Inlämning G

Detta dokument är ett komplement till videofilmerna, inte en ersättning.

Följande tabeller visar exempeldata för kunder, fordon och bokningar som du kan använda i lösningen om du vill slippa hitta på eget data. För att uppnå G behövs ingen inmatning i gränssnittet. Applikationen behöver endast visa data som hämtas genom **Business** och **Data** projekten och skickas till gränssnittet där datat visas.

Customers

SSN	Last Name	First Name
12345	Doe	John
98765	Doe	Jane

Vehicles

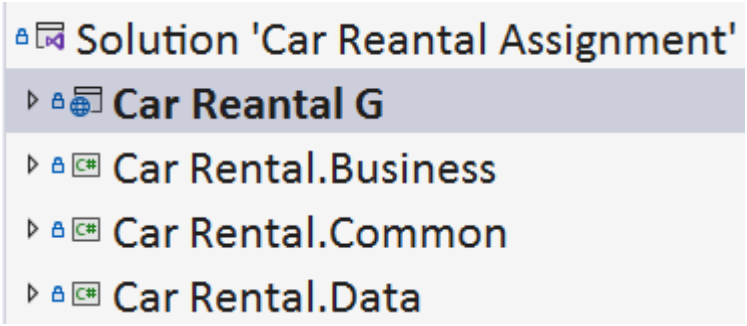
RegNo	Make	Odometer	Cost Km	Vehicle Type	$ Day	Status
ABC123	Volvo	10000	1	Combi	200	Available
DEF456	Saab	20000	1	Sedan	100	Available
GHI789	Testla	1000	3	Sedan	100	Booked
JKL012	Jeep	5000	1.5	Van	300	Available
MNO234	Yamaha	30000	0.5	Motorcycle	50	Available

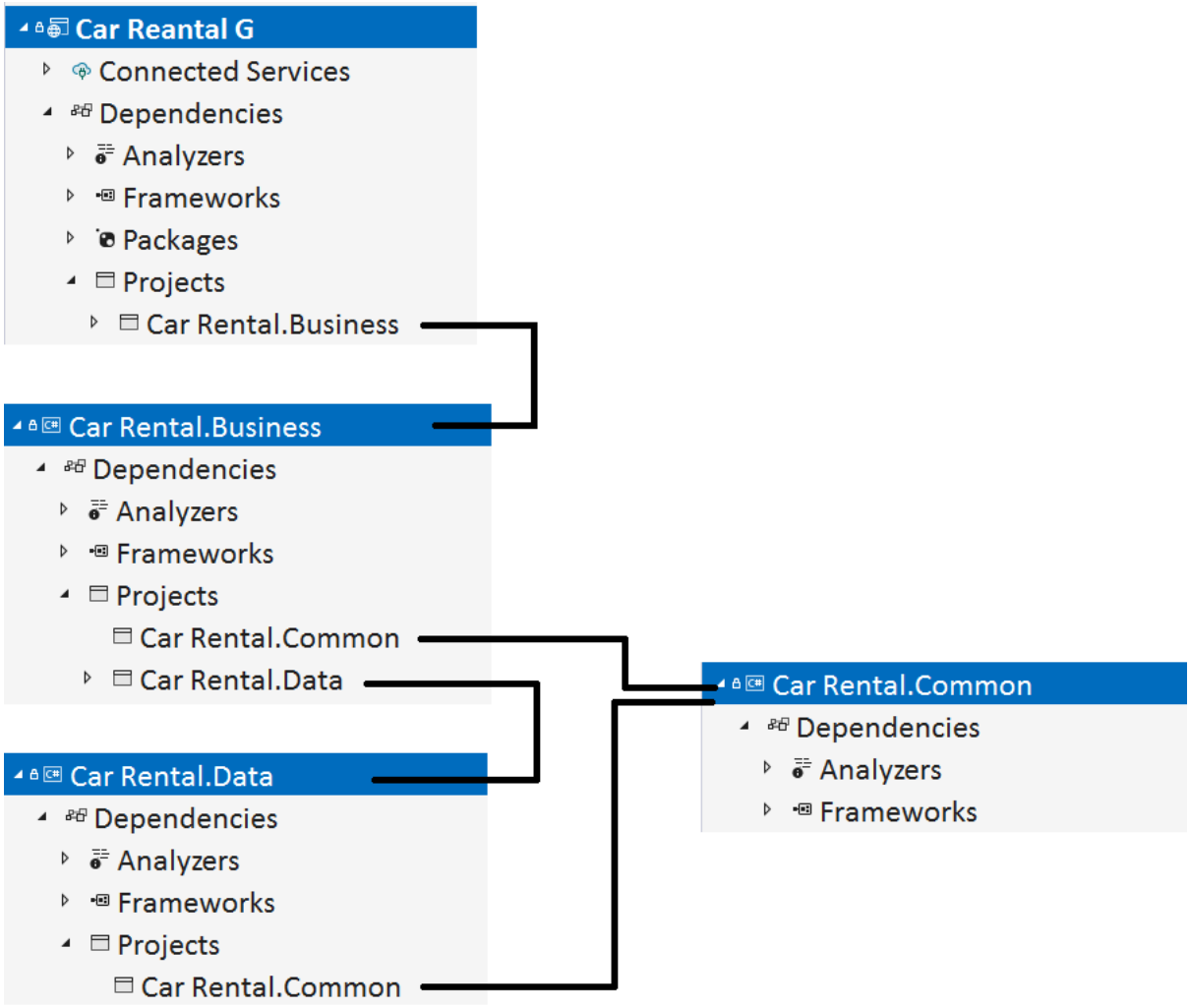
Bookings

RegNo	Customer	Km Reneted	Km Returned	Reneted	Returned	Cost	Status
GHI789	Doe John (12345)	1000		9/20/2023			Open
JKL012	Doe Jane (98765)	5000	5000	9/20/2023	9/20/2023	300	Closed

Projketstruktur

Du ska bygga en treskiktad lösning med följande projektstruktur där gränssnittet är en **WebAssembly** applikation och de resterande projekten är **Klassbibliotek**. Observera att **WebAssembly** projektet får tillgång till de andra projektens klasser genom **Business** projektets dependency.





WebAssembly Applikationen

Du kan visa tabellerna i en eller flera Razor sidor. Välj själv. **BookingProcessor** klassen måste injiceras i alla Razor sidor som visar data. Du använder det objektet (tjänsten) för att komma åt logiken som hämtar data från **Data** projektet. Inget **@code** block med C# får finnas i Razor sidorna. All C# kod ska anropas via den injicerade **BookingProcessor** tjänstens metoder.

För att kunna injicera **BookingProcessor** klassen så måste den och **CollectionData** klassen, som den är beroende av, registreras som tjänster (services) i *Program.cs* filen med **AddSingleton** metoden (likt koden för **HttpClient** från exempel koden som kommer med **WebAssembly** projektet när det skapas.)

Använd HTML/Bootstrap/Razor till att visa datat.

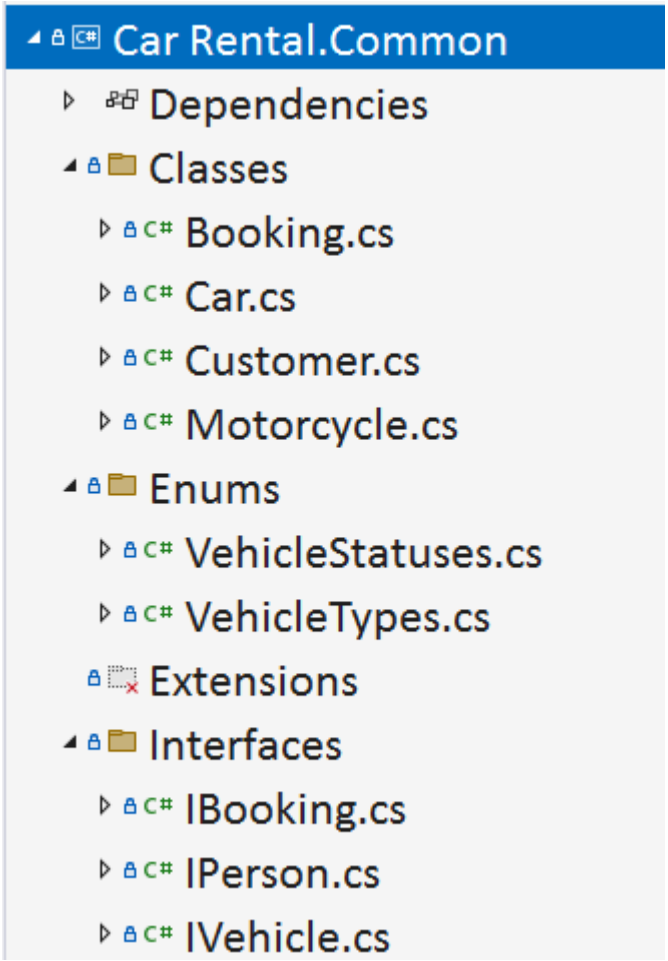
Common Projektet

Detta projekt innehåller alla klasser, enums, och interface som delas mellan projekten. **Customer** klassen implementerar **IPerson** interfacet. **Car** och **Motorcycle** klasserna implementerar **IVehicle** interfacet eftersom de ska innehålla samma properties och metoder. **Booking** klassen implementerar **IBooking** interfacet.

Booking klassen ska innehålla en metod som anropas då ett fordon lämnas tillbaks. Den ska beräkna km och dag kostnaden för uthyrningen.

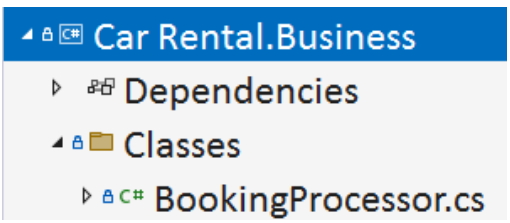
```
// Assign values to the class' properties with values from the vehicle parameter
// Cost = days * vehicle.CostDay + km * vehicle.CostKm;
```

```
void ReturnVehicle(IVehicle vehicle) {}
```



BookingProcessor Klassen (Tjänsten)

BookingProcessor klassen injiceras som en tjänst till alla Razor sidor där du visar data.



```
public class BookingProcessor
{
    private readonly IData _db;

    0 references
    public BookingProcessor(IData db) ⇒ _db = db;

    1 reference
    public IEnumerable<Customer> GetCustomers()...
    1 reference
    public IEnumerable<IVehicle> GetVehicles(VehicleStatuses status = default)...
    1 reference
    public IEnumerable<IBooking> GetBookings()...
}
```

BookingProcessor klassens metoder anropar metoder i **CollectionData** klassen i **Data** projektet för att hämta data som skickas vidare till gränssnittet i **WebAssembly** projektet.

CollectionData Klassen (Tjänsten)

Denna klass (som lägs till som en tjänst i **WebAssembly** projetets *Program.cs* fil) sköter allt data som lagras i collections eftersom projektet inte har tillgång till en databas via Entity Framework.

Klassen ska innehålla metoder som returnerar alla personer, bokningar, och fordon (antingen bokade, lediga, eller alla.)

```
public interface IData
{
    IEnumerable<IPerson> GetPersons();
    IEnumerable<IVehicle> GetVehicles(VehicleStatuses status = default);
    IEnumerable<IBooking> GetBookings();
}
```

Inlämning VG

För att uppnå VG ska **Data** projektet ha generiska metoder. **Car** och **Motorcycle** ska ärva en **Vehicle** klass istället för att implementera **IVehicle** interfacet i G uppgiften. Och flera hämta metoder implementeras så väl som metoder för att lägga till kunder, fordon, och bokningar.

Customers

SSN	Last Name	First Name	
SSN	Last Name	First Name	Add
12345	Doe	John	
98765	Doe	Jane	

Bookings

RegNo	Customer	Km Reneted	Km Returned	Reneted	Returned	Cost	Status
GHI789	Doe John (12345)	1000		9/21/2023			Open
JKL012	Doe Jane (98765)	5000	5000	9/21/2023	9/21/2023	300	Closed

Vehicles

RegNo	Make	Odometer	Cost Km	Vehicle Type	$ Day	Action	Status
RegNo	Make	0	0	Sedan		Add	
ABC123	Volvo	10000	1	Combi	200	Rent	Available
DEF456	Saab	20000	1	Sedan	100	Rent	Available
GHI789	Testla	1000	3	Sedan	100	Distance Return	Booked
JKL012	Jeep	5000	1.5	Van	300	Rent	Available
MNO234	Yamaha	30000	0.5	Motorcycle	50	Rent	Available

När en bil hyrs ut ska den asynkrona **RentVehicle** metoden fejka ett anrop till ett API och knappar och drop-downs ska inaktiveras (se bild nedan) så att man inte kan hyra, lämna tillbaka och skapa fordon. En *Processing* badge visas medan den asynkrona metoden körs och knappar och fält är inaktiverade.

Vehicles

RegNo	Make	Odometer	Cost Km	Vehicle Type	$ Day	Action	Status
RegNo	Make	0	0	Sedan		Add	Processing
ABC123	Volvo	10000	1	Combi	200	Doe John (12345) Rent	Available
DEF456	Saab	20000	1	Sedan	100	Doe John (12345) Rent	Available
GHI789	Testla	1123	3	Sedan	100	Doe John (12345) Rent	Available
JKL012	Jeep	5000	1.5	Van	300	Doe John (12345) Rent	Available
MNO234	Yamaha	30000	0.5	Motorcycle	50	Doe John (12345) Rent	Available

CollectionData Klassen

Du ersätter de icke-generiska metoderna med generiska metoder i **CollectionData** klassen som hämtar och returnerar data . De generiska metoder som returnerar data ska implementera en **Expression** function som tillåter filtrering med Lambda uttryck. Du kan använda reflection för att implementera den gnersiaka koden i metoderna.

Klassen ska även implementera metoder för att hyra och lämna tillbaka fordon. **IData** interfacet ska ha default interface metoder som returnerar enum konstanterna som arrayer till drop-down kontrollerna i Razor sidan(orna) som används vid uthyrning av fordon. Använd **Enum** klassens **GetNames** metod för att hämta konstnternas namn.

```
public interface IData
{
    List<T> Get<T>(Expression<Func<T, bool>>? expression);
    T? Single<T>(Expression<Func<T, bool>>? expression);
    public void Add<T>(T item);

    int NextVehicleId { get; }
    int NextPersonId { get; }
    int NextBookingId { get; }
```

```
    IBooking RentVehicle(int vehicleId, int customerId);
    IBooking ReturnVehicle(int vehicleId);

    // Default Interface Methods
    public string[] VehicleStatusNames => Retunera enum konstanterna
    public string[] VehicleTypeNames => Retunera enum konstanterna
    public VehicleTypes GetVehicleType(string name) => Retunera en enum konstants
värde med hjälp av konstantens namn
}
```

BookingProcessor Klassen

RentVehicle metoden ska vara asynkron. Använd **Task.Delay** för att simulera tiden det tar att hämta data från ett API.

```
public class BookingProcessor
{
    private readonly IData _db;

    public BookingProcessor(IData db) => _db = db;

    public IEnumerable<IBooking> GetBookings() { }

    public IEnumerable<Customer> GetCustomers() { }

    public IPerson? GetPerson(string ssn) { }

    public IEnumerable<IVehicle> GetVehicles(VehicleStatuses status = default){ }

    public IVehicle? GetVehicle(int vehicleId) { }

    public IVehicle? GetVehicle(string regNo) { }

    public lägg till asynkron returdata typ RentVehicle(int vehicleId, int
customerId)
    {
            // Använd Task.Delay för att simulera tiden det tar
            // att hämta data från ett API.
    }

    public IBooking ReturnVehicle(int vehicleId, double ditance) { }

    public void AddVehicle(string make, string registrationNumber, double
odometer, double costKm, VehicleStatuses status, VehicleTypes type) { }

    public void AddCustomer(string socialSecurityNumber, string firstName, string
lastName) { }

    // Calling Default Interface Methods
    public string[] VehicleStatusNames => _db.VehicleStatusNames;
    public string[] VehicleTypeNames => _db.VehicleTypeNames;
    public VehicleTypes GetVehicleType(string name) => _db.GetVehicleType(name);

}
```

Lycka Till!