

Tisdag 27 Mars 2018



Databasteknik & PHP

70 YHP

# Idag kan vi det mesta 😊

- PHP
- GET vs POST
- Klasser
- Databaser
- SQL
- ER DIAGRAM

# Nu ska vi gå in på det sista

- **Normalisering**
- **Objektorienterad programmering med databaser**
- **Imorgon: Säkerhet**

# Normalisering

# Normalisering

- När vi ritat diagram så har vi aktivt ignorerat ifall databasen går att bygga eller inte
- Istället har vi fokuserat på att göra en modell över "verkligheten"

# Normalisering

- När vi ritat diagram så har vi aktivt ignorerat ifall databasen går att bygga eller inte
- Istället har vi fokuserat på att göra en modell över "verkligheten"
- Detta eftersom det är viktigt att modellera verkligheten rätt

# Normalisering

- För att sedan skapa ett diagram som går att bygga en databas kring går diagrammet igenom en process som kallas "normalisering"
- Man gör stegvis förändringar där modellen innehåller samma information, men är annorlunda strukturerad

# Normalisering

- Normalisering har formellt fem stycken steg:
  - **UNF:** Full förankring i verkligheten
  - Vidare, 1NF, 2NF, 3NF och 4NF
  - Där 4NF ska vara ett diagram som går att skapa med SQL DDL

# Normalisering

- Varje steg görs **en** process, exempelvis plocka bort alla många till många förhållanden eller bryta ut alla dubbelvärden (namn -> för & efternamn)
- **Inför tentamen:**  
Ni behöver inte kunna normalisera diagram, men ni ska veta vad det är och vad man har det till 🧐

# **Objektorienterad programmering med databaser**

# Objektorienterad programmering med databaser

- ER Diagram kan som sagt användas för att skapa diagram för både databaser, men även för klasser
- Det ger oss en hint om att klasser och databastabeller är lätta att koppla ihop
- Varför vill man göra detta?

# Objektorienterad programmering med databaser

- ER Diagram kan som sagt användas för att skapa diagram för både databaser, men även för klasser.
- Både i teorin om relationsdatabaser och om objektorienterad programmering används termen "entitet"
- Det ger oss en hint om att klasser och databastabeller är lätta att koppla ihop
- Men varför vill man göra detta?

# Objektorienterad programmering med databaser

- En klass kan kopplas till en tabell eller vy i databasen, detta kallas för "Entity relationship mapping"
- Föreställ er denna kod:

```
const kangaroo = new Animal("1");
```

```
kangaroo.jump();  
kangaroo.save();
```

# Objektorienterad programmering med databaser

- En klass kan kopplas till en tabell eller vy i databasen, detta kallas för "Entity relationship mapping"
- Föreställ er denna kod:

```
const kangaroo = new Animal("1");
```

```
kangaroo.jump();  
kangaroo.save();
```

Hämtar en känguru med id 1 från databasen

# Objektorienterad programmering med databaser

- En klass kan kopplas till en tabell eller vy i databasen, detta kallas för "Entity relationship mapping"
- Föreställ er denna kod:

```
const kangaroo = new Animal("1");
```

```
kangaroo.jump();    Hoppar 1 gång  
kangaroo.save();
```

# Objektorienterad programmering med databaser

- En klass kan kopplas till en tabell eller vy i databasen, detta kallas för "Entity relationship mapping"
- Föreställ er denna kod:

```
const kangaroo = new Animal("1");
```

```
kangaroo.jump();
```

```
kangaroo.save();
```

**Sparar kängurun i databasen igen**

# Objektorienterad programmering med databaser

- Ett annat kodexempel:

```
new Order($customerId);
```

```
Order.addProduct( new Product('14') );
```

```
Order.send($shippingAdress);
```

# Objektorienterad programmering med databaser

- Ett annat kodexempel:

```
new Order($customerId);
```

```
Order.addProduct( new Product('14') );
```

```
Order.send($shippingAdress);
```

# Objektorienterad programmering med databaser

Fördelar med ERM:

- Man kan gömma databaskopplingar och metodik från resten av programmet
- Man kan *sanitera* och *kryptera* information, så som lösenord.
- Man kan ge sin databasinformation "liv" med hjälp av funktioner.

# Objektorienterad programmering med databaser

Fördelar med ERM:

- Man kan gömma databaskopplingar och metodik från resten av programmet
- Man kan *sanitera* och *kryptera* information, så som lösenord.
- Man kan ge sin databasinformation "liv" med hjälp av funktioner.

Nackdelar med ERM:

- Det är svårt att leva utan det!

# Objektorienterad programmering med databaser

Det är viktigt att vi har koll på våra olika datautrymmen, vart saker kan sparas:

- minnet
- cookies
- session
- databasen

# Objektorienterad programmering med databaser

Det är viktigt att vi har koll på våra olika datautrymmen, vart saker kan sparas:

- minnet **\$variabler**
- cookies
- session
- databasen

# Objektorienterad programmering med databaser

Det är viktigt att vi har koll på våra olika datautrymmen, vart saker kan sparas:

- minnet **\$variabler**
- cookies **setcookie**
- session
- databasen

# Objektorienterad programmering med databaser

Det är viktigt att vi har koll på våra olika datautrymmen, vart saker kan sparas:

- minnet **\$variabler**
- cookies **setcookie**
- session **\$\_SESSION**
- databasen

# Objektorienterad programmering med databaser

Det är viktigt att vi har koll på våra olika datautrymmen, vart saker kan sparas:

- minnet **\$variabler**
- cookies **setcookie**
- session **\$\_SESSION**
- databasen **mysqli**

# Objektorienterad programmering med databaser

Hur får varje klass tillgång till databasanslutningen?

- Skicka in det som en parameter:  
`new Statsminister($conn, '1')`
- Spara anslutningen i `$GLOBALS`
- Bygg en *Factory*

# Factories

En klass eller en funktion som returnerar nya objekt kallas för en Factory

# Factories

Exempel på en factory:

```
function animalFactory($name, $type) {  
  
  if ($type == "worm") { return new Worm($name) }  
  
  if ($type == "monkey") { return new Monkey($name) }  
  
  if ($type == "cat") { return new Cat($name) }  
  
}
```

# Factories

En factory kan använda den inbyggda PHP funktionen

*fetch\_object()*

Används istället för *fetch\_assoc()*

# Factories

En factory kan använda den inbyggda PHP funktionen

*fetch\_object( \$classNameAsString, \$arrayOfConstructorParameters);*

# Factories

En factory kan använda den inbyggda PHP funktionen

```
$results->fetch_object("Animal", [ $name ]);
```

# Factories

```
$results->fetch_object("Animal", [ $name ])
```

Samma resultat som:

```
$row = $results->fetch_assoc();
```

```
$obj = new Animal($name);
```

```
$obj->antalBen = $row['antalBen'];
```

```
$obj->typ = $row['typ'];
```

# Factories

```
$results->fetch_object("Animal", [ $name ])
```

Samma resultat som:

```
$row = $results->fetch_assoc();
```

```
$obj = new Animal($name);
```

```
$obj->antalBen = $row['antalBen'];
```

```
$obj->typ = $row['typ'];
```

# Tänkvärt

*fetch\_object()* kan och bör användas även om man inte har en factory

# Tänkvärt

*Arv kan användas för gömma databasdetaljer från subklasserna, detta gör det enkelt och snyggt att implementera nya klasser.*

# Tänkvärt

*Arv kan användas för gömma databasdetaljer från subklasserna, detta gör det enkelt och snyggt att implementera nya klasser.*

# Tänkvärt

En klass som används till ERM bör innehålla metoder för

create  
update  
delete  
read

# Tänkvärt

Vissa metoder skulle kunna uppdatera databasen automatiskt, säg exempelvis, *placeOrder()*;

# Vidare läsning

- <https://phpenthusiast.com/object-oriented-php-tutorials/create-classes-and-objects>
- <http://propelorm.org>
- [https://en.wikipedia.org/wiki/Object-relational\\_mapping](https://en.wikipedia.org/wiki/Object-relational_mapping)
- <http://www.oodesign.com>

Tack!