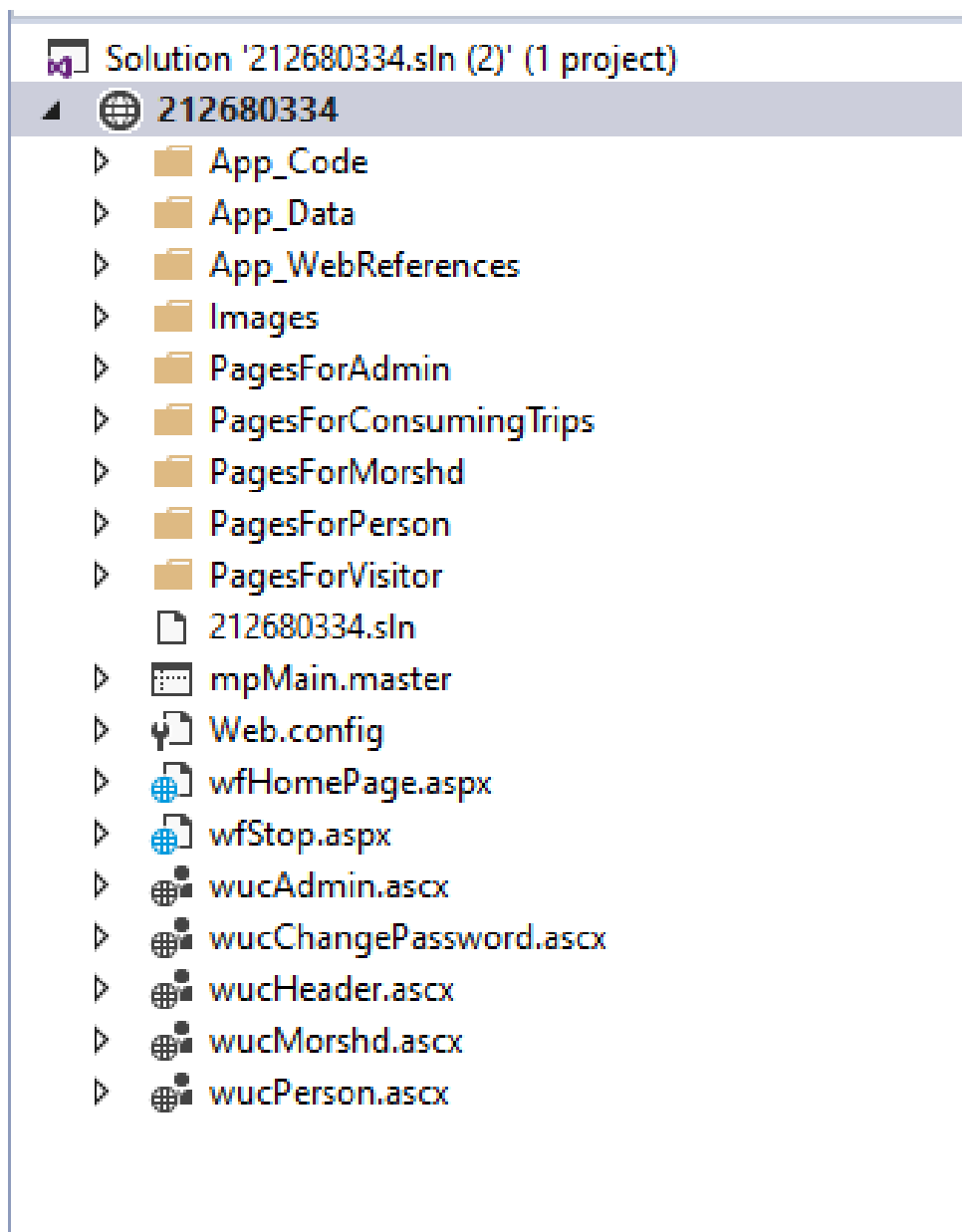
هذا الاستعلام يقوم بإضافة طلبية جديدة الى قاعدة البيانات.

```
INSERT INTO tblOrderTrip ( IDTrip, IDPerson, OrderQuantity )
```















```
VALUES ([@IDTrip], [@IDPerson], [@OrderQuantity]);
```



## מבנה התיקיות



## Business logic layers (Classes)

▷  Project References	▶  App_Code
▷  Admin	C# Admin.cs
▷  AdminDetails	C# DoQueries.cs
▷  DoQueries	C# Morshd.cs
▷  Morshd	C# OrderTrip.cs
▷  MorshdDetails	C# Person.cs
▷  OrderTrip	C# PicList.cs
▷  OrderTripDetails	C# Trip.cs
▷  Person	
▷  PersonDetails	
▷  PicList	
▷  Trip	
▷  TripDetails	

## המחלקות

### 1. Admin:

الصفات:

```
private int _IDAdmin;  
private string _Name;  
private string _Email;  
private string _UserName;  
private string _Password;
```

عمليات Get(), Set() لكل صفة من هذه الفئة

```
//Constructor  
0 references  
public Admin()  
{  
  
}
```

لا يوجد عمليات أخرى في هذه الفئة

كود الفئة:

### AdminDetails

```
public class AdminDetails  
{  
    //Encapsulation  
    private int _IDAdmin;  
    private string _Name;  
    private string _Email;  
    private string _UserName;  
    private string _Password;  
  
    //Properties  
    public int IDAdmin  
    {  
        get { return _IDAdmin; }  
        set { _IDAdmin = value; }  
    }  
    public string Name  
    {  
        get { return _Name; }  
        set { _Name = value; }  
    }  
    public string Email  
    {  
        get { return _Email; }  
    }  
}
```

```

        set { _Email = value; }
    }
    public string UserName
    {
        get { return _UserName; }
        set { _UserName = value; }
    }
    public string Password
    {
        get { return _Password; }
        set { _Password = value; }
    }
    public AdminDetails()
    {
    }
}

```

## Admin

### الصفات:

```

ArrayList prmlist;
OleDbParameter prm;
string strSQLName;

```

```

//Constructor
public Admin()
{
}

```

عمليات إضافية:

```
public void AddAdmin(AdminDetails AdmDetails)
```

هدف هذه الدالة هو اضافة مدير جديد للموقع .

إدعاء الدخول: الدالة تستقبل كائن من نوع AdminDetails.

إدعاء الخروج: الدالة لا ترجع قيمة، بل إنها فقط تنفذ عملية وهي إدخال قيم إلى الكائن (برامتر الدالة).

```
public Boolean ExistAdmin(AdminDetails AdmDetails)
```

هدف هذه العملية هو التأكد من وجود المدير في قاعدة البيانات، واستعمالها يكون قبل تنفيذ الدالة AddAdmin، لأنه ليس منطقياً إدخال نفس الكائن (مدير) أكثر من مرة إلى قاعدة البيانات.

إدعاء الدخول: هذه الدالة تتلقى كائن من نوع AdminDetails.

إدعاء الخروج: هذه الدالة ترجع قيم True/False، أي ترجع قيمة True إذا

الكائن موجود في قاعدة البيانات، وترجع False إذا الكائن غير موجود في

كود الفئة:

```
public class Admin
{
    //Encapsulation
    ArrayList prmlist;
    OleDbParameter prm;
    string strSQLName;

    //Constructor
    public Admin()
    {
        //
        // TODO: Add constructor logic here
        //
    }
    //Methods

    //_____AddAdmin_____

    public void AddAdmin(AdminDetails admDetails)
    {
        strSQLName = "spAddAdmin";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@Name", OleDbType.VarChar);
        prm.Value = admDetails.Name;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Email", OleDbType.VarChar);
        prm.Value = admDetails.Email;
        prmlist.Add(prm);

        prm = new OleDbParameter("@UserName", OleDbType.VarChar);
        prm.Value = admDetails.UserName;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Password", OleDbType.VarChar);
        prm.Value = admDetails.Password;
        prmlist.Add(prm);

        DoQueries.ExecuteSPNonQuery(strSQLName, prmlist);
    }
    //_____ExistAdmin_____

    public Boolean ExistAdmin(AdminDetails admDetails)
    {
        strSQLName = "spExistAdmin";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@Name", OleDbType.VarChar);
        prm.Value = admDetails.Name;
        prmlist.Add(prm);

        int intCount = (int)DoQueries.ExecuteSPScalar(strSQLName, prmlist);
        return intCount > 0;
    }
}
```

## 2. Person:

### PersonDetails

الصفات:

```
private int _IDPerson;  
private string _TZ;  
private string _FirstName;  
private string _LastName;  
private DateTime _Birthday;  
private string _Address;  
private string _Phone;  
private string _Email;  
private string _UserName;  
private string _Password;
```

عمليات () Get ، Set() لكل صفه من هذه الفئة

Constructor

```
public PersonDetails ()  
{  
  
}
```

لا يوجد عمليات أخرى لهذه الفئة

```
public class PersonDetails  
{  
    //Encapsulation  
    private int _IDPerson;  
    private string _TZ;  
    private string _FirstName;  
    private string _LastName;  
    private DateTime _Birthday;  
    private string _Address;  
    private string _Phone;  
    private string _Email;  
    private string _UserName;  
    private string _Password;  
  
    //Properties  
    public int IDPerson  
    {  
        get { return _IDPerson; }  
        set { _IDPerson = value; }  
    }  
}
```

```

    }
    public string TZ
    {
        get { return _TZ; }
        set { _TZ = value; }
    }
    public string FirstName
    {
        get { return _FirstName; }
        set { _FirstName = value; }
    }
    public string LastName
    {
        get { return _LastName; }
        set { _LastName = value; }
    }
    public DateTime Birthday
    {
        get { return _Birthday; }
        set { _Birthday = value; }
    }
    public string Address
    {
        get { return _Address; }
        set { _Address = value; }
    }
    public string Phone
    {
        get { return _Phone; }
        set { _Phone = value; }
    }
    public string Email
    {
        get { return _Email; }
        set { _Email = value; }
    }
    public string UserName
    {
        get { return _UserName; }
        set { _UserName = value; }
    }
    public string Password
    {
        get { return _Password; }
        set { _Password = value; }
    }

    //Constructor
    public PersonDetails()
    {
        _Birthday = new DateTime();
    }
}
//*****Class PersonDetails - end *****

```

## Person

الصفات:

```
ArrayList prmlist;  
OleDbParameter prm;  
string strSQLName;
```

Constructor

```
public Person()  
{  
  
}
```

### عمليات أخرى في الفئة:

```
public void AddPerson(PersonDetails prsDetails)
```

هدف هذه الدالة هو إدخال شخص جديد إلى قاعدة البيانات.

إدعاء الدخول: الدالة تستقبل كائن من نوع PersonDetails.

إدعاء الخروج: الدالة لا ترجع قيمة، إنما تنفذ عملية معينة.

```
public Boolean ExistProduct(ProductsDetails PrDetails)
```

هدف هذه الدالة هي التأكد من عدم وجود هذا الشخص في قاعدة البيانات مسبقاً.

ادعاء الدخول: الدالة تستقبل كائن من نوع PersonDetails.

ادعاء الخروج: الدالة Boolean أي ترجع قيم من نوع False/True, أي ترجع True إذا الشخص موجود في قاعدة البيانات، وترجع False إذا لم يكن موجوداً.



```
public class Person
{
    //Encapsulation
    ArrayList prmlist;
    OleDbParameter prm;
    string strSQLName;

    //Constructor
    public Person()
    {
        //
        // TODO: Add constructor logic here
        //
    }
    //Methods

    //_____AddPerson_____

    public void AddPerson(PersonDetails prsDetails)
    {
        strSQLName = "spAddPerson";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@TZ", OleDbType.VarChar);
        prm.Value = prsDetails.TZ;
        prmlist.Add(prm);

        prm = new OleDbParameter("@FirstName", OleDbType.VarChar);
        prm.Value = prsDetails.FirstName;
        prmlist.Add(prm);

        prm = new OleDbParameter("@LastName", OleDbType.VarChar);
        prm.Value = prsDetails.LastName;
        prmlist.Add(prm);

        prm = new OleDbParameter("@BirthDay", OleDbType.DBDate);
        prm.Value = prsDetails.Birthday;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Address", OleDbType.VarChar);
        prm.Value = prsDetails.Address;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Phone", OleDbType.VarChar);
        prm.Value = prsDetails.Phone;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Email", OleDbType.VarChar);
        prm.Value = prsDetails.Email;
        prmlist.Add(prm);

        prm = new OleDbParameter("@UserName", OleDbType.VarChar);
        prm.Value = prsDetails.UserName;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Password", OleDbType.VarChar);
        prm.Value = prsDetails.Password;
        prmlist.Add(prm);

        DoQueries.ExecuteSPNonQuery(strSQLName, prmlist);
    }
}
```

```

    }
    //_____ExistPerson_____

    public Boolean ExistPerson(PersonDetails prsDetails)
    {
        strSQLName = "spExistPerson";
        prmList = new ArrayList();

        prm = new OleDbParameter("@TZ", OleDbType.VarChar);
        prm.Value = prsDetails.TZ;
        prmList.Add(prm);

        int intCount = (int)DoQueries.ExecuteSPScalar(strSQLName, prmList);
        return intCount > 0;
    }
}
//*****Class Person - end *****

```

### 3. Morshd:

#### MorshdDetails

صفات:

```

private int _IDMorshd;
private string _Name;
private DateTime _Birthday;
private string _Picture;
private string _UserName;
private string _Password;

```

عمليات ( ) Get ، Set لكل صفه من هذه الفئة

```

//Constructor
public MorshdDetails()
{
    Birthday = new DateTime();
}

```

لا يوجد صفات أخرى لهذه الفئة

كود الفئة:

```

public class MorshdDetails
{
    //Encapsulation
    private int _IDMorshd;

```

```

private string _Name;
private DateTime _Birthday;
private string _Picture;
private string _UserName;
private string _Password;

//Properties
public int IDMorshd
{
    get { return _IDMorshd; }
    set { _IDMorshd = value; }
}
public string Name
{
    get { return _Name; }
    set { _Name = value; }
}
public DateTime Birthday
{
    get { return _Birthday; }
    set { _Birthday = value; }
}
public string Picture
{
    get { return _Picture; }
    set { _Picture = value; }
}
public string UserName
{
    get { return _UserName; }
    set { _UserName = value; }
}
public string Password
{
    get { return _Password; }
    set { _Password = value; }
}

//Constructor
public MorshdDetails()
{
    Birthday = new DateTime();
}
}

```

## Morshd

الصفات:

```

ArrayList prmlist;
OleDbParameter prm;
string strSQLName;

//Constructor
public TripDetails()
{
    TripDate = new DateTime();
}

```

## عمليات أخرى في الفئة:

```
public void AddMorshd(MorshdDetails mrdDetails)
```

هدف هذه الدالة هو إدخال مرشد جديد إلى قاعدة البيانات.

إدعاء الدخول: الدالة تستقبل كائن من نوع MorshdDetails.

إدعاء الخروج: الدالة لا ترجع قيمة، إنما تنفذ عملية معينة.

```
public Boolean ExistMorshd(MorshdDetails mrdDetails)
```

هدف هذه الدالة هي التأكد من عدم وجود هذا المرشد في قاعدة البيانات مسبقاً.

ادعاء الدخول: الدالة تستقبل كائن من نوع MorshdDetails.

ادعاء الخروج: الدالة Boolean أي ترجع قيم من نوع True/False, أي ترجع True إذا المرشد موجود في قاعدة البيانات، وترجع False إذا لم يكن موجوداً.

كود الفئة:

```
public class Morshd
{
    //Encapsulation
    ArrayList prmlist;
    OleDbParameter prm;
    string strSQLName;

    //Constructor
    public Morshd()
    {
    }
    //Methods

    // _____AddMorshd_____

    public void AddMorshd(MorshdDetails mrdDetails)
    {
        strSQLName = "spAddMorshd";
        prmlist = new ArrayList();
    }
}
```

```

        prm = new OleDbParameter("@Name", OleDbType.VarChar);
        prm.Value = mrdDetails.Name;
        prmlist.Add(prm);

        prm = new OleDbParameter("@DateTime", OleDbType.DBDate);
        prm.Value = mrdDetails.Birthday;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Picture", OleDbType.VarChar);
        prm.Value = mrdDetails.Picture;
        prmlist.Add(prm);

        prm = new OleDbParameter("@UserName", OleDbType.VarChar);
        prm.Value = mrdDetails.UserName;
        prmlist.Add(prm);

        prm = new OleDbParameter("@Password", OleDbType.VarChar);
        prm.Value = mrdDetails.Password;
        prmlist.Add(prm);

        DoQueries.ExecuteSPNonQuery(strSQLName, prmlist);
    }
    // _____ ExistMorshd _____

    public Boolean ExistMorshd(MorshdDetails mrdDetails)
    {
        strSQLName = "spExistMorshd";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@Name", OleDbType.VarChar);
        prm.Value = mrdDetails.Name;
        prmlist.Add(prm);

        prm = new OleDbParameter("@DateTime", OleDbType.DBDate);
        prm.Value = mrdDetails.Birthday;
        prmlist.Add(prm);

        int intCount = (int)DoQueries.ExecuteSPScalar(strSQLName, prmlist);
        return intCount > 0;
    }
}
//*****Class Morshd - end *****

```

## Trip:

## TripDetails

الصفات:

```
private int _IDTrip;  
private int _TripTypeCode;  
private string _TripName;  
private int _TripPrice;  
private DateTime _TripDate;  
private string _TripPlace;  
private int _MaxPlaces;  
private string _TripPictures;  
private int _TripRating;  
private int _IDMorshd;
```

عمليات Get () ، Set() لكل صفه من هذه الفئة

```
//Constructor  
public TripDetails()  
{  
    TripDate = new DateTime();  
}
```

لا يوجد عمليات أخرى في هذه الفئة

```
public class TripDetails  
{  
    //Encapsulation  
    private int _IDTrip;  
    private int _TripTypeCode;  
    private string _TripName;  
    private int _TripPrice;  
    private DateTime _TripDate;  
    private string _TripPlace;  
    private int _MaxPlaces;  
    private string _TripPictures;  
    private int _TripRating;  
    private int _IDMorshd;  
  
    //Properties  
    public int IDTrip  
    {  
        get { return _IDTrip; }  
        set { _IDTrip = value; }  
    }  
    public int TripTypeCode  
    {  
        get { return _TripTypeCode; }  
        set { _TripTypeCode = value; }  
    }  
    public string TripName  
    {  
        get { return _TripName; }  
        set { _TripName = value; }  
    }  
}
```

```

    }
    public int TripPrice
    {
        get { return _TripPrice; }
        set { _TripPrice = value; }
    }
    public DateTime TripDate
    {
        get { return _TripDate; }
        set { _TripDate = value; }
    }
    public string TripPlace
    {
        get { return _TripPlace; }
        set { _TripPlace = value; }
    }
    public int MaxPlaces
    {
        get { return _MaxPlaces; }
        set { _MaxPlaces = value; }
    }
    public string TripPictures
    {
        get { return _TripPictures; }
        set { _TripPictures = value; }
    }
    public int TripRating
    {
        get { return _TripRating; }
        set { _TripRating = value; }
    }
    public int IDMorshd
    {
        get { return _IDMorshd; }
        set { _IDMorshd = value; }
    }

    //Constructor
    public TripDetails()
    {
        TripDate = new DateTime();
    }
}

```

## Trip

الصفات:

```

//Encapsulation
ArrayList prmList;
OleDbParameter prm;
string strSQLName;

//Constructor
public Trip()
{
}

```

## عمليات أخرى في الفئة:

```
public void AddTrip(TripDetails trpDetails)
```

هدف هذه الدالة هو إدخال رحلة جديده إلى قاعدة البيانات.

إدعاء الدخول: الدالة تستقبل كائن من نوع TripDetails.

إدعاء الخروج: الدالة لا ترجع قيمة، إنما تنفذ عملية معينة.

```
public Boolean ExistTrip(TripDetails trpDetails)
```

هدف هذه الدالة هي التأكد من عدم وجود هذه الرحلة في قاعدة البيانات مسبقاً.

ادعاء الدخول: الدالة تستقبل كائن من نوع TripDetails.

ادعاء الخروج: الدالة Boolean أي ترجع قيم من نوع True/False, أي ترجع True إذا الرحلة موجودة في قاعدة البيانات، وترجع False إذا لم تكن موجوده.

```
public class Trip
{
    //Encapsulation
    ArrayList prmlist;
    OleDbParameter prm;
    string strSQLName;

    //Constructor
    public Trip()
    {
    }

    //Methods

    //_____AddTrip_____

    public void AddTrip(TripDetails trpDetails)
    {
        strSQLName = "spAddTrip";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@TripTypeCode", OleDbType.Integer);
        prm.Value = trpDetails.TripTypeCode;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripName", OleDbType.VarChar);
```



```

        prm.Value = trpDetails.TripName;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripPrice", OleDbType.Integer);
        prm.Value = trpDetails.TripPrice;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripDate", OleDbType.DBDate);
        prm.Value = trpDetails.TripDate;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripPlace", OleDbType.VarChar);
        prm.Value = trpDetails.TripPlace;
        prmlist.Add(prm);

        prm = new OleDbParameter("@MaxPlaces", OleDbType.Integer);
        prm.Value = trpDetails.MaxPlaces;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripPictures", OleDbType.VarChar);
        prm.Value = trpDetails.TripPictures;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripRating", OleDbType.Integer);
        prm.Value = trpDetails.TripRating;
        prmlist.Add(prm);

        prm = new OleDbParameter("@IDMorshd", OleDbType.Integer);
        prm.Value = trpDetails.IDMorshd;
        prmlist.Add(prm);

        DoQueries.ExecuteSPNonQuery(strSQLName, prmlist);
    }
    // _____ExistTrip_____

    public Boolean ExistTrip(TripDetails trpDetails)
    {
        strSQLName = "spExistTrip";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@TripName", OleDbType.VarChar);
        prm.Value = trpDetails.TripName;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripDate", OleDbType.DBDate);
        prm.Value = trpDetails.TripDate;
        prmlist.Add(prm);

        prm = new OleDbParameter("@TripPlace", OleDbType.VarChar);
        prm.Value = trpDetails.TripPlace;
        prmlist.Add(prm);

        prm = new OleDbParameter("@IDMorshd", OleDbType.Integer);
        prm.Value = trpDetails.IDMorshd;
        prmlist.Add(prm);

        int intCount = (int)DoQueries.ExecuteSPScalar(strSQLName, prmlist);
        return intCount > 0;
    }
}

```

## 1. OrderTrip:

### OrderTripDetails

الصفات:

```
private int _IDOrderTrip;  
private int _IDPerson;  
private int _IDTrip;  
private int _OrderQuantity;
```

عمليات Get(), Set() لكل صفه في هذه الفئة

```
//Constructor  
public OrderTripDetails()  
{  
  
}
```

لا يوجد عمليات أخرى في هذه الفئة

```
public class OrderTripDetails  
{  
    //Encapsulation  
    private int _IDOrderTrip;  
    private int _IDPerson;  
    private int _IDTrip;  
    private int _OrderQuantity;  
  
    //Properties  
    public int IDOrderTrip  
    {  
        get { return _IDOrderTrip; }  
        set { _IDOrderTrip = value; }  
    }  
    public int IDPerson  
    {  
        get { return _IDPerson; }  
        set { _IDPerson = value; }  
    }  
    public int IDTrip  
    {  
        get { return _IDTrip; }  
        set { _IDTrip = value; }  
    }  
    public int OrderQuantity  
    {  
        get { return _OrderQuantity; }  
        set { _OrderQuantity = value; }  
    }  
  
    //Constructor  
    public OrderTripDetails()  
    {  
  
    }  
}
```

## OrderTrip

الصفات

```
ArrayList prmlist;  
OleDbParameter prm;  
string strSQLName;
```

```
//Constructor  
public OrderTrip()  
{  
}
```

### عمليات أخرى في الفئة:

```
public void AddOrder(OrderTripDetails ordDetails)
```

هدف هذه الدالة هو إدخال طلب جديد إلى قاعدة البيانات.

إدعاء الدخول: الدالة تستقبل كائن من نوع OrderDetails.

إدعاء الخروج: الدالة لا ترجع قيمة، إنما تنفذ عملية معينة.

```
public Boolean ExistOrder(OrderTripDetails ordDetails)
```

هدف هذه الدالة هي التأكد من عدم وجود هذا الطلب في قاعدة البيانات مسبقاً.

ادعاء الدخول: الدالة تستقبل كائن من نوع OrderDetails.

ادعاء الخروج: الدالة Boolean أي ترجع قيم من نوع False/True, أي ترجع True إذا الطلب موجود في قاعدة البيانات، وترجع False إذا لم يكن موجوداً.

```

public class OrderTrip
{
    //Encapsulation
    ArrayList prmlist;
    OleDbParameter prm;
    string strSQLName;

    //Constructor
    public OrderTrip()
    {
    }
    //Methods

    //_____AddOrderTrip_____

    public void AddOrder(OrderTripDetails ordDetails)
    {
        strSQLName = "spAddOrder";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@IDTrip", OleDbType.Integer);
        prm.Value = ordDetails.IDTrip;
        prmlist.Add(prm);

        prm = new OleDbParameter("@IDPerson", OleDbType.Integer);
        prm.Value = ordDetails.IDPerson;
        prmlist.Add(prm);

        prm = new OleDbParameter("@OrderQuantity", OleDbType.Integer);
        prm.Value = ordDetails.OrderQuantity;
        prmlist.Add(prm);

        DoQueries.ExecuteSPNonQuery(strSQLName, prmlist);
    }
    //_____ExistOrderTrip_____

    public Boolean ExistOrder(OrderTripDetails ordDetails)
    {
        strSQLName = "spExistOrder";
        prmlist = new ArrayList();

        prm = new OleDbParameter("@IDTrip", OleDbType.Integer);
        prm.Value = ordDetails.IDTrip;
        prmlist.Add(prm);

        prm = new OleDbParameter("@IDPerson", OleDbType.Integer);
        prm.Value = ordDetails.IDPerson;
        prmlist.Add(prm);

        int intCount = (int)DoQueries.ExecuteSPScalar(strSQLName, prmlist);
        return intCount > 0;
    }
}

```

## 2. PicList:

### PicList

```
public class PicList
{
    //Constructor
    public PicList()
    {
    }
    //Methods

    public DataTable FillFiles()
    {
        WebService ws = new WebService();
        DataTable dt = new DataTable();
        DataColumn column;
        DataRow row;

        column = new DataColumn();
        column.DataType = System.Type.GetType("System.String");
        column.ColumnName = "PicName";
        column.ReadOnly = true;
        column.Unique = false;

        dt.Columns.Add(column);

        foreach (string fl in
Directory.GetFiles(ws.Server.MapPath(@"..\Images")))
        {
            row = dt.NewRow();
            row["PicName"] = Path.GetFileName(fl);
            dt.Rows.Add(row);
        }
        return dt;
    }
}
```

### 3. DoQueries:

## DoQueries

صفات:

```
private static string strConnection =  
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +  
System.Web.HttpContext.Current.Server.MapPath(@"../App_Data/EVBooking.mdb");
```

هذه الصفة هي string strConnection أي حين استخدام دالة التي تريد التواصل مع قاعدة البيانات (لاستخدام استعلام في برنامج Access أو أي شيء آخر...) يجب علينا "فتح" طريق لها لكي نتواصل مع قاعدة البيانات اينما كانت (نوجه الدالة إلى قاعدة البيانات).

## Methods:

```
public static OleDbConnection Connection()  
-----  
  
-----  
public static OleDbDataReader ExecuteReader(string strSQLName,  
ArrayList prmlist)  
-----  
  
-----  
public static OleDbDataReader ExecuteReader(string strSQLName)  
-----  
  
-----  
public static Object ExecuteSPScalar(string strSQLName, ArrayList  
prmlist)  
-----  
  
-----  
public static Object ExecuteSPScalar(string strSQLName)  
-----  
  
-----  
public static Object ExecuteScalar(string strSQL)
```

```
-----
public static bool ExecuteSPNonQuery(string strSQLName, ArrayList
prmList)
-----
```

```
-----
public static bool ExecuteSPNonQuery(string strSQLName)
-----
```

```
-----
public static bool ExecuteNonQuery(string strSQL)
-----
```

```
-----
public static void UpdateTableInDataBase(string strSQL, DataTable
dtNew)
-----
```

```
-----
public static DataSet ExecuteDataSet(string strSQL)
-----
```

```
-----
public static DataTable ExecuteDataTable(string strSQL)
-----
```

هناك العديد من الاسماء المتكررة للدوال لكن كل واحدة مختلفة في طريقة عملها وفي عدد المتغيرات (البارامترات) التي تستعملهم. مثلاً الدالتان: ExecuteSPNonQuery(string strSQLName) و ExecuteNonQuery(string strSQL) تختلفان في طريقة عملهما، الاولى تستخدم استعلام جاهز مسبقاً في برنامج ال Access لذلك كتبنا في اسم الدالة sp (procedure stored) ، والدالة الثانية تستخدم استعلام في لغة ال c# المكتوبة في برنامج ال Visualstudio .

```
public class DoQueries
{
    private static string strConnection = "Provider=
Microsoft.ACE.OLEDB.12.0;Data Source=" +
System.Web.HttpContext.Current.Server.MapPath(@"../App_Data/Trips.accdb ");

    public DoQueries()
    {
    }
    public static OleDbConnection Connection()
    {
```

```

        OleDbConnection cnn = new OleDbConnection(strConnection);
        return cnn;
    }
    //-----
    public static OleDbDataReader ExecuteReader(string strSQLName, ArrayList
prmList)
    {
        OleDbConnection cnn = Connection();

        OleDbCommand cmd = new OleDbCommand(strSQLName, cnn);
        cmd.CommandType = CommandType.StoredProcedure;

        foreach (OleDbParameter prm in prmList)
        {
            cmd.Parameters.Add(prm);
        }

        OleDbDataReader dr = null;
        cnn.Open();
        dr = cmd.ExecuteReader();
        cnn.Close();
        return dr;
    }
    public static OleDbDataReader ExecuteReader(string strSQLName)
    {
        OleDbConnection cnn = Connection();

        OleDbCommand cmd = new OleDbCommand(strSQLName, cnn);
        cmd.CommandType = CommandType.StoredProcedure;

        OleDbDataReader dr = null;
        cnn.Open();
        dr = cmd.ExecuteReader();
        cnn.Close();
        return dr;
    }
    //-----

    public static Object ExecuteSPScalar(string strSQLName, ArrayList prmList)
    {
        OleDbConnection cnn = Connection();

        OleDbCommand cmd = new OleDbCommand(strSQLName, cnn);
        cmd.CommandType = CommandType.StoredProcedure;

        foreach (OleDbParameter prm in prmList)
        {
            cmd.Parameters.Add(prm);
        }
        Object val = new Object();
        val = null;
        cnn.Open();
        val = cmd.ExecuteScalar();
        cnn.Close();

        return val;
    }

    public static Object ExecuteSPScalar(string strSQLName)
    {

```



```

        OleDbConnection cnn = Connection();

        OleDbCommand cmd = new OleDbCommand(strSQLName, cnn);
        cmd.CommandType = CommandType.StoredProcedure;

        Object val = new Object();
        val = null;

        cnn.Open();
        val = cmd.ExecuteScalar();
        cnn.Close();

        return val;
    }
    //-----
    public static Object ExecuteScalar(string strSQL)
    {
        OleDbConnection cnn = Connection();
        OleDbCommand cmd = new OleDbCommand(strSQL, cnn);

        Object val = new Object();
        val = null;

        cnn.Open();
        val = cmd.ExecuteScalar();
        cnn.Close();

        return val;
    }
    //-----
    public static bool ExecuteSPNonQuery(string strSQLName, ArrayList prmList)
    {
        OleDbConnection cnn = Connection();

        OleDbCommand cmd = new OleDbCommand(strSQLName, cnn);
        cmd.CommandType = CommandType.StoredProcedure;

        foreach (OleDbParameter prm in prmList)
        {
            cmd.Parameters.Add(prm);
        }
        bool r = false;

        cnn.Open();
        r = (cmd.ExecuteNonQuery() != 0);
        cnn.Close();

        return r;
    }
    public static bool ExecuteSPNonQuery(string strSQLName)
    {
        OleDbConnection cnn = Connection();

        OleDbCommand cmd = new OleDbCommand(strSQLName, cnn);
        cmd.CommandType = CommandType.StoredProcedure;

        bool r = false;

        cnn.Open();
        r = (cmd.ExecuteNonQuery() != 0);
    }

```

```

        cnn.Close();

        return r;
    }
    //-----
    public static bool ExecuteNonQuery(string strSQL)
    {
        OleDbConnection cnn = Connection();

        OleDbCommand cmd = new OleDbCommand(strSQL, cnn);
        bool r = false;

        cnn.Open();
        r = (cmd.ExecuteNonQuery() != 0);
        cnn.Close();

        return r;
    }
    //-----
    public static void UpdateTableInDataBase(string strSQL, DataTable dtNew)
    {
        OleDbDataAdapter adp = new OleDbDataAdapter(strSQL, strConnection);
        OleDbCommandBuilder autoCmdBuild = new OleDbCommandBuilder(adp);

        adp.UpdateCommand = autoCmdBuild.GetUpdateCommand();
        adp.DeleteCommand = autoCmdBuild.GetDeleteCommand();
        adp.InsertCommand = autoCmdBuild.GetInsertCommand();
        adp.Update(dtNew);

        dtNew.Clear();
        return;
    }
    //-----
    public static DataSet ExecuteDataSet(string strSQL)
    {
        DataSet ds = new DataSet();
        OleDbDataAdapter adp = new OleDbDataAdapter(strSQL, strConnection);

        adp.Fill(ds);

        return ds;
    }
    //-----
    public static DataTable ExecuteDataTable(string strSQL)
    {
        DataTable dt = new DataTable();
        OleDbDataAdapter adp = new OleDbDataAdapter(strSQL, strConnection);

        adp.Fill(dt);

        return dt;
    }
}

```

## User interface (web forms)





صفحات الأنترنت موجودة في 4 ملفات رئيسية (بالإضافة إلى صفحة

منفردة تدعى

HomePage، وهي بمثابة الصفحة الرئيسية في الموقع). وهذه الـ 3

ملفات هي

كالآتي:

- ▶  PagesForAdmin
- ▶  PagesForMorshd
- ▶  PagesForPerson
- ▶  PagesForVisitor

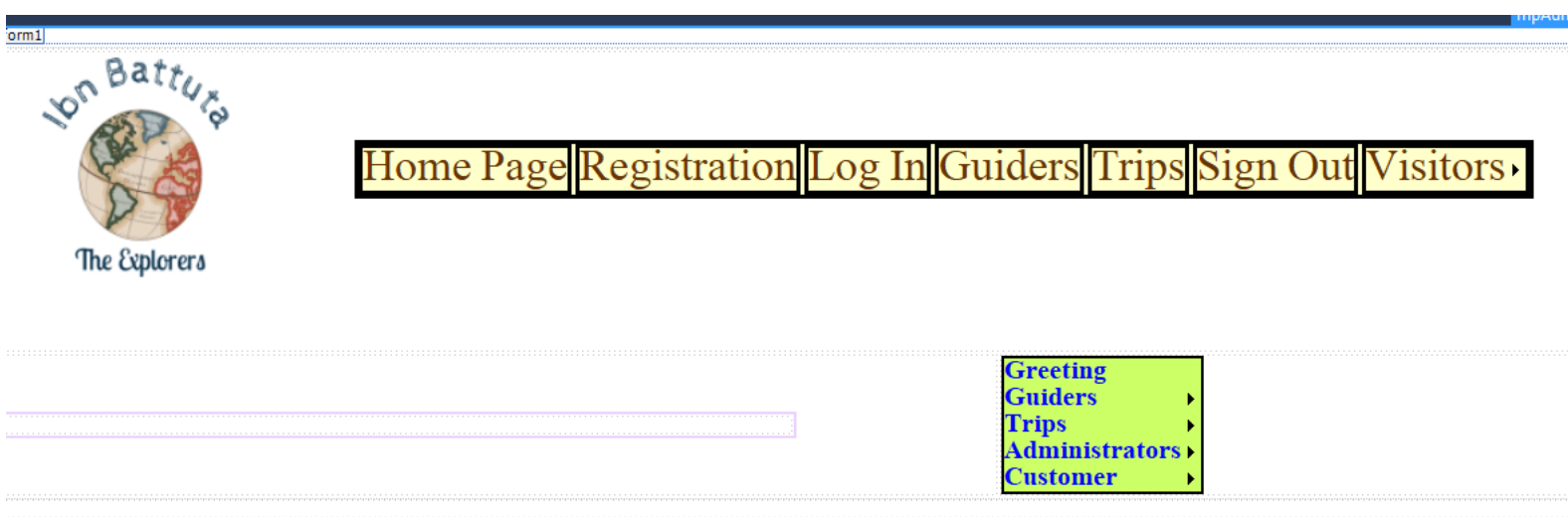
نقسم صفحات الإنترنت بهذا الأسلوب لمنع دخول المستخدم لمعلومات سرية ولا يسمح له أن يتحكم فيها، أي أن كل واحد من الـ Person، Admin، Visitor، Morshd لديه قيود ويوجد خدمات لا يسمح له بالولوج إليها (התראת גישה) وهذا لمنع تسريب المعلومات الخاصة والسرية.

## Pages For Admin :

- 📁 PagesForAdmin
  - ▶ 📄 mpAdmin.master
  - ▶ 📄 wfAddAdmin.aspx
  - ▶ 📄 wfAddMorshd.aspx
  - ▶ 📄 wfAddTrip.aspx
  - ▶ 📄 wfGreetingAdmin.aspx
  - ▶ 📄 wfUpdDelAdmin.aspx
  - ▶ 📄 wfUpdDelMorshd.aspx
  - ▶ 📄 wfUpdDelPerson.aspx
  - ▶ 📄 wfUpdDelTrip.aspx

هذه الصفحات عبارة عن الملفات التي يستخدمها مدير الموقع، ومن خلالها نرى أن لديه سيطرة على الموقع وأن لديه ولوج لمعلومات أكثر وخدمات أكثر

## mpAdmin (master page for admins):



هذه الصفحة هي Page master لجميع ال web forms الخاصة بالمدير، وهي من مركبة wucAdmin ، wucHeader ، Image (logo)، ContentPlaceHolder.

### شرح عن صفحات الإنترنت في الملف:

- PagesForAdmin تستخدم نفس صفحة ال master Page

اعلاه.

- wfAddAdmin – صفحة انترنت لاضافة مدير جديد للموقع.
- wfAddMorshd – صفحة انترنت لاضافة مرشد جديد.
- wfAddTrip – صفحة انترنت لاضافة رحلة جديدة.
- wfUpdDelAdmin - صفحة انترنت لكي يحتلن المدير معلوماته الشخصية او يحذفها
- wfGreetingAdmin - صفحة انترنت للترحيب بالمدير.
- wfUpdDelMorshd - صفحة انترنت تتيح للمدير إمكانية حتلنة تفاصيل او حذف مرشد معين بقاعدة البيانات.
- wfUpdDelPerson - صفحة انترنت تتيح للمدير إمكانية حتلنة تفاصيل او حذف شخص معين بقاعدة البيانات.
- wfUpdDelTrip - صفحة انترنت تتيح للمدير إمكانية حتلنة تفاصيل او حذف رحلة معين بقاعدة البيانات.

## مثال لصفحة انترنت (Web form) من ملف PagesForAdmin:













Home Page | Registration | Log In | Guiders | Trips | Sign Out | Visitors

ContentPlaceHolder1 (Custom)  
(lblMessage)

- Error message 1.
- Error message 2.

SqlDataSource - SqlDataSource1

	Name	Email	UserName	Password
 	abc	abc	abc	abc
 	abc	abc	abc	abc
 	abc	abc	abc	abc
 	abc	abc	abc	abc
 	abc	abc	abc	abc
1 2				

Greeting  
Guiders  
Trips  
Administrators  
Customer

هذه الصفحة هي صفحة UpdDelAdmin – هدفها حذف او

حتالة معلومات مدير معين

التالي هو ملف الـ c# للصفحة:

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class PagesForAdmin_wfUpdDelAdmin : System.Web.UI.Page
{
    int editInd;
```

```

string strUserNameNew,
strPasswordNew;

protected void Page_Load(object sender, EventArgs e)
{
    if (Session["IDAdmin"] == null)
    {
        Response.Redirect("../wfStop.aspx");
    }
    lblMessage.Text = "";
    if (GridView1.Rows.Count == 0)
    {
        lblMessage.Text = "There are not administrators";
    }
}

protected bool CheckUserNamePassword(string strUserNameNew, string
strPasswordNew,
                                     string strUserNameOld, string
strPasswordOld)
{
    //UserName + Password unique
    string strSQL = "Select * From spAllUserNamesAndPasswords Where " +
        "UserName<>' " + strUserNameOld +
        "' or Password<>' " + strPasswordOld + "'";

    //UserName unique and Password unique
    //string strSQL = "Select * From spAllUserNamesAndPasswords Where " +
    //    "UserName<>' " + strUserNameOld +
    //    "' and Password<>' " + strPasswordOld + "'";
    try
    {
        DataTable dt = DoQueries.ExecuteDataTable(strSQL);
        foreach (DataRow r in dt.Rows)
        {
            if (r["UserName"].ToString() == strUserNameNew &&
                r["Password"].ToString() == strPasswordNew)
            {
                return false;
            }
            //if (r["UserName"].ToString() == strUserNameNew ||
            //    r["Password"].ToString() == strPasswordNew)
            //{
            //    return false;
            //}
        }
    }
    catch (Exception ex)
    {
        lblMessage.Text = ex.Message;
        return false;
    }
    lblMessage.Text = "";
    return true;
}

protected void CustomValidator2_ServerValidate(object source,
ServerValidateEventArgs args)
{
    editInd = GridView1.EditIndex;
    strUserNameNew =
((TextBox)GridView1.Rows[editInd].Cells[4].Controls[1]).Text;
    strPasswordNew =
((TextBox)GridView1.Rows[editInd].Cells[5].Controls[1]).Text;

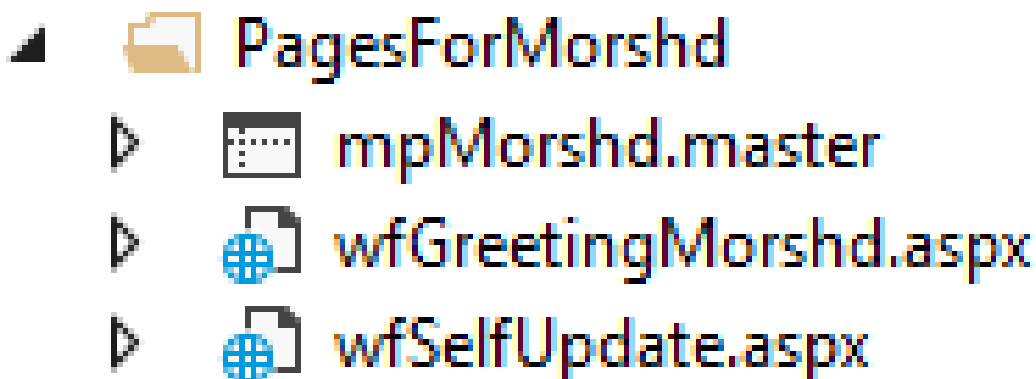
```

```

        args.IsValid = CheckUserNamePassword(strUserNameNew, strPasswordNew,
        ViewState["UserNameOld"].ToString(),
        ViewState["PasswordOld"].ToString());
    }
    protected void CustomValidator1_ServerValidate(object source,
    ServerValidateEventArgs args)
    {
        editInd = GridView1.EditIndex;
        strUserNameNew =
        ((TextBox)GridView1.Rows[editInd].Cells[4].Controls[1]).Text;
        strPasswordNew =
        ((TextBox)GridView1.Rows[editInd].Cells[5].Controls[1]).Text;
        args.IsValid = CheckUserNamePassword(strUserNameNew, strPasswordNew,
        ViewState["UserNameOld"].ToString(),
        ViewState["PasswordOld"].ToString());
    }
    protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
    {
        ViewState["UserNameOld"] =
        ((Label)GridView1.Rows[e.NewEditIndex].Cells[4].Controls[1]).Text;
        ViewState["PasswordOld"] =
        ((Label)GridView1.Rows[e.NewEditIndex].Cells[5].Controls[1]).Text;
    }
}

```

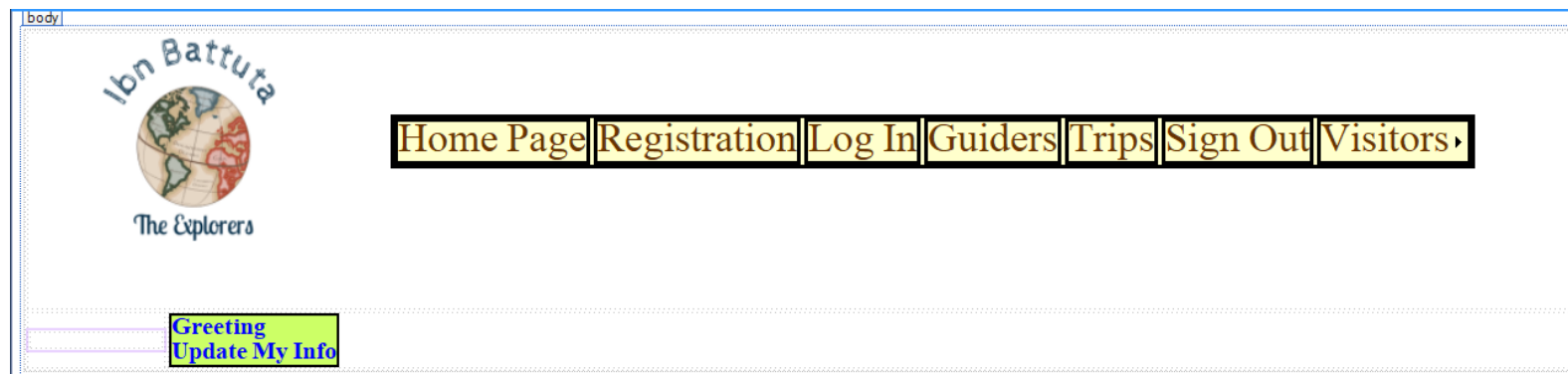
## Pages For Morshd :



هذا الملف عبارة عن الصفحات التي فقط المرشد يمكنه استخدامها.



## mpMorshd (master page for Morshds):



هذه الصفحة هي ال page master لكل صفحات الانترنت (forms web) التي  
في ملف PagesForMorshd.

### شرح عن صفحات الإنترنت في الملف:

**WfGreetingMorshd**: صفحة انترنت للترحيب بالمرشد.

**WfSelfUpdate**: صفحة انترنت كي يستطيع المرشد تعديل تفاصيله الشخصية

## مثال لصفحة انترنت من الملف PagesForMorsd : (wfSelfUpdate)

mpM




Home Page | Registration | Log In | Guiders | Trips | Sign Out | Visitors »

ContentPlaceholder1 (Custom)

Update My Data


**Guider**


**Birthday**

**Upload Picture**  Browse... 

**User Name**

**Password**





[lblMessage]  
 Required Field  
 Required Field  
 Wrong Date  
 [lblMUpload]  
 Required Field  
 [lblMUserNamePassword]  
 4-10 letters and digits  
 Required Field  
 4-10 letters and digits  
 • Error message 1.  
 • Error message 2.

Greeting Update My Info

## كود الصفحة:

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;

public partial class PagesForMorshd_wfSelfUpdate : System.Web.UI.Page
{
    DataTable dt = new DataTable();
    //TimeSpan bt = new TimeSpan(); - שעה/תאריך בטבלה
    DateTime bt = new DateTime();
    string strSQL;
    protected void FillDataTableAndFields()
    {
        strSQL = "Select * From tblMorshd Where IDMorshd=" +
```

```

        Session["IDMorshd"].ToString();
    try
    {
        dt = DoQueries.ExecuteDataTable(strSQL);
    }
    catch (Exception ex)
    {
        lblMessage.Text = ex.Message;
        return;
    }
    tbName.Text = dt.Rows[0]["Name"].ToString();
    bt = DateTime.Parse(dt.Rows[0]["Birthday"].ToString());
    imgPic.ImageUrl = @"../Images/" + dt.Rows[0]["Picture"].ToString();
    tbUserName.Text = dt.Rows[0]["UserName"].ToString();
    tbPassword.Text = dt.Rows[0]["Password"].ToString();
    ViewState["dt"] = dt;
    ViewState["strSQL"] = strSQL;
}
protected void ForRowUpdating()
{
    dt = (DataTable)ViewState["dt"];
    dt.Rows[0]["Name"] = tbName.Text;
    dt.Rows[0]["UserName"] = tbUserName.Text;
    dt.Rows[0]["Password"] = tbPassword.Text;
    dt.Rows[0]["BirthTime"] = tbBirthday.Text;

    //--Upload
    string fn = dt.Rows[0]["Picture"].ToString(), saveLocation;
    if (fuPic.PostedFile.ContentLength != 0 &&
        fuPic.PostedFile != null)
    {
        try
        {
            fn = Path.GetFileName(fuPic.PostedFile.FileName);
            saveLocation = Server.MapPath(@"../Images/") + fn;
            fuPic.PostedFile.SaveAs(saveLocation);
            lblMUpload.Text = "The File has been uploaded";
            imgPic.ImageUrl = @"../Images/" + fn;
        }
        catch (Exception ex)
        {
            lblMUpload.Text = "Error : " + ex.Message;
        }
    }
    dt.Rows[0]["Picture"] = fn;

    //--Update
    try
    {
        strSQL = ViewState["strSQL"].ToString();
        DoQueries.UpdateTableInDataBase(strSQL, dt);
    }
    catch (Exception ex)
    {
        lblMessage.Text = ex.Message;
    }
}

protected bool CheckUserNamePassword()
{
    dt = (DataTable)ViewState["dt"];

```

```

        string strUserNameNew = tbUserName.Text, strPasswordNew =
tbPassword.Text;
        string strUserNameOld = dt.Rows[0]["UserName"].ToString(),
            strPasswordOld = dt.Rows[0]["Password"].ToString();
        string strSQL = "Select * From spAllUserNamesAndPasswords Where" +
            " UserName<>'"+ strUserNameOld +
            "' or Password<>'"+ strPasswordOld + "'";

        try
        {
            DataTable dtUP = DoQueries.ExecuteDataTable(strSQL);
            foreach (DataRow r in dtUP.Rows)
            {
                if (r["UserName"].ToString() == strUserNameNew &&
                    r["Password"].ToString() == strPasswordNew)
                {
                    lblUserNamePassword.Text = "UserName and/or Password
already exist";
                    return false;
                }
            }
        }
        catch (Exception ex)
        {
            lblMessage.Text = ex.Message;
            return false;
        }
        lblMessage.Text = "";
        return true;
    }

    private bool Check()
    {
        bool chk = true;

        chk = chk && CheckUserNamePassword();
        return chk;
    }

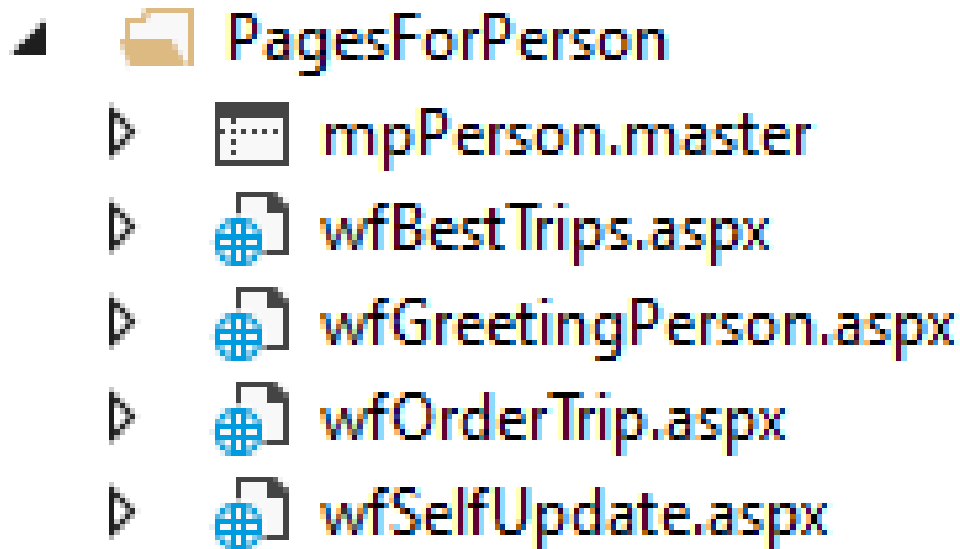
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["IDMorshd"] == null)
        {
            Response.Redirect("../wfStop.aspx");
        }
        if (!Page.IsPostBack)
        {
            FillDataTableAndFields();
        }
        lblMessage.Text = "";
        lblMUpload.Text = "";
        lblMUserNamePassword.Text = "";
    }

    protected void ibtClear_Click(object sender, ImageClickEventArgs e)
    {
        lblMessage.Text = "";
        lblMUpload.Text = "";
        lblMUserNamePassword.Text = "";
        FillDataTableAndFields();
    }

```

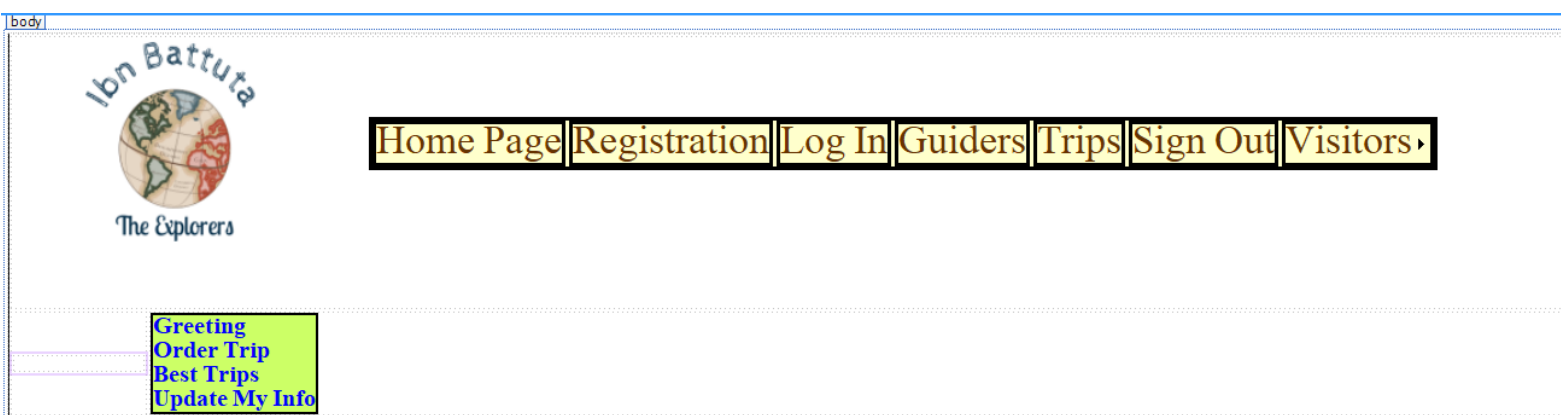
```
protected void ibtUpdate_Click(object sender, ImageClickEventArgs e)
{
    if (Check()) ForRowUpdating();
    FillDataTableAndFields();
}
}
```

## Pages For Person :



هذا الملف عبارة عن الصفحات التي فقط المستخدم يمكنه استخدامها.

## MpPerson (master page for Persons):



هذه الصفحة هي ال page master لكل صفحات الانترنت (forms web) التي

في ملف PagesForMorshd

شرح عن صفحات الإنترنت في الملف:

wfBestTrips: صفحة انترنت تعرض افضل الرحل

wfGreetingPerson: صفحة انترنت للترحيب

بالمستخدم

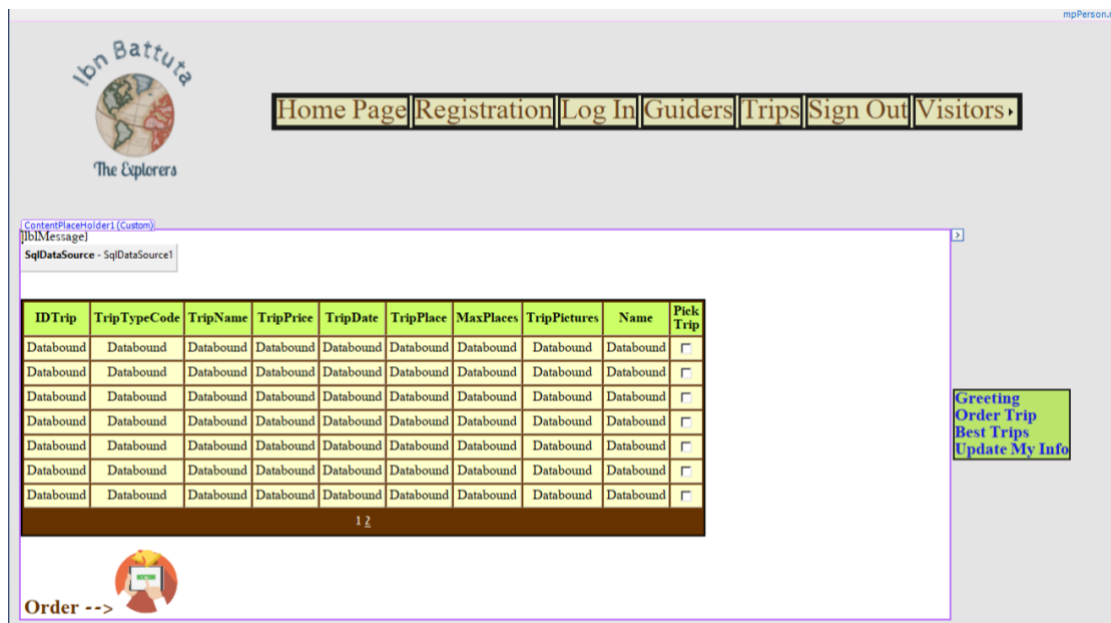
wfOrderTrip: صفحة انترنت للترحيب بالمستخدم

wfSelfUpdate: صفحة انترنت تتيح للمستخدم

إمكانية تعديل معلوماته الشخصية

مثال لصفحة انترنت من الملف PagesForPerson:

(wfSelfUpdate)



using System;

## كود الصفحة:

```
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class PagesForPerson_wfOrderTrip : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["IDPerson"] == null)
        {
            Response.Redirect("../wfStop.aspx");
        }
        lblMessage.Text = "";
        if (GridView1.Rows.Count == 0)
        {
            lblMessage.Text = "There are no trips for order";
        }
    }







    protected void ibtOrder_Click(object sender, EventArgs e)
    {
        OrderTripDetails ordDetails = new OrderTripDetails();
        OrderTrip ord = new OrderTrip();
        bool chkNew;
        ordDetails.IDPerson = int.Parse(Session["IDPerson"].ToString());

        for (int i = 0; i < GridView1.Rows.Count; i++)
        {
            chkNew =
            ((CheckBox)GridView1.Rows[i].Cells[10].Controls[1]).Checked;
            if (chkNew)
            {
                ordDetails.IDTrip =
                int.Parse(((Label)GridView1.Rows[i].Cells[0].Controls[1]).Text);
                try
                {
                    if (!ord.ExistOrder(ordDetails) && chkNew)
                    {
                        ord.AddOrder(ordDetails);
                        lblMessage.Text = "The Trip Ordered Successfully";
                    }
                }
                catch (Exception ex)
                {
                    lblMessage.Text = ex.Message;
                }
            }
        }
        Response.Redirect(@"../PagesForPerson/wfOrderTrip.aspx");
    }
}
```

Pages For Visitor :



## PagesForVisitor

- ▶  mpVisitor.master
- ▶  wfLogin.aspx
- ▶  wfMorshd.aspx
- ▶  wfRegistration.aspx
- ▶  wfSignOut.aspx
- ▶  wfTrip.aspx

في هذا الملف يوجد صفحات الانترنت التي يمكن للزائر (مستخدم للموقع لكنه لم يسجل في قاعدة البيانات).

### mpVisitor (master page for Visitors):



Home Page	Registration	Log In	Guiders	Trips	Sign Out	Visitors
-----------	--------------	--------	---------	-------	----------	----------

صفحة ال page master هذه تركز عليها جميع صفحات الانترنت التي في الملف PagesForVisitor.




- wfLogin - صفحة انترنت لتسجيل دخول المستخدم.
- wfRegistration - صفحة انترنت لتسجيل الزائر نفسه في قاعدة البيانات.
- wfSignOut - صفحة انترنت لكي يخرج المستخدم من الموقع (لكي ال تبقى تفاصيله محفوظة).

## مثال لصفحة انترنت من الملف PagesForVisitor:

### wfRegistration

D:\Users\user\Desktop\212680334\فريقتي\PagesForVisitor\wfRegistration.aspx\*



Home Page | Registration | Log In | Guiders | Trips | Sign Out | Visitors »

New Customer		
TZ	<input type="text"/>	[lblMessage] Required Field Wrong TZ
FirstName	<input type="text"/>	Required Field Wrong First Name
LastName	<input type="text"/>	Required Field Wrong Last Name
Birthday	<input type="text"/>	[lblMBirthday] Required Field Wrong Date
Address	<input type="text"/>	Required Field
Phone	<input type="text"/>	Wrong Phone
Email	<input type="text"/>	Required Field Wrong Email
UserName	<input type="text"/>	Required Field 4-10 letters and digits
Password	<input type="password"/>	[lblMUserNamePassword] Required Field 4-10 letters and digits
<input type="button" value="⊕"/>		<input type="button" value="✕"/> <ul style="list-style-type: none"> <li>• Error message 1.</li> <li>• Error message 2.</li> </ul>

### كود الصفحة:

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
```

```

using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class PagesForVisitor_wfRegistration : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblMessage.Text = "";
        lblMBirthday.Text = "";
        lblMUserNamePassword.Text = "";
    }
    protected void ForRowInserting()
    {
        PersonDetails prsDetails = new PersonDetails();
        Person prs = new Person();

        prsDetails.TZ = tbTZ.Text;
        prsDetails.FirstName = tbFirstName.Text;
        prsDetails.LastName = tbLastName.Text;
        prsDetails.Birthday = DateTime.Parse(tbBirthday.Text);
        prsDetails.Address = tbAddress.Text;
        prsDetails.Phone = tbPhone.Text;
        prsDetails.Email = tbEmail.Text;
        prsDetails.UserName = tbUserName.Text;
        prsDetails.Password = tbPassword.Text;

        try
        {
            if (!prs.ExistPerson(prsDetails))
            {
                prs.AddPerson(prsDetails);
                lblMessage.Text = "The Person added successfully";
            }
            else
            {
                lblMessage.Text = "The Person already exist";
            }
        }
        catch (Exception ex)
        {
            lblMessage.Text = ex.Message;
        }
    }
    protected bool CheckUserNamePassword()
    {
        string strSQL = "Select * From spAllUserNamesAndPasswords";
        try
        {
            DataTable dt = DoQueries.ExecuteDataTable(strSQL);
            foreach (DataRow r in dt.Rows)
            {
                if (r["UserName"].ToString() == tbUserName.Text &&
                    r["Password"].ToString() == tbPassword.Text)
                {
                    lblMUserNamePassword.Text = "UserName and/or Password
already exist";
                    return false;
                }
            }
        }
        catch (Exception ex)
    }
}

```

```

        {
            lblMessage.Text = ex.Message;
            return false;
        }
        return true;
    }
}
protected bool CheckAge(int intAge, string strBirthday)
{
    DateTime dtb = DateTime.Parse(strBirthday);
    DateTime dtnow = DateTime.Now;
    DateTime dth = new DateTime(dtb.Year + intAge, dtb.Month, dtb.Day);
    return dth.CompareTo(dtnow) == -1 || dth.CompareTo(dtnow) == 0;
}
private bool Check()
{
    bool chk = true;

    //Code for specific check
    if (!CheckAge(18, tbBirthday.Text))
    {
        lblMBirthday.Text = "You are too young";
        chk = false;
    }

    chk = chk && CheckUserNamePassword();
    return chk;
}
protected void ibtAdd_Click(object sender, ImageClickEventArgs e)
{
    if (Check()) ForRowInserting();
}
protected void ibtClear_Click(object sender, ImageClickEventArgs e)
{
    lblMessage.Text = "";
    lblMBirthday.Text = "";
    lblMUserNamePassword.Text = "";

    //Emptying Textbox Controls (1)
    //tbTZ.Text = "";
    //tbFirstName.Text = "";
    //tbLastName.Text = "";
    //tbBirthday.Text = "";
    //tbAddress.Text = "";
    //tbPhone.Text = "";
    //tbEmail.Text = "";
    //tbUserName.Text = "";
    //tbPassword.Text = "";

    //Emptying Textbox Controls (2)
    foreach (Control c in this.Form.Controls)
    {
        if (c is ContentPlaceholder)
        {
            foreach (Control cc in c.Controls)
            {
                if (cc is TextBox)
                {
                    ((TextBox)cc).Text = "";
                }
            }
        }
    }
}

```

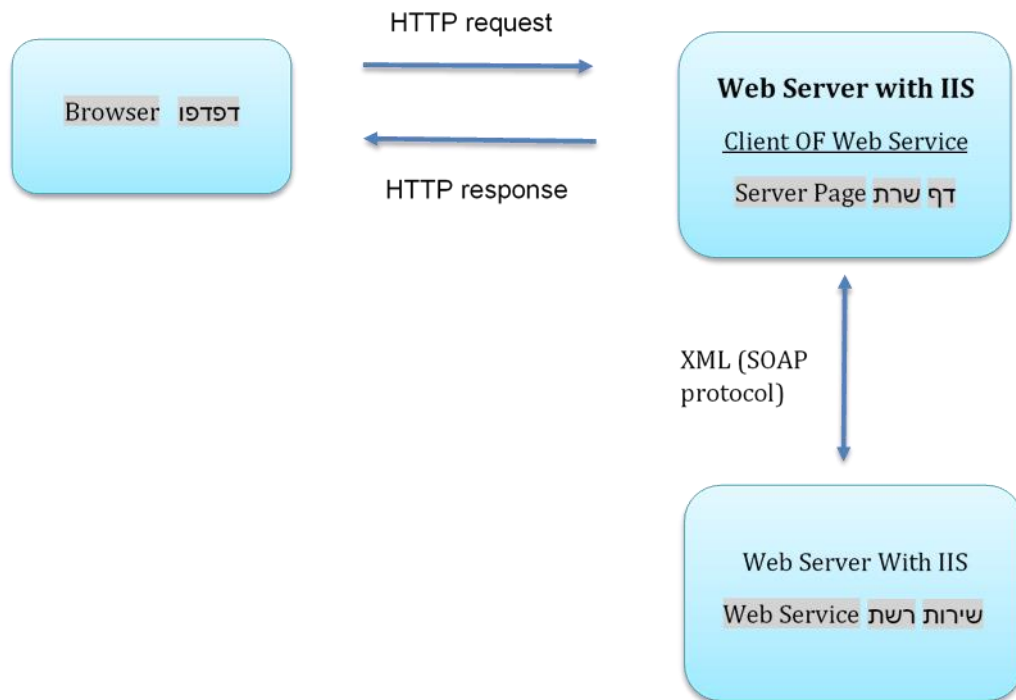
## web service

بالإضافة إلى صفحات الانترنت العديدة التي يمكن أن نستخدمها في الموقع، بإمكاننا استخدام خدمة تدعى web service وهي عبارة عن استغلال خدمات "بعيدة"، أي في موقع أو مكان آخر، وتطبيقها في الموقع الخاص بنا.

خدمات الانترنت من نوع XML web service هي تطبيق ASP.NET ، وعوضاً عن استخدام ومعالجة صفحات انترنت من نوع ASPX، نتعامل مع طلبات صفحات انترنت من نوع ASMX. قبل استخدام صفحات انترنت من نوع ASMX نضطر لبناء تطبيق من نوع ASP.NET website وبعدها اضافة صفحات ASMX لهذا التطبيق. بالإمكان أيضاً استعماله في تطبيق ال ASP.NET، وبعدها اضافة عليه صفحات ال ASMX.

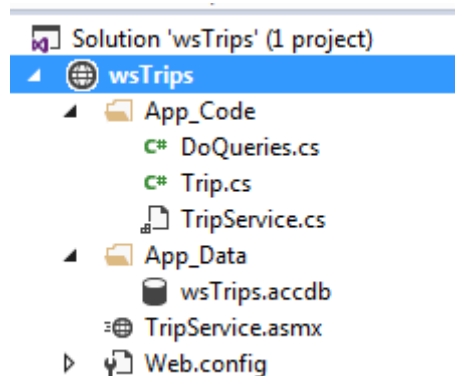
يمكن القول أن خدمة الانترنت XML web service هي تطبيق ASP.NET عادي الذي يحتوي على فئة (class) تتيح لتطبيقات كثيرة الولوج لخدماتها من بعيد، هذه الخدمات معروفة باسم web methods. الخدمات هذه عبارة عن دوال معينة. يمكن تنفيذ الدوال web methods التابعة ل XML web service عن طريق كل صفحة انترنت من نوع ASPX التابعة لتطبيق ASP.NET web application. في هذه الحالة يمكن القول أن التطبيق يعمل كمستهلك ل web service وهو يدعى client web service.

أمامك رسم مخطط يشرح عن عملية حدوث ال web service :



في الموقع Web service :

لاستخدام שירות הרשת في موقعي، قمت ببناء خدمة في موقع web service



## Trip.cs

```
public class Trip
{
    private string _TripID;
    private string _TripName;
    private string _TripPlace;
    private string _TripDate;

    //Properties

    public string TripID
    {
        get { return _TripID; }
        set { _TripID = value; }
    }
    public string TripName
    {
        get { return _TripName; }
        set { _TripName = value; }
    }
    public string TripPlace
    {
        get { return _TripPlace; }
        set { _TripPlace = value; }
    }

    public string TripDate
    {
        get { return _TripDate; }
        set { _TripDate = value; }
    }
    public Trip()
    {
        //
        // TODO: Add constructor logic here
        //
    }
}
```

## TripService.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;
using System.Collections;
using System.Collections.Generic;
using System.Web.Services;
```

```

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX,
// uncomment the following line.
// [System.Web.Script.Services.ScriptService]

public class TripService : System.Web.Services.WebService
{
    public TripService() {
        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public bool ExistTrip(string ID)
    {
        string str = string.Format("SELECT * FROM Trip WHERE TripID='{0}'",
ID);

        DataSet ds = DoQueries.ExecuteDataSet(str);
        if (ds.Tables[0].Rows.Count == 0)
            return false;
        return true;
    }

    [WebMethod]
    public void UpdateTripDetails(Trip t)
    {
        string str = string.Format("UPDATE Trip SET
TripName='{0}',TripPlace='{1}',TripDate='{2}' WHERE TripID='{3}'", t.TripName,
t.TripPlace, t.TripDate, t.TripID);

        if (ExistTrip(t.TripID) == true)
        {
            try
            {
                DoQueries.ExecuteNonQuery(str);
            }
            catch (Exception)
            {
                throw;
            }
        }
    }

    //*****8

    [WebMethod]
    public void DeleteTripDetails(string ID)
    {
        if (ExistTrip(ID) == true)
        {

```

```

        try
        {
            string str = string.Format("DELETE FROM Trip WHERE
TripID='{0}'", ID);
            DoQueries.ExecuteNonQuery(str);
        }
        catch (Exception)
        {
            throw;
        }
    }

}

//*****
[WebMethod]
public void AddNewTrip(string ID, string Name, string Place, string Date)
{

    if (ExistTrip(ID) == false)
    {

        try
        {
            string str = string.Format("INSERT INTO
Trip(TripID,TripName,TripPlace,TripDate)VALUES('{0}','{1}','{2}','{3}')" , ID,
Name, Place, Date);
            DoQueries.ExecuteNonQuery(str);
        }
        catch (Exception)
        {
            throw;
        }
    }
}

//*****

[WebMethod]
public DataSet GetAllTrips()
{
    string str = string.Format("SELECT * FROM Trip");

    return DoQueries.ExecuteDataSet(str);
}

//*****

[WebMethod]
public Trip GetTripById(string ID)
{

    Trip e = new Trip();

    if (ExistTrip(ID))
    {

        string strDetails = string.Format("SELECT * FROM Trip WHERE
TripID='{0}'", ID);

```



```

        DataSet ds = DoQueries.ExecuteDataSet(strDetails);

        e.TripID = ds.Tables[0].Rows[0]["TripID"].ToString();
        e.TripName = ds.Tables[0].Rows[0]["TripName"].ToString();
        e.TripPlace = ds.Tables[0].Rows[0]["TripPlace"].ToString();
        e.TripDate = ds.Tables[0].Rows[0]["TripDate"].ToString();
    }
    return e;
}

//*****
}

```

## App\_Data:

تحتوي على قاعدة البيانات التي نستعملها في الموقع ال web service، وهي نسخة عن قاعدة البيانات التي نستخدمها في موقع ال (ASP.NET web site) الموقع الرئيسي، لأن هذا الموقع هو موقع جانبي لتطبيق فكرة ال web service .

## Web Services:

### TripService

#### TripService

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [AddNewTrip](#)
- [DeleteTripDetails](#)
- [ExistTrip](#)
- [GetAllTrips](#)
- [GetTripById](#)
- [UpdateTripDetails](#)

## wfAddTrip

**Add Trip** **Update Trip** **Delete Trip** **All Trips** **Trip By ID**

Enter Trip ID   
Enter Trip Name   
Enter Trip Place   
Enter Trip Date

**Add New Trip**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

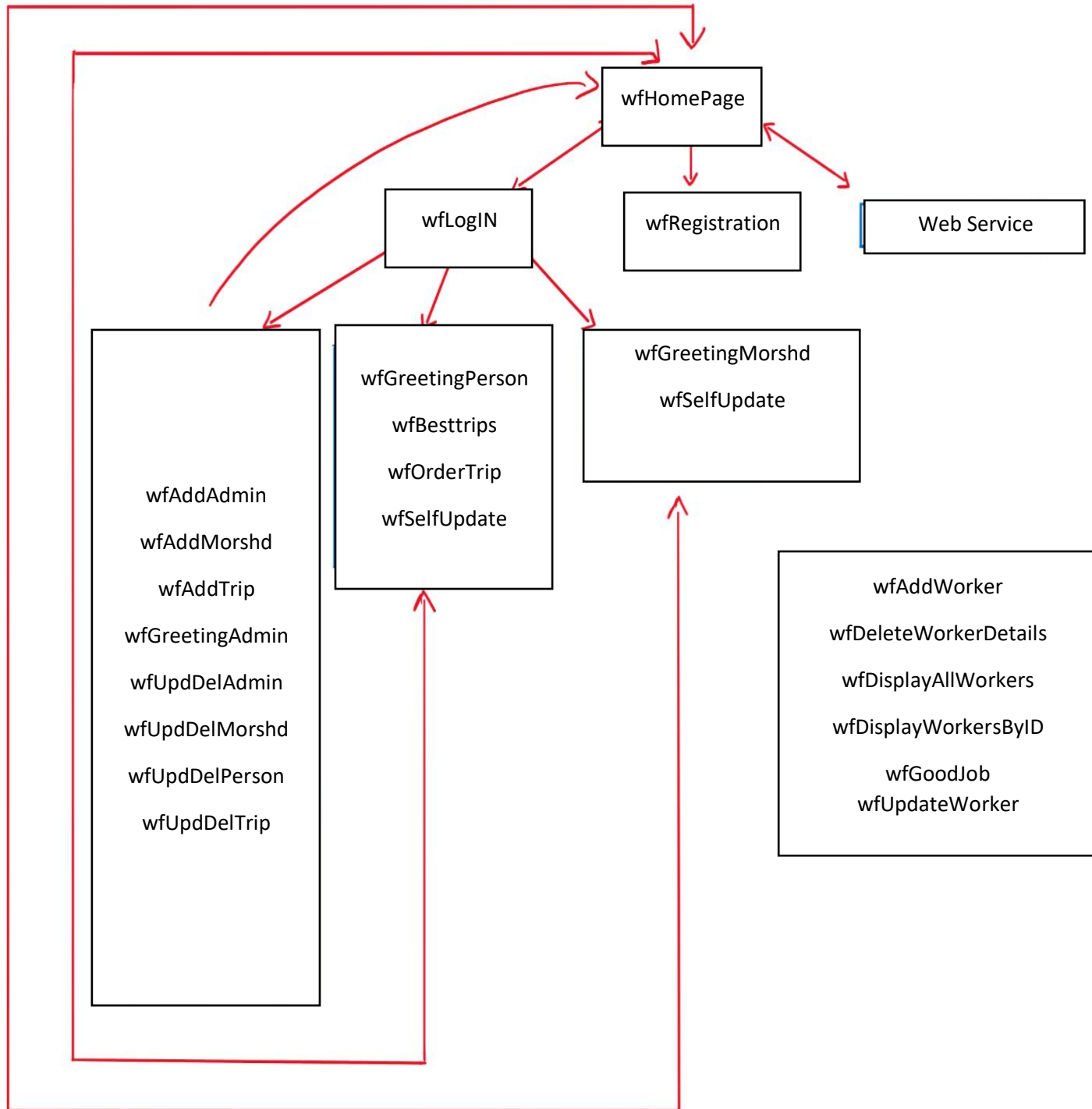
public partial class PagesForConsumingTrips_wfAddTrip : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void btAddTrip_Click(object sender, EventArgs e)
    {
        wsTrip.TripService t = new wsTrip.TripService();

        if (!t.ExistTrip(tbID.Text))
        {
            t.AddNewTrip(tbID.Text, tbName.Text, tbPlace.Text, tbDate.Text);

            Response.Redirect("../PagesForConsumingTrips/wfGoodJob.aspx");
        }
        else
        {
            lblMessage.Visible = true;
            lblMessage.Text = "The Trip already exist";
        }
    }
}
```

# Site Map



## מדריך משתמש

### מדריך למשתמש ( Customer ) :

المستخدم بإمكانه التسجيل للموقع التصفح بالرحل المتاحة، افضل الرحل، المرشدين وحجز رحلة من الرحل المتاحة.

### מדריך למשתמש ( Guider ) :

المرشد يقوم بالتسجيل للموقع ويتم اضافته لرحل معينة ليكون مرشدا لها وبهذا يجد عملا ورحلا ليخرج معها من خلال الموقع.

### מדריך למנהל ( Admin ) :

المدير له صلاحيات اكثر في الموقع فبإمكانه إضافة مرشد، تعديل تفاصيل المرشد، إضافة مدير اخر للموقع، تعديل تفاصيل المدير، تعديل تفاصيل المستخدم، إضافة رحل للموقع، تعديل تفاصيل للرحل.

שם משתמש ל ( Customer ) : Tareksh

סיסמה ל ( Customer ) : 1234

שם משתמש ל ( Guider ) : Yasmeen

שם סיסמה ל ( Guider ) : 12345

שם משתמש ל ( Admin ) : tarek

שם סיסמה ל ( Admin ) : 1234