PROJECT

1. New project > Web > ASP.NET Web Application
2. Clean project (delete: Account, App_Data, Styles, About.aspx, Default.aspx, Site.master)

DATAMODELL

3. Add > New Folder (Data)
4.
    a. Data: Add > New Item > ADO.NET Entity Data Model (bookstore.edmx)
    b. Empty Model
5. Create DATA MODEL
    a. Book:
        - **Id** – default
        - **Title** - default
        - **Price** - Type: decimal
    b. Author:
        - **Id, FirstName, LastName** - default
    c. User:
        - **Id, Username** - default
    d. Associations:
        - **Author -> Book**: End1 Multiplicity: *
        - **User -> Book**: End1 Multiplicity: *
    e. rename associations: Book.Author -> Book.Authors, Book.User -> Book.Users, Author.Book -> Author.Books, User.Book -> User.Books

6. Generate Database From Model
    a. indítani managerből a db-t és létrehozni a sémát
    b. New Connection
    c. "yes, include sensitive data..."
    d. megnézni a generált scriptet
7. Execute SQL
    a. megnézni a fizikai DB-t (entitySets, kapcsolótáblák)
8. Init data (execute "init.sql"): **http://bit.ly/szolgint-dotnet**

Classic .NET WS

9. Add > New Folder (Services)
10. Add New Item > Web > Web Service (Bookstore.asmx)
    a. megnézni a Hello World példát
11. ObjectContext, ObjectSetek + konstruktorok

```
private ObjectContext objectContext;
private ObjectSet<Book> bookSet;
private ObjectSet<Author> authorSet;
private ObjectSet<User> userSet;

public Bookstore() : this(new bookstoreContainer()) { }

public Bookstore(ObjectContext objectContext)
        {
            this.objectContext = objectContext;
            this.bookSet = objectContext.CreateObjectSet<Book>();
            this.authorSet = objectContext.CreateObjectSet<Author>();
            this.userSet = objectContext.CreateObjectSet<User>();
        }
```

12. Szolgáltatás1: GetBooks()
13. Szolgáltatás2: AddBook(...)
14. Szolgáltatás3: BuyBook(...)

```
[WebMethod]
        public List<Book> GetBooks()
        {
            return bookSet.ToList<Book>();
        }

        [WebMethod]
        public void AddBook(string title, double price, int authorId)
        {
            Book book = new Book();
            book.Price = price;
            book.Title = title;

            Author author = authorSet.Where(a => a.Id == authorId).FirstOrDefault();

            book.Authors.Add(author);

            objectContext.SaveChanges();
        }

        [WebMethod]
        public void BuyBook(int bookId, int userId)
        {
            Book book = bookSet.Where(b => b.Id == bookId).FirstOrDefault();
            User user = userSet.Where(u => u.Id == userId).FirstOrDefault();
            user.Books.Add(book);
            objectContext.SaveChanges();
        }
```

15. New Folder (DTO)
16. Add New Class (BookWithAuthors)
17. BookWithAuthors propertyk + default getters/setters

```
public class BookWithAuthors
    {
        public int id { get; set; }
        public string title { get; set; }
        public double? price { get; set; }
        public string author { get; set; }
    }
```

18.Szolgáltatás4: GetBooksWithAuthors()

a. v.ö.: EntityKey itt nincs, ellentétben az EF által kiajánlott entitásokkal

```
[WebMethod]
        public List<BookWithAuthors> GetBooksWithAuthors()
        {
            var q = from b in bookSet
                    from a in b.Authors
                    where b.Authors.Contains(a)
                    orderby a.LastName, a.FirstName, b.Title
                    select new BookWithAuthors()
                    {
                        id = b.Id,
                        title = b.Title,
                        price = b.Price,
                        author = a.FirstName + " " + a.LastName
                    };

            return q.ToList<BookWithAuthors>();
        }
```

WCF

19.Add > New Folder (ServicesWCF)

20.Add New Item > Web > WCF Service (BookstoreWCF.svc)

a. elmondani:

- WS.NET vs WCF (contract-based)
- interface szerepe itt

21.Szolgáltatás1 az interfészben: "public Books[] GetBooks()"

a. visszatérési típus: Books[](!)

- miért? mert a wcf célja a platformfüggetlen viselkedés megvalósítása, ezzel pedig együtt jár a .NET generikusok mellőzése; ez egyfajta ajánlás

```
public interface IBookstoreWCF
    {
        [OperationContract]
        void DoWork();
        [OperationContract]
        Book[] GetBooks();
    }
```

22.implementáció

a. objectcontext, objectset, konstruktorok - ahogy az előbb

b. GetBooks() - miért IQueryable?

```
public Book[] GetBooks()
        {
            IQueryable<Book> q = from b in bookSet select b;
            return q.ToArray<Book>();
        }
```

23. megosztott adattípusok: Book
    a. nézzük meg az Entity definíciók attribútumait (DataContractAttribute)
    b. osszuk meg a Bookot (az interfészen keresztül)

```
[ServiceContract]
    [ServiceKnownType("GetKnownTypes", typeof(DataHelper))]
    public interface IBookstoreWCF
…
```

```
public static class DataHelper
    {
        public static IEnumerable<Type>
GetKnownTypes(ICustomAttributeProvider provider)
        {
            System.Collections.Generic.List<System.Type> knownTypes =
                new System.Collections.Generic.List<System.Type>();
            knownTypes.Add(typeof(Book));
            knownTypes.Add(typeof(Author));
            knownTypes.Add(typeof(User));
            knownTypes.Add(typeof(BookWithAuthors));
            return knownTypes;
        }
    }
```

24. build, run, wsdl megnéz
25. a teszteléshez: console app
26. new project > windows > Console Application (BookstoreClient)
27. jobb klikk>add service reference > discover services in solution (select ServicesWCF/ BookstoreWCF.svc), nyissuk le: látszanak a tulajdonságai, stb
    a. elmondani, hogy ez mire jó
    b. név: BSService
    c. Advancedbe belenézni (collectiontype: Array)
28. class Program -> tegyük public-ká
29. kliens példányosítása
    a. BSService.BookstoreClient client = new BSService.BookstoreClient();
30. könyvek kiírátása
    a. foreach...

```
public class Program
    {
        static void Main(string[] args)
        {
            BSService.BookstoreWCFClient client = new
                        BSService.BookstoreWCFClient();
            try
            {
                var result = client.GetBooks();
                foreach (var r in result)
                {
                    Console.WriteLine("Book \"{0}\" with ID: \"{1}\"", r.Title, r.Id);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

REST

(csak GetBooks() POST-tal, böngészőből - a kliensnek csak megnézzük a kódját)

31. New folder: ServicesREST
32. New item > WCF service (BookstoreREST)
33. jobb klikk BookstoreREST.svc > View Markup
    a. add to this code: Factory="System.ServiceModel.Activation.WebServiceHostFactory"

(WebServiceHostFactory Class: A factory that provides instances of WebServiceHost in managed hosting environments where the host instance is created dynamically in response to incoming messages.)

```
<%@ ServiceHost Language="C#" Debug="true"
Service="bookstore.ServicesREST.BookstoreREST"
CodeBehind="BookstoreREST.svc.cs"
Factory="System.ServiceModel.Activation.WebServiceHostFactory" %>
```

34. Add following reference to the service project: System.ServiceModel.Web.dll
35. interfaceben: `using System.ServiceModel.Web;`

```
[ServiceContract]
    [ServiceKnownType("GetKnownTypes", typeof(DataHelper))]
    public interface IBookstoreREST
    {
        [OperationContract(Name = "GetBooks")]
        [WebInvoke(UriTemplate = "/Books/All", Method="POST")]
        Book[] GetBooks();
…
```

36. implementáció (ObjectContext, ObjectSetek + konstruktorok + GetBooks impl)

```
public Book[] GetBooks()
{
   IQueryable<Book> q = from b in bookSet select b;
   return q.ToArray<Book>();
}
```

FINAL PROJECT LAYOUT