

MVC


Model-View-Control

Av Marcus Brohede



Friheten hos PHP, En styrka och en svaghet

- Vi har full frihet i PHP
- Ordna kod i valfritt antal filer
- Blanda presentations-kod med datalager-kod och server-logik



```
<html>
<body>
<form action="PHP_Exempel_FOUPregistrator.php" method="post">
  <input type="text" name="Custno" /><br>
  <input type="text" name="Name" /><br>
  <input type="text" name="SSN" /><br>
  <input type="text" name="SSN2" /><br>
  <input type="text" name="Regdata" /><br>
  <input type="submit" />
</form>
</body>
```

VIEW

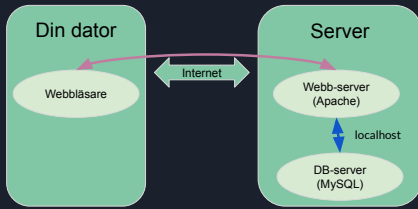
```
<?php
$pdo = new PDO('mysql:host=localhost; dbname=foop', 'myusername', 'mypassword');
$stmt = $pdo->prepare('INSERT INTO CUSTOMER (CUSTNO, SSN, NAME, REG,
// Only make insert if there is a form post to process
if(isset($_POST['Custno'])) {
    $custno = $_POST['Custno'];
    $stmt = $pdo->prepare('INSERT INTO CUSTOMER (CUSTNO, SSN, NAME, REG,
    $stmt->bindParam(':CUSTNO', $custno, PDO::PARAM_INT);
    $stmt->bindParam(':SSN', $_POST['SSN'], PDO::PARAM_INT);
    $stmt->bindParam(':NAME', $_POST['Name'], PDO::PARAM_STR);
    $stmt->bindParam(':REGDATE', $_POST['Regdata'], PDO::PARAM_STR);
    $stmt->execute();
}
// Read all customers
$stmt = $pdo->query('SELECT * FROM CUSTOMER') as $row {
    echo "<table>";
    echo "<thead>";
    echo "<tr>";
    echo "<th>Custno</th>";
    echo "<th>SSN</th>";
    echo "<th>Name</th>";
    echo "<th>Regdate</th>";
    echo "</tr>";
    echo "<tbody>";
    while($row = $stmt->fetch()) {
        echo "<tr>";
        echo "<td>";
        echo "</td>";
        echo "</tr>";
    }
    echo "</tbody>";
    echo "</table>";
}
```

MODEL

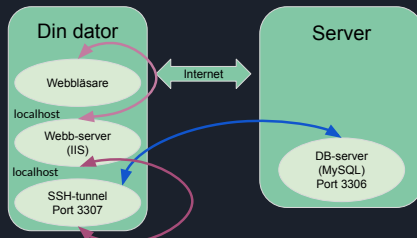
```
<table>
<thead>
<tr>
<th>Custno</th>
<th>SSN</th>
<th>Name</th>
<th>Regdate</th>
</tr>
<tbody>
<tr>
<td>1</td>
<td>123456789</td>
<td>John Doe</td>
<td>2023-01-01</td>
</tr>
</tbody>
</table>
```

VIEW

Ansluta till vår PHP-webbapplikation



Ansluta till vår .NET-webbapplikation



Skapa SSH-tunneln

Du behöver skapa en SSH-tunnel eftersom MySQL-servern endast tillåter anslutningar från localhost av säkerhetsskäl.

```
brom -- ubuntu@webbug-cloud: ~ -- ssh databaskonstruktion@wwwlab.webbug.se -L3308:127.0.0.1:3306 -N -...
brom@Marcuss-MacBook-Pro ~ % ssh databaskonstruktion@wwwlab.webbug.se -L3308:127.0.0.1:3306 -N
databaskonstruktion@wwwlab.webbug.se's password: [ ]
```

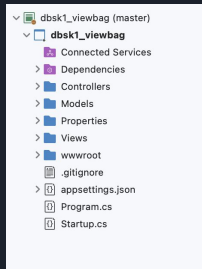
Du behöver ha igång SSH-tunneln så länge som din applikation körs.

```
brom -- ubuntu@webbug-cloud: ~ -- ssh databaskonstruktion@wwwlab.webbug.se -L3308:127.0.0.1:3306 -N -...
brom@Marcuss-MacBook-Pro ~ % ssh databaskonstruktion@wwwlab.webbug.se -L3308:127.0.0.1:3306 -N
databaskonstruktion@wwwlab.webbug.se's password: [ ]
```

Model - View - Control (MVC)

Model-View-Control är ett designmönster som tydligt separerar kod för:

- Presentation (VIEW)
- Datalagring (MODEL)
- Applikationslogik (CONTROL)

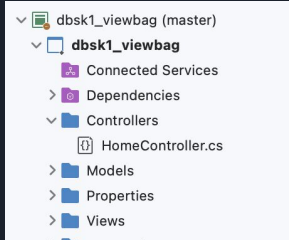


Controller

En **controller** definierar ett antal "end-points" eller **actions** vilket motsvarar de olika tjänster som vår webbapplikation skall kunna hantera. Till exempel, skapa användare, lista kunder, betala faktura, osv

I .NET måste filnamnet avslutas med ...Controller.cs

Standard-controllern heter: HomeController.cs



Deklarera End-points

```
...
public IActionResult Index()
{
    ViewBag.AllCustomersTable = sp.GetAllCustomers();
    return View();
}

public IActionResult SearchInvoiceRows(string custno)
{
    ViewBag.SearchResults = sm.SearchInvoiceRows(custno);
    return View();
}
...
```

Namnet på end-point som du surfar till

Surfa till en End-point

Visual Studio startar vår applikation på localhost och en specifik port. I mitt exempel är det 5001

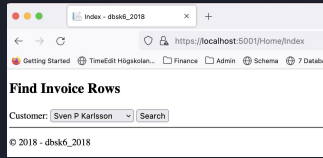
Sökvägen till en specifik end-point är:

[CONTROLLER]/[END-POINT]

I filen **Startup.cs** anges en standard end-point som man skickas till om man inte anger controller och end-point

Som standard är detta Home/Index

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute("default", "{controller=Home}/{action=Index}");
});
```



Controller - Data-inhämtning

Controllern ansvarar för att hämta rätt data från våra olika modeller och sedan anropa rätt vy för utritning av vår end-point

```
...
private CustomersModel customersModel = new CustomersModel();
...
public IActionResult Index()
{
    ViewBag.AllCustomersTable = customersModel.GetAllCustomers();
    return View();
}
...
```

Definiera models

Normalt vill man definiera en model per tabell/view i databasen.

Sedan skapar man ett antal metoder i sin modell som kan utföra de olika operationer man vill kunna utföra på sin tabell. T ex lista användare, lägga till kund, uppdatera faktura, osv

```
> Dependencies
> Controllers
▼ Models
  CustomersModel.cs
  ErrorViewModel.cs
  InvoiceRowsModel.cs
> Properties
> Views
```

```

public class CustomersModel
{
    private string connectionString =
        "Server=localhost;Port=3308;Database=a00leifo;
        User ID=sqllab;
        Password=Hare#2022;
        Pooling=false;SslMode=none;convert zero datetime=True;";

    public CustomersModel()
    {
    }

    public DataTable GetAllCustomers()
    {
        MySqlConnection dbcon = new MySqlConnection(connectionString);
        dbcon.Open();
        MySqlDataAdapter adapter =
            new MySqlDataAdapter("SELECT * FROM CUSTOMER;", dbcon);
        DataSet ds = new DataSet();
        adapter.Fill(ds, "result");
        DataTable CustomersTable = ds.Tables["result"];
        dbcon.Close();
        return CustomersTable;
    }
}

```

Nytt anrop till databasen vid varje anrop till GetAllCustomers()
Fördel: Alltid rätt info
Nackdel: Kan vara påfrestande för DB

```

public class CustomersModel
{
    private string connectionString =
        "Server=localhost;Port=3308;Database=a00leifo;
        User ID=sqllab;
        Password=Hare#2022;
        Pooling=false;SslMode=none;convert zero datetime=True;";

    private DataTable CustomersTable;

    public CustomersModel()
    {
        MySqlConnection dbcon = new MySqlConnection(connectionString);
        dbcon.Open();
        MySqlDataAdapter adapter =
            new MySqlDataAdapter("SELECT * FROM CUSTOMER;", dbcon);
        DataSet ds = new DataSet();
        adapter.Fill(ds, "result");
        CustomersTable = ds.Tables["result"];
        dbcon.Close();
    }

    public DataTable GetAllCustomers()
    {
        return CustomersTable;
    }
}

```

Endast ett anrop till databasen vid skapandet av modellen när applikationen startas.
Fördel: Snabbare svarstid vid anrop till GetAllCustomers()
Nackdel: Om kunder läggs till efter applikationsstart så är informationen felaktig

Deklarera vyer

De webbsidor som visas upp kallas vyer och dessa skrivs i filer som avslutas med .cshtml

Projekt-strukturen måste efterföljas. Det vill säga att vyn för end-point Index som hanteras av kontrollern Home måste finnas i:

/Views/Home/Index.cshtml

```

> Controllers
> Models
> Properties
✓ Views
  ✓ Home
    Index.cshtml
    SearchInvoiceRows.cshtml
  > Shared
    _ViewImports.cshtml
    _ViewStart.cshtml

```

Uppbyggnaden av en vy

I filen `_ViewStart.cshtml` deklareras vilken fil som innehåller startpunkten för HTML-koden som skall genereras.

Normalt är det filen `_Layout.cshtml` som återfinns i mappen `/Views/Shared` som håller start-koden.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - @Max6_2018</title>
  </head>
  <body>
    <div class="container body-content">
      @RenderBody()
    </div>
    <footer>
      <p>©copy; 2018 - @Max6_2018</p>
    </footer>
  </div>

  @RenderSection("Scripts", required: false)
</body>
</html>
```

Här skapas den specifika vyns HTML

Razor-kod, server-kod som gör sidan dynamisk @ indikerar Razor-markup jmf <?php ?>

```
@{
    ViewBag.Title = "Index";
}
<h2>Find Invoice Rows</h2>
<form method="post" action="@Url.Action("SearchInvoiceRows", "Home")">
  <label>
    Customer:
    <select name="custno">
      @foreach (var row in ViewBag.AllCustomersTable.Rows)
      {
        <option value="@row["CUSTNO"]">@row["NAME"]</option>
      }
    </select>
  </label>
  <input type="submit" value="Search" />
</form>
```

ViewBag, en special-variabel som tillåter controllern att skicka data till View-genereringen

```
//Controllers/HomeController.cs:
...
public IActionResult Index()
{
    ViewBag.AllCustomersTable = CustomersModel.GetAllCustomers();
    return View();
}
...

//Views/Home/Index.cshtml:
...
<select name="custno">
  @foreach (var row in ViewBag.AllCustomersTable.Rows)
  {
    <option value="@row["CUSTNO"]">@row["NAME"]</option>
  }
</select>
...

```

Namn och datatyp är av vikt vid POST/GET

```
//Views/Home/Index.cshtml:
<form method="post" action="@Url.Action("SearchInvoiceRows", "Home")">
...
<select name="custno">
  @foreach (var row in ViewBag.AllCustomersTable.Rows)
  {
    <option value="@row["CUSTNO"]">@row["NAME"]</option>
  }
</select>
...
</form>

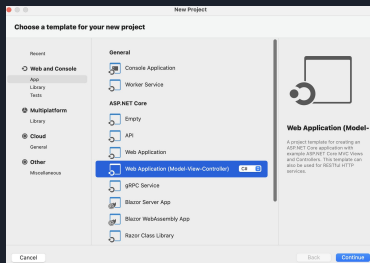
//Controllers/HomeController.cs:
public IActionResult SearchInvoiceRows(string custno)
{
    ViewBag.SearchResults = invoiceRowsModel.SearchInvoiceRows(custno);
    return View();
}
```

Skapa ett nytt projekt i Visual Studio

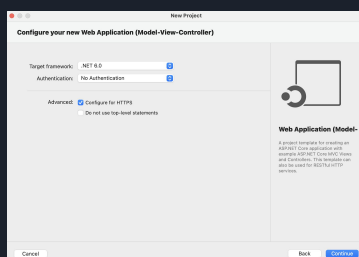
Notera att utseendet
skiljer sig mellan
Mac/Linux/Windows

Det du vill välja är:
Web Application
(Model-View-Controller)

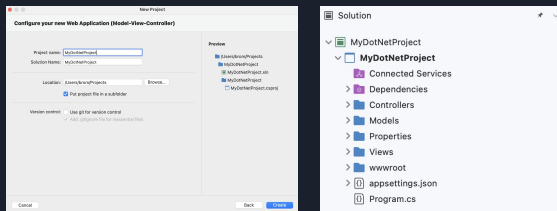
C#



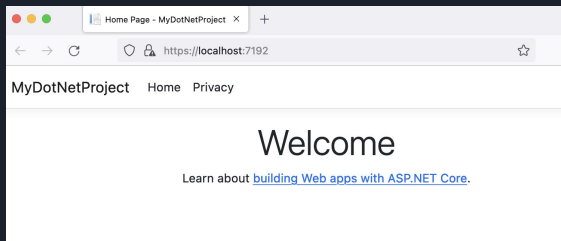
Välj .NET 6.0



Namnge projektet och skapa sedan projektet



Template sidan



Rensa ut template-kod

Det finns ingen tom template utan Visual Studio skickar med en massa kod som inte behövs. Om du vill rensa bort denna kod så går det bra.

Du hittar den mesta av koden som kan rensas bort i:

`/Views/shared/_Layout.cshtml`

OBS! Var noga med att inte rensa bort kod-raden med `@RenderBody()`
