# Database 06

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf4:** anvende centrale metoder og teknikker til at realisere modeller i et databasesystem […] |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan nævne flere kendetegn ved **database design** * [Unistrukturel] Du kan beskrive flere kendetegn ved en tabel (**database implementering**) |
| **Din forberedelse** | Programmeringssporet:   * Læs om [refaktorering vs redesign](https://www.javacodegeeks.com/2017/11/refactoring-vs-redesign.html) * Læs om [SQL-injection](https://www.w3schools.com/sql/sql_injection.asp)   + Med fokus på ’Use SQL Parameters for Protection’ i ASP.NET   + Husk tidligere forberedelse med forskellen på [SqlCommand.Parameters](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlcommand.parameters) (Add vs AddWithValue) * [SQL injection](https://www.linkedin.com/learning/programming-foundations-databases-2/access-control-compliance-and-injection) [video: 1:20 - 2:27] * Læs om [Desugaring the using statement](https://wouterdekort.com/2013/09/30/desugaring-the-using-statement/)   + Se bort fra intermediate language (IL)-koden i artiklen |

Som supplement til din forberedelse (læs: valgfrit), så se om du kan besvare følgende spørgsmål efter du har læst/set forberedelsen (*og lidt fra tidligere):*

* Bliver der taget højde for exceptions internt i vores using-statement?
* Hvad kan man gøre for at undgå et SQL injektionsangreb?
* Hvordan kommenterer man kode ud i en SQL forespørgsel (en: query)?
* Skal ens Repository altid indeholde en Add, GetAll, Update og Remove metode?
  + Er det tilladt at have et repository med flere metoder end ovennævnte? (Begrund svar)
* Hvor mange typer af data må man operere på i et repository?
* Hvis man omdøber en metode i ens program, har man så benyttet sig af redesign eller refaktorering?
* Hvor ofte bør man benytte sig af refaktorering?
* Hvad sker der, hvis man benytter et class library som ens primære opstartsprojekt (en: startup projekt)?
* Er ens database sårbar overfor forekomsten af anomalier, hvis man har implementeret ens referentielle integritetsrestriktioner i ens tabeller (primærnøgle, fremmednøgle)? (Begrund svar)
* Hvad benytter man *@@Identity* til?

# Webcomic of the day



# Øvelse 1: Terminologi

Del teamet op i to mindre grupper (3-4 pers. i hver), og brug **Ordet rundt** til at reflektere over begreberne ”Repository”, ”SQL Injection”, ”using statement”, ”Refaktorering” og ”Redesign”. Sørg for, at alle får mulighed for at tale.

*Tidsramme: ca. 20 minutter*

# Øvelse 2: SQL Injektion

Du har som en del af forberedelsen til i dag læst omkring SQL injektioner. Hvis du som programmør ikke har taget højde for denne problematik i dine løsninger, så er du sårbar overfor, at en ondsindet aktør kan udføre tilfældige kommandoer imod din database. Du vil i disse deløvelser blive holdt i hånden til at udføre et aktuelt SQL-angreb. Det skal siges, at det **ikke er tilladt** at benytte denne viden til at angribe andre hjemmesider på internettet end den side, som I vil blive præsenteret for i denne øvelse (du vil blive retsforfulgt jvf. [Straffeloven §263](https://danskelove.dk/straffeloven/263)). Du vil du se på, hvordan du som programmør i en .NET verden, kan undgå denne problematik.

## 2.1: Udfør et SQL injektionsangreb

Du og din sidekammerat skal i denne øvelse se nærmere på, hvordan et konkret SQL injektionsangreb ser ud. Tag jer god tid til at forstå, hvad der sker – trin for trin.

Fremgangsmåde:

* Gå ind på <https://www.hacksplaining.com/exercises/sql-injection>

## 2.2: Hvordan undgår vi dette problem?

Efter ovenstående øvelse er udført, bliver du ført videre til en [side](https://www.hacksplaining.com/prevention/sql-injection) med gode råd om, hvordan man kan undgå SQL injektioner i forskellige programmeringssprog. Forhold dig kursorisk (læs: skim) til rådene på siden, men se om du kan finde noget på listen, som du har kendskab til. Vend det, som du har fundet frem til med dit team.

Bonusspørgsmål**:** Beskriv, hvad forskellen er på at parameterisere ens input og sanitere det

# Øvelse 3: PetParadise konsol-applikation

**Benyt parprogrammering til alle resterende øvelser.**

Du skal fortsætte med PetParadise-casen og have primær fokus på at implementere Repositories og database-forespørgsler til PetParadise-databasen via C#.

Sagen er nu den, at klinikken PetParadise havde henvendt sig til det lille udviklerhus FishyCode, som prøvede at skrive en applikation til PetParadise, men havde svært ved at komme i mål med det. PetParadise måtte erfare, at ’Fishy’ desværre ikke havde så meget med dyr at gøre. Derfor skal du sørge for, at deres kode bliver videreimplementeret. Der er allerede udviklet en del artefakter, som du så skal tage udgangspunkt i:

* **’Ex44-CreateAndInsertPetParadise.sql’**, tekstfil med alle de SQL-sætninger, der skal til for at oprette tabellerne OWNER, PET og TREATMENT samt indsætte data i dem.
* **’Ex44-PetParadise.zip’**, en zippet .NET konsolapplikation, som du skal udvikle videre på.
* **’Ex44-PetParadise-DCD.pdf’**, en DCD for PetParadise-applikationen
* **’Ex44-PetParadise-UnitTest.txt’**, en unit-test til at teste applikationen med.

Disse udleverede artefakter kan hentes fra materiale-repositoriet på GitLab.

## Øvelse 3.1: Indledende forberedelse

Udfør følgende:

* I din MS SQL database, opret de 3 tabeller ved at køre SQL-sætningerne i den udleverede sql-tekstfil ’Ex44-CreateAndInsertPetParadise.sql’. Check, at alle tabeller er oprettet korrekt, og der er data i dem. *Selvom du allerede har set på opbygningen af disse tabeller tidligere, er det vigtigt, at du anvender den udleverede tekstfil til at oprette nye tabeller + data.*
* Udpak zip-filen til konsolapplikationen PetParadise. Åbn løsningen, og check om du kan kompilere den (build). Orienter dig i løsningen og se hvorledes den stemmer overens med den udleverede DCD.
* Opret et test-projekt i PetParadise-løsningen: **vær opmærksom på at vælge projektskabelonen ’MSTest Test Project’ til C#** til unittest i .NET projekter.
  + Indsæt teksten fra ’Ex44-PetParadise-UnitTest.txt’ i unit-testen (husk reference & using)
  + Afprøv testen; den skulle gerne **fejle** på alle tests til at starte med

Orientér dig først i de udleverede artefakter, så du forstår sammenhængen mellem dem. Tal med din makker om de udleverede artefakter, så I begge har en god forståelse af dem, inden I starter.

I PetParadise-løsningen er følgende dele færdigimplementeret:

* Unit-testen (når I har oprettet testprojektet og indsat de udleverede testmetoder)
* Controller-klassen

Derudover er følgende klasser delvist implementerede:

* OwnerRepo
* PetRepo
* TreatmentRepo
* DatabaseHelper
* Domæneklasserne Owner, Pet og Treatment, hvor kun ToString()-metoden mangler i hver af dem

Det er disse klasser, som du i det følgende skal færdigimplementere, så alle tests kan gennemføres.

Når du skal skabe en forbindelse til databasen, skal du anvende en connection-string på formen:

* ”Server=10.56.8.36; database=<databasenavn>; user id=<user>; password=<password>;”

Hvor du skal erstatte <database> med det databasenavn, du anvender til øvelsen, <user> med dit brugernavn og <password> med dit password. Så skulle du have en fungerende connection-string.

**Bonusøvelse:** Opret en App.config-fil og anvend den viden du fik i ”Ex41-Kom godt i gang med Sql i et C# Visual Studio Projekt.docx”

## Øvelse 3.2: Implementér ToString()-metoden i hver domæneklasse

Du skal først implementere ToString()-metoder i hver domæneklasse. Se i testprojektet, hvad de enkelte ToString()-implementeringer skal returnere.

## Øvelse 3.3: Implementér OwnerRepo-klassen

Du skal implementere OwnerRepo-klassen, så alle tests kan gennemføres.

Fokusér i første omgang kun på at implementere:

* OwnerRepo-constructor’en
* GetAll()
* GetById()

## Øvelse 3.4: Implementér PetRepo-klassen

Implementér PetRepo-klassen, så alle tests kan gennemføres.

Fokusér i første omgang kun på at implementere:

* PetRepo-constructor’en
* GetAll()
* GetById()

## Øvelse 3.5: Implementér TreatmentRepo-klassen

Implementér TreatmentRepo-klassen, så alle tests kan gennemføres.

Fokusér i første omgang kun på at implementere:

* TreatmentRepo-constructor’en
* GetAll()
* GetById()

## Øvelse 3.6: Implementér DatabaseHelper

Implementér DatabaseHelper-klassen, så alle tests kan gennemføres.

DatabaseHelper-klassen skal kontakte SQL databasen direkte for at udføre de ønskede forespørgsler.

# Øvelse 4: CRUD-operationer

Fortsæt nu med PetParadise-løsningen, hvor du skal implementere de resterende metoder Add(), Update() og Remove() i hver af de tre repositories.

Udvid tilsvarende Controller-klassen med metoder (f.eks. CreateOwner (…), UpdateOwner(…), DeleteOwner(…), osv.) som kalder Add()-, Update()- og Remove()-metoderne for hver af de tre repositories.

Skriv mindst én testmetode som via Controller’en tester hver af de 3 metoder for hver af de tre repositories.