Del 1 – C# & .NET

Oppgave 1 – Hello World

Den første oppgaven er temmelig enkel:

Lag en 'Hello World' applikasjon.

Oppgave 2 – Personer

Vår applikasjon vil omhandle personer. Disse personene har egenskaper som navn, alder, hårfarge, høyde og kjønn. Disse egenskapene skal registreres. En persons alder kan ikke være mindre enn 0.

Lag en klasse med properties som reflekterer dette. Husk at klassen må overskrive ToString metoden.

Oppgave 3 – Log

Kunden vår ønsker å ha mulighet til å logge hendelser.

Lag en metode på person-klassen som logger utsagn til konsollet (Console.WriteLine()). Metoden skal logge ett eller flere utsagn om gangen. Hvert utsagn skal prefikses med personens navn. (Bonuspoeng til de som klarer å legge til et default utsagn: "Blah blah blah..")

Eksempellogg:

Starter loggen…

KongenDin: "min karse vokser fortere enn Sonja sin!! YESSSS!!!!!"

KongenDin: "spilte vri åtter og vridde akkurat med en knekt. Han der lille marius er ikke så veldig observant."

Oppgave 4 – Enums

Kunden vår holder oppsikt med et stort antall personer. Det er ikke alltid alle oppfører seg like fint, og kunden vil ha mulighet til å registrere hvorvidt en person oppfører seg bra eller dårlig. Kunden har dessverre ikke full oversikt, så systemet må også ha mulighet for å registrere at oppførselen er ukjent. Begynn derfor denne oppgaven med å lage en enum som gjenspeiler dette.

Opprett en property på personobjektet som kan lagre en persons oppførsel.

Oppgave 5 – Liste av personer

Med vår nye kunnskap ser vi at deler av systemet kan forenkles. Benytt object initializers til å opprette en array med personer. Se forslag til testpersoner under:

Navn	Alder	Hårfarge	Høyde	Tilbøyelighet	Kjønn
Patrick Bateman	27	Brun	184cm	Ond	M
Mystique	127	Rød	177cm	Ond	K
Two Face	58	Brun	183cm	Ond	M
Cruella De Vil	65	Svart og Hvitt	168cm	Ond	K
Orochimaru	100	Svart	180cm	Ond	M
Harvey Dent	56	Brun	183cm	God	M
KongenDin	77	Ukjent	150cm	God	M

Kunden ønsker å få skrevet ut en liste over alle personene som er registrert i systemet.

Iterer igjennom personene i registeret og kall ToString metoden på hvert personobjekt.

Oppgave 6 – Filtrering

Nå som dere har bygget litt kompetanse ønsker kunden å få se noen reelle resultater.

Personer som har svart hår eller er over 100 år gamle er klart mistenkelige. Kunden vil gjerne ha en utskrift over disse personene så snart som mulig.

Bruk LINQ til å hente ut en liste over disse personene.

Oppgave 7 – Group by

Kunden har nok en god dag og er strålende fornøyd. Nå som han har fått gjennomslag for såpass mye ny funksjonalitet så ønsker han å fortsette å utvikle systemet.

Kunden ønsker seg en oversikt over personene som er registrert i systemet fordelt etter tilbøyelighet.

Ta utgangspunkt i lista fra forrige oppgave og grupper på tilbøyelighet. Skriv ut grupperingene.

Eksempel:

Eeevil:

Hans

Gretchen

Good:

Petter Sprett

Kristoffer Robin

Dontknow:

Supermann

Oppgave 8 – Extension methods

Det viser seg at systemet inneholder en del feilregistrerte telefonnummere. I ekstreme tilfeller går systemet ned med en uhåndtert NullReferenceException.

Lag en extension-metode for string-klassen som kan validere telefonnummere. Gi metoden navnet IsValidPhoneNumber. Det holder å sjekke at nummeret er satt, og at det er minst 8 tegn langt.

Oppgave 9 – Lambda-uttrykk

Array-klassen har en generisk, statisk metode som heter Find. Metoden tar inn en array og ett predikat.

Lag et lambda-uttrykk som returnerer true hvis personen som gis inn er over 80 år.

Benytt Find-metoden sammen med lambdauttrykket for å finne en person i lista som er over 80.

Hint: Et predikat er en delegat som tar inn ett objekt og returnerer en boolsk verdi.

Oppgave 10 – Spørreoperatorer

Oppgave 10.1 Where

En lokal kiosk har blitt ranet, og ransmannen kom seg unna i en stor rød folkevogn. Opptakene fra kioskens overvåkningskamera viser at raneren var over 1.80 meter høy og vitner hevder han hadde brunt hår. Benytt Where-operatoren til å hente ut en liste over mulige forbrytere.

