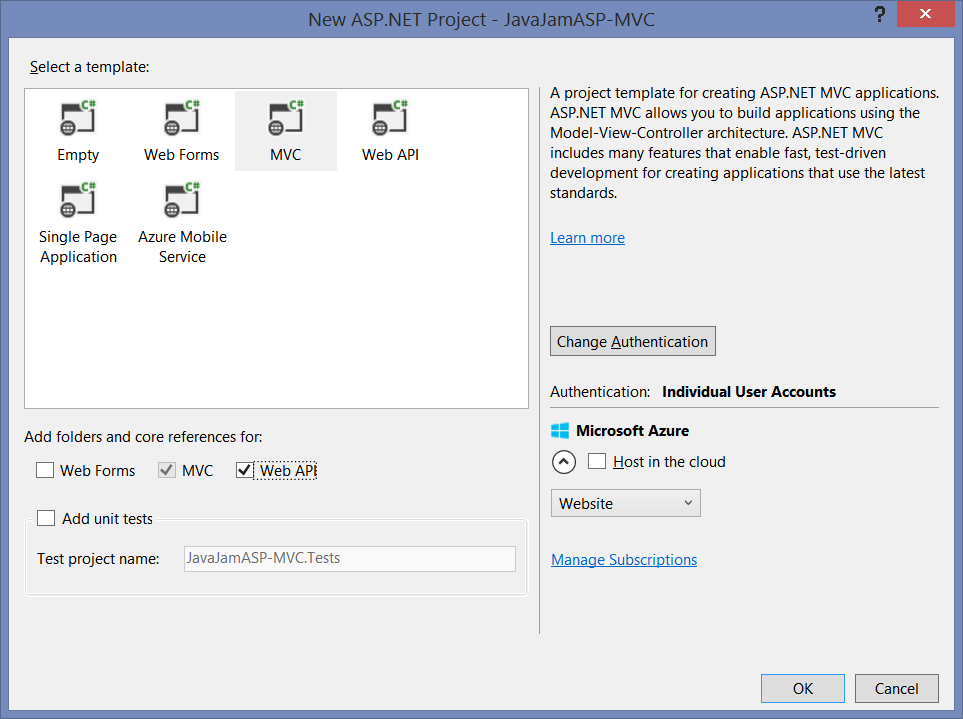
# JavaJamAjax

# Kommentarer til løsningsforslag

*Det var ikke en del af opgaven, men jeg har valgt at bruge Bootstrap i mit løsningsforslag. Så jeg startede forfra med at oprette et nyt projekt, og har så kopiret de forskellige elementer over fra løsningsforslaget JavaJamASP.  
Bootstrap ligger op til en top-menu (fungerede bedst på smartphone), så jeg har flyttet menuen til toppen. Header’en med billedet som overskrift fylder unødigt på en smartphone, så det skjules når skærmbredden er lille (class=*"hidden-xs").   
*Jeg omskriver også Jobs-siden til at bruge en modelklasse (*JobApplicant*) og* @Html-helpers.

### Lab1

Opretter et projekt af typen ASP.NET MVC, og tilføj Web API.



Laver de samme ændringer og tilføjelser som i løsningsforslaget JavaJamASP, men undtagelse af menuen, der forbliver i toppen, og fordi jeg benytter Bootstrap, er der ikke så mange styles, som skal flyttes over.

**AJAX optimering**:  
Nu er jeg så klar til det som opgaven egentlig går ud på: at benytte ajax-teknikken.

På **serversiden** gøres det ganske enkelt ved brug af denne kode (som indsættes i alle controller-actions):  
 if (Request.IsAjaxRequest())

{

return PartialView();

}

return View();

På **clientsiden** (i de forskellige Views) skal alle interne links ændres til ajax-request. