WCF

רקע

שיטה לבניית service-ים

service יכול לשמש גם לעדכון class library שיש per יחסית אל references

- control-ים ב MVC הם כמו service
- web API
- WCF

Server Side		Client Side
	web services	
C#	MVC	
JAVA	WebAPI	
PHP	WCF	
	שרת אינטרנט	טכנולוגיה
	Asp.Net	JEE
	Asp	Jsp

new project -> WCF Service Application

נשמרת סטטוס service קבוצה עם כל הנתונים

attribute מיוחד שמאפשר לנו בונקציות מקומיות לשים עליהן serviceContract

- פרטי המידע שמורים במסד נתונים
  - טבלת עיר: קוד, שם עיר
  - טבלת ייעוד (מבוגרים, ילדים, משפחתי): קוד ייעוד, שם ייעוד
  - טבלת אתר: קוד, שם, קוד ייעוד, קוד עיר, מחיר כניסה.
  - טבלת מסלול הליכה- יורשת מטבלת אתר: אורך המסלול, מוצל(כן/לא)

הפונקציות בService:

- פונקציה המחזירה את מסלול ההליכה הארוך ביותר.
- פונקציה GetCities, המחזירה את רשימת כל הערים.
- פונקציה GetTargets, המחזירה את רשימת כל הייעודים.
- פונקציה GetSites(1), המקבלת עיר, ומחזירה רשימה של מסלולים ואתרי הליכה בעיר שהתקבלה (KnownType).
- פונקציה GetSites(2), המקבלת ייעוד ומחזירה רשימה של אתרים ומסלולי הליכה התואמים לייעוד שהתקבל.

2. ה*Client*

צרי טופס המציג מידע על אתרים.

אפשרי את הפעולות הבאות:

- רשימת הערים תוצג בComboBox, בבחירת עיר תוצג רשימת אתרים בDataGridView.
- רשימת הייעודים תוצג בComboBox , בבחירת ייעוד תוצג רשימת אתרים בDataGridView.
- הציגי את פרטי מסלול ההליכה הארוך ביותר.

בהצלחה!

בס"ד

WCF – ראשי פרקים

- **הקדמה:** מושגים בסיסים והצגת הצורך בסרויס כולל סקירה של מגוון צורות במשק ליצירת סרויסים.
- מהו WCF?
  - Windows Communication Foundation

    בסיס לפיתוח יישומי תקשורת, נועד ליישומי Windows
  - הודעות מועברות מהclient אל הserver ולהפך, בפורמט טקסטואלי

יצירת שירות בסיסי/ צד השרת

איך יוצרים Service? (נתאר חלק מהשיטות)

1. יצירת פרויקט מסוג WCF. בפרויקט מסוג זה יש אלמנט מסוג svc , ובהרצתו מוצג tester המאפשר לנסות את הפונקציות.
2. יצירת פרויקט רגיל – שאליו מוסיפים service , נוצר Config שונה מהצורה הקודמת ויש להריצו לבד עי כתיבת קוד.

מה מכיל Service?

1. הפניה לספריית System.ServiceModeService
2. Contract – ממשק המכיל את הפונקציות החשופות למשתמש.
   - הAttribute [ServiceContract] - מופיע מעל הממשק , ומציין שבממשק זה מוגדרות פונקציות החשופות לקריאות של מחשב שונה.
   - הAttribute [OperationContract] – מופיע מעל פונקציות, ומציין שפונקציה זו חשופה למשתמש.
3. Service – מחלקה שמממשת את הממשק הקודם
4. קובץ Config המכיל את ההגדרות השונות של השרות.

בתוך תגית בשם Services מופיעים חלקים שונים של ההגדרה:

- EndPoint

- Address - היכן השירות נמצא?
- Binding - איך תתבצע ההתקשרות אל השירות? (מי פרוטוקול התקשורת, מהי צורת העברת הנתונים, הגדרות אבטחה ועוד)
- Contract – שם הממשק שמכיל את הפונקציות שהשירות מספק.

- בניית קובץ קונפיגורציה- (נבנה אוטומטית ע"י V. S .Net). יכיל הגדרות תצורה
- הכרות עם קובץ הקונפיגורציה (שנבנה אוטומטית ומתאים לצורה השניה של יצירת הסרויס)

```
<system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="">
          <serviceMetadata httpGetEnabled="true"
                  httpGetUrl="http://localhost:8412/Design_Time_Addresses/
                                                    BaseService/Calc/Help"/>
                  <!--כדי לחשוף את הwsdl של השירות--!>
          <serviceDebug includeExceptionDetailInFaults="false" />
           <!--האם להכליל בשגיאה שהלקוח מקבל את פרטי השגיאה--!>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service name="BaseService.Calc"> <!--Namespace.ClassName-->
        <endpoint address="" binding="wsHttpBinding"
                                contract="BaseService.ICalc">
                                      <!--Namespace.InterfaceName-->
          <identity> <dns value="localhost" />
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding"
                                          contract="IMetadataExchange" />
        <host>
          <baseAddresses>
            <add baseAddress = "http://localhost:8412/Design_Time_Addresses/
                                                    BaseService/Calc/" />
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
```

- Host - האפליקציה המארחת את השירות

ביצירת Service בצורה הראשונה אין צורך בו, כי הוא מוטבע בפרויקט. בצורה השנייה יש צורך מאחר ושירות שאינו מורץ בפועל א"א לגשת לפונקציות שהוא מציע

איך עושים זאת?

יצירת מופע של ServiceHost. (נשלח לc-tor: (typeof(ServiceClass)

- Host.Open(); המארח פותח את השירות, ו"מעלה אותו לאוויר"
- יש לדאוג שהשירות ימשיך לרוץ (Console.Read();)

<u>יצירת שירות בסיסי/ צד הלקוח</u>

- בניית Client
  - סוג הפרוייקט- כל אפליקציה, בהתאם לצורך (צרכן השירות – תוכנה/ אתר/ אפליקציה ולא משתמש)
  - השירות חייב להיות מורץ לצורך זיהויו (כאשר הclient נמצא בSolution של הService, יש להריץ את קובץ הexe. מתוך תיקיית הbin/debug בתיקייה של פרוייקט הhost)
  - References/ Add Service Reference
  - בתיבת הטקסט Address:

`http://localhost:8412/Design_Time_Addresses/BaesService/Calc/[help]`

(כתובת השירות כפי שהיא רשומה ב`baseAddress`, או ב`httpGetUrl`) לצורך כך על הClient להכיר את הגדרות הendpoint שדרכן השירות נחשף

  - Go
  - בתיבת הטקסט Namespace: `MyServiceCalc` (השם שבחרנו לשירות)
  - נוצרה תיקייה ServiceReferences ובתוכה תיקייה בשם שנבחר לשירות
  - (`MyServiceCalc`) שם השירות.using namespace
  - הגדרת מופע של מחלקת השירות

ClaasNameClient c=new ClassNameClient();

  - שימוש בפונקציות השירות
  - הרצת הService והClient יחד, Multiple startup projects

העברת נתונים

תקן WSDL: Web Description Language

- תקן כללי, ללא קשר לשפה מסוימת. לפיכך יש לבדוק לגבי כל סוג האם הוא נתמך. מאחר הערכים המוחזרים מהפונקציות עוברים ברשת בפורמט טקסטואלי , ומאחר wcfו מעונין לתמוך במגוון שפות ישנות- לא כל סוגי הנתונים נתמכים.

להלן סקירת טיפוסי הנתונים השונים והדרך שהסרויס מתמודד איתם.

1. טיפוסים פרימיטיביים- עוברים
2. מערכים- עוברים
3. מטריצה- לא ניתן להעביר. במקום זה מחזירים Jugged Arrays, מערך של מערכים. (int[][] mat)
4. רשימות (List)- מתקבלת בלקוח כמערך, והוא יכול להמיר לרשימה. ניתן לשנות בConfigure Service Reference, בחלק של Data Type\Collection type שיתקבל כרשימה (ואז ניתן להמיר למערך)
5. Enums – עוברים, ללא צורך בAttributes. אם מעוניינים שחלק מהאופציות יהיו מוסתרות, נשתמש בAttribute [DataContract] מעל enum, וAttribute [EnumMember] מעל לכל אופציה שנרצה לחשוף. אפשרויות שלא נצמיד אליהן את ה Attribute הזה לא יחשפו.
6. (DataTable- כדי שיעבור יש להקפיד לתת שם בתכונה TableName)
7. מחלקה, מוגדרת בשירות- ניתן לרשום מעל הגדרת המחלקה את הAttribute [DataContract], ומעל כל member שנרצה לחשוף [DataMember], באופן כזה נוכל לחשוף גם private member
8. אובייקט פולימורפי- ייחוס של מחלקת בסיס המצביע לאובייקט ממחלקה נגזרת. כדי שיהיה ניתן לראות את רכיבי מחלקה הנגזרת בעזרת הייחוס למחלקת הבסיס, יש צורך לבצע אחת מ2 אפשרויות:

- לפני מחלקת הבסיס, להוסיף עבור כל אחת מהמחלקות היורשות:

[KnownType(typeof(DerivedClassName))

- מעל הInterface שחושף את השירות, להוסיף עבור כל אחת מהמחלקות היורשות:

[ServiceKnownType(typeof(DerivedClassName))]

9. מחלקות מוכלות. class1 מכיל member מסוג class2- בclass1 נרשום את הAttributes [DataContract], [DataMember]. בclass2 אין צורך להוסיף שום Attribute (Attributes *עוברים* בהרכבה)

10. מחלקה הממשת את Interface1, שיורש מInterface2- מעל לInterface1 יש לרשום [ServiceContract], ועבור כל פונקציה שנרצה לחשוף, נצטרך להוסיף בInterface1 את הAttribute [OperationContract], גם אם הAttributes האלה כבר רשומים בInterface2. (Attributes *לא עוברים* בהורשה)

11. ניתן להעביר פרמטרים By Ref.ש

12. Interfaces- לא ניתן להעביר, נחזיר אובייקט הממש אותו

13. Overloading Methods- כיון שלא כל השפות שיפעילו את הService תומכות באפשרות הזו, היא לא נתמכת ע"י פרוטוקול WSDL. ניסיון לארח Service המכיל Overloading ייכשל. פתרונות:

- נשנה את שם אחת מהפונקציות
- [OperationContract(Name="AnotherName")]

14. אובייקטים שנוצרו ע"י EF- כדי למנוע ייחוס מעגלי: יש לכתוב את השורה הבאה בctor של קובץ הtt

```
this.Configuration.ProxyCreationEnabled = false;
```

תרגיל

כתבי Service של מידע למטייל

1. השירות

הService יכיל את המרכיבים הבאים:

End Points

- תפקיד: חשיפת השירות לעולם החיצון
- מרכיבי הEndPoint
  - Address. היכן נמצא השירות?
    - התכונה baseAddress
  - Binding. איך תתבצע ההתקשרות? אלמנט זה מחולק לשלושה חלקים:
    - Transport - אחראי על הפרוטוקול עצמו, http, tcp, msmq ועוד.
    - Encoder - האם המידע יעבור ב - Text, Binary, Custom.
    - Protocol - אחראי על שאר הדברים כמו Security, Transactions...
  - בTransport ובEncoder יש רק אופציה אחת לבחור. לעומת זאת בProtocol אפשר לבחור יותר מאחד. ההודעה שנשלחת לשירות עוברת בין ה - Protocols השונים שנקבעו, ורק במידה שכולם מאשרים את ההודעה זה מגיע לשירות.
    - דוגמאות לסוגים מובנים- (פרוטוקול התקשורת מסומן בקו תחתי)
    - basicHttpBinding, בסיסי- ברירת המחדל, מתאים לתקשורת עם שירותי אינטרנט, המידע עובר בתקן soap (מעין xml)
    - WsHttpBinding, מאובטח ואמין- תמיכה בסטנדרט WS בנוגע לאבטחה ועוד, לא מעביר מידע בsoap ולכן מתאים לשימוש בajax
    - WsDualHttpBinding, כנ"ל בתוספת תמיכה בתקשורת דו סיטרית (CallBack)
    - NetPeerTcpBinding, מתאים לשימוש בpeer to peer, רשת קטנה ומקומית
    - NetTcpBinding עושה שימוש בהגדרות אבטחה ואימות של Windows, מתאים לשימוש ברשת מקומית, ולתקשורת דו סיטרית

- NetNamedPipeBinding מאובטח ואמין, המידע מועבר בקידוד בינארי באמצעות צינורות (pipes)
- NetMsmqBinding מאובטח, אמין ומטפל באיזון עומסים ברשת

דוגמא להגדרת Binding מותאם אישית

```
<bindings>
        <customBinding>
                <binding name="HTTPbinary">
                        <binaryMessageEncoding />
                        <httpTransport transferMode="Streamed" />
                </binding>
        </customBinding>
</bindings>
```

הגדרנו שהמידע יעבור בבינארי ובפרוטוקול http. המידע יעבור ב - stream (לדוג' לבניית נגן שמתקשר עם השרת כדי לקבל את הסרט שצריך לנגן)

כעת נצטרך להגדיר את ה - binding באלמנט service

```
<endpoint address="http://localhost:8412/MyBinaryService"
    binding="customBinding"
    bindingConfiguration="HTTPbinary"
    contract="MyService.Interface">
</endpoint>
```

- Contract. ה"חוזה", מה השירות מספק?

הגדרת מספר EndPoints לשירות אחד

- הצורך
  - מחלקת השירות מממשת מספר ממשקים- כל endpoint נחשף ע"י ממשק אחד, ומיועד בד"כ ללקוח אחר

```
<services>
    <service name="Service.Calc">
        <!--Namespace.ClassName-->
        <host>
          <baseAddresses>
            <add baseAddress ="http://localhost:8412/"/>
          </baseAddresses>
        </host>

              <!--address="http://localhost:8412/Msg"-->
        <endpoint address="Msg"
                  binding="basicHttpBinding"
```

```
                contract="Service.IMsg"></endpoint>

            <!--address="http://localhost:8412/Clc"-->
        <endpoint address="Clc"
                binding="basicHttpBinding"
                contract="Service.ICalc"></endpoint>

    </service>
</services>
```

הערה: במידה ובמאפיין address אין כתובת מלאה , המילה הכתובה שם תשורשר לסוף הערך של הbaseAddress

- סוגי binding שונים, שיכולות התקשורת שלהם שונות. כדי לאפשר לסוגי clients שונים לצרוך את השירות

```
<services>
      <service name="Service.Calc"
            behaviorConfiguration="calcServiceBehavior">
         <endpoint bindingConfiguration="webHttpBindingCalcBinding"
            behaviorConfiguration="calcEpBehavior"
            address="http://localhost:8412/MyCalcService"
            binding="webHttpBinding"
            contract="Contarcts.ICalc">
         </endpoint>
         <endpoint address="http://localhost:3333/MyBinaryCalcService"
            binding="customBinding"
            bindingConfiguration="HTTPbinary"
            contract="Contarcts.ICalc">
         </endpoint>
         <endpoint address="http://localhost:2222/MyBasicCalcService"
            binding="basicHttpBinding"
            contract="Contarcts.ICalc">
         </endpoint
      </service>
</services>
```

במקרה זה יש שלוש יציאות לאותו שירות,
אחת ב - webHttpBinding עבור ajax, אחת בבינארי, ואחת ב - basicHttp לעבודה בפרוטוקול SOAP (כמו Web Service)

בס"ד

Instancing (טיפול במשתני הסרויס)

- אפשרויות יצירת מופע חדש
  - PerCall: מופע חדש לכל *קריאה*, (בגישה פעמיים לפונ' לא נשמרים ערכי המשתנים.) ברירת המחדל.

    שימוש:

    * כאשר לא נדרשת שמירת מידע בנוגע להתקשרות.

    * מחוייב לצורך CallBack

    למשל אם יש משתנה בשם x שפונקציה Add מקדמת אותו ופונקציית GetNum מחזירה אותו ערכו לעולם יהיה 0.
  - PerSession: מופע חדש לכל *התקשרות* – לגולש מסוים כל עוד הוא גולש המידע נשמר.

    שימוש: צורת עבודה סטנדרטית, כאשר נדרשת שמירת מידע בנוגע להתקשרות
  - Single: מופע *יחיד*, משותף לכל ההתקשרויות, שימוש: לצורך שיתוף מידע בין התקשרויות שונות (לדוג': משחק ברשת)
- הגדרת האפשרות הרצויה
  - צורת ההגדרה, [ServiceBehavior] attribute
  - המאפיין InstanceContextMode
  - מיקום הרישום, מעל מחלקת השירות
- יצירת שירות באפשרות של PerSession
  - כברירת מחדל אפשרות זו אינה עובדת עם פרוטוקול Http
  - פתרונות אפשריים:
    - עבודה עם פרוטוקול Tcp

```
<endpoint address="net.tcp://localhost:8412/..."
```

```
          binding="netTcpBinding" …>

 <serviceMetadata httpGetEnabled="false"…>
```

- עבודה עם פרוטוקול Http

```
<endpoint address="http://localhost:8412/..."
          binding="wsHttpBinding"…>
<serviceMetadata httpGetEnabled="true"
<bindings>
   <wsHttpBinding>
      <binding>
         <reliableSession ordered="true" enabled="true"/>
      </binding>
   </wsHttpBinding>
</bindings>
```

- ניהול משתמשים באפשרות של PerCall
  - במקרים מסוימים חובה להשתמש בו
  - במקרים אלו, כאשר מעוניינים לשמור נתונים שהתקבלו מהלקוח, ניתן להגדיר מילון סטטי בשירות. (מפתח- מזהה הלקוח, ערך- המידע שנרצה לשמור)

בס"ד

CallBack

- תבניות מקובלות
  - תבנית סטנדרטית: לקוח פונה לשרת, באחת מ2 צורות:
    - מודל Response-Request (בקשה-תגובה). הלקוח פונה לשרת, והשרת עונה (הלקוח קורא לפונקציה שמחזירה ערך) זהו מודל ברירת המחדל בכל תקשורת נתונים, אם לא נציין אחרת.
    - OneWay. הלקוח פונה לשרת, ומוודא רק שפנייתו התקבלה אצל השרת. הוא לא מחכה שהפעולה תסתיים, וממילא לא ממתין לתגובה (הלקוח קורא לפונקציה שלא מחזירה ערך)

כדי לפנות בצורה הזאת, יש להגדיר פונקציה המחזירה void, ולציין במפורש שזו צורת העבודה הרצויה

```
[OperationContract (IsOneWay = true)]
```

  - תבנית CallBack, שרת חוזר ללקוח. תיאור התהליך:
    - פניה ראשונה חייבת להיות מהלקוח אל השרת. (כיוון שהשרת לא יודע מי יהיו לקוחותיו). בהמשך יכולות (ולא חייבות) להיות פניות נוספות
    - השרת מבצע את הנדרש
    - בעיתוי המתאים השרת יוזם פניה ללקוח, (או ללקוחות,) שפרטיהם שמורים אצלו.

במובן מסוים, בשלב זה הופך הלקוח להיות שרת, והשרת הופך ללקוח. עובדה זו מחייבת גם את הלקוח לחשוף את הפונקציות שהוא מציע, בדומה לממשק שמעליו רשמנו ServiceContract, ושתפקידו היה חשיפת הפונקציות שמציע השירות.

- יצירת שירות בתבנית CallBack, צד הService
  - פונקציית OneWay, שתאפשר רישום (פנייה ראשונה), של הלקוח לשירות
  - הגדרת Interface נוסף (שתפקידו חשיפת הפונקציות של הלקוח, שהשרת יכול לקרוא להן)
    - ללא הattribute [ServiceContract]
    - כל הפונקציות בInterface הזה יחזירו void ויוגדרו OneWay
    - מימוש הInterface הזה יתבצע בלקוח
  - יידוע השירות בממשק החושף את פונקציות הלקוח. (נרשם בInterface החושף את השירות)

```
[ServiceContract(CallbackContract = typeof(ICallBack))
```

  - יצירת אובייקט(ים) מסוג ממשק הלקוח במחלקת השירות, שבאמצעותו (באמצעותם) נוכל לזמן את הפונקציות שבלקוח

```
ICallBack cb=OperationContext.Current.GetCallBackChannel<ICallBack>();
```

  - תבנית CallBack אפשרית רק בעבודה PerCall
- יצירת שירות בתבנית CallBack, הHost
  - הbinding חייב לתמוך בתקשורת דו-סיטרית, נשתמש בWsDualHttpBinding
- יצירת שירות בתבנית CallBack, צד הClient
  - נגדיר את המחלקה שמחזיקה מופע של השירות (טופס/ חלון וכו'), כמממשת את הממשק ICallBack, וכמובן נממש את הפונקציות הכלולות בו. (הלקוח יכיר את הממשק לאחר שביצע Add Service Reference)
  - הפונקציות האלה הן אלה שיופעלו ע"י הService מרחוק
  - נציב לתוך מופע של מחלקת System.ServiceModel.InstanceContext this- ייחוס למופע של המחלקה הנוכחית (הclient) שרץ

```
InstanceContext ic = new InstanceContext(this);
```

  - נגדיר אובייקט מסוג השירות ונשלח לc-tor שלו את המופע הנ"ל