Oppgave 10.2

Benytt Select-operatoren til å hente ut en array med alle navnene til alle i registeret.

Del 2 – ASP.NET MVC

Oppgave 1 – To sider som lenker til hverandre

Kunden er fornøyd med registeret og ønsker at flere skal kunne bruke det via web. Opprett en ny ASP.NET MVC 3 applikasjon (Empty) og legg til to Controllere; Home og Persons. Legg til en Index-action på hver av Controllerne, som returnerer et View.

Oppgave 2 – Utlisting av personer

Kopier Person-klassen fra tidligere oppgaver til Models-mappa i ASP.NET MVC prosjektet. Gjør Index Viewet til Persons Controlleren sterkt typet til en liste av Personer. Gjenbruk koden fra Del 1, Oppgave 5 og send inn disse personene til Viewet. Skriv ut listen av personer i en tabell.

For enkelhetsskyld dropper vi å registrere om en person er ond/god, så du kan fjerne Enum + Property for dette.

Oppgave 3 – Registrere nye personer

Vi har behov for å kunne registrere nye personer i systemet. Lag et nytt Insert View, og en ny Insert Action på Person Controlleren. Viewet burde være sterkt typet til Person-typen. Du må og lage en Insert-action som tar inn en Person, og dekorere den med HttpPost attributet, for å kunne sende en person til serveren.

Legg til valideringsregler som sørger for at en person må være mellom 0 og 150 år, 0 og 250cm, at navn og kjønn er påkrevd, og at kjønn kun kan være verdiene M eller K (hint: ^[MK]{1}$).

Sett et break-point i Insert-actionen for å sjekke at du får inn et gyldig Person-objekt. Send brukeren tilbake til listen over personer dersom objektet er gyldig.

Oppgave 4 – Registrere personer i database

For at systemet skal bli virkelig nyttig må vi kunne registrere nye personer i en database. Legg inn SqlCompact og Entity Framework ved hjelp av NuGet:

- `Install-Package EntityFramework.SqlServerCompact`

Opprett en «App_Data» mappe. Dette gjør du ved å høyre klikke på prosjektet og velge «Add – ASP.NET Folder – App Data».

Legg til ny Connection String i Web.Config (nederst, før </Configuration>):

```
<connectionStrings>
  <add name="PersonWeb.Model.PersonContext"
       connectionString="Data Source=|DataDirectory|PersonWeb.sdf"
       providerName="System.Data.SqlServerCe.4.0" />
</connectionStrings>
```

Du må og lage en DbContext klasse i Model-mappen.

```
public class PersonContext : DbContext
{
    public DbSet<Person> Persons { get; set; }

    static PersonContext()
    {
        Database.SetInitializer(new CreateDatabaseIfNotExists<PersonContext>());
    }
}
```

Nå er du klar til å begynne å bruke databasen. Oppdater Index Actionen på Persons Controlleren til å lese alle personene fra databasen ved hjelp av LINQ.

Legg så til en ny Action, Insert og lenke til denne fra Index Viewet. Bruk innebygget scaffolding til å lage et nytt Insert View. Legg til en Insert action som tar en Person model, og som er satt til HTTP POST. Lagre denne i databasen ved hjelp av LINQ, og send brukeren tilbake til oversikten.

Oppgave 5 – Oppdatere en eksisterende person

Kunden har behov for å kunne oppdatere registrerte personer. Lag en ny Update Action med View. Lenke til denne fra tabellen med personer, f.eks. ved å legge inn en ny kolonne som heter «Rediger» på slutten av hver rad. Husk å sende inn ID til personen som skal redigeres.

```
<td>@Html.ActionLink("Rediger", "Update", new {Id = person.Id})</td>
```

Her må du og ha to Action som heter Update, en som tar inn en Id og henter personen som skal redigeres fra databasen og sender han til Viewet, og en som tar inn den oppdaterte Personen i en HTTP POST, og lagrer endringene i databasen.

Send brukeren tilbake til oversikten når personen er oppdatert.

Oppgave 6 – Bonusoppgave: Slette en eksisterende person

Om du har tilgjengelig tid, se om du får til å legge inn funksjonalitet for å slette en person.

Oppgave 7 – Bonusoppgave: Returnere en liste av personer som JSON

For å kunne bruke systemet fra flest mulig enheter ønsker kunden ett enkelt web api. Lag en ny Action som returnerer en oversikt over alle kundene som JSON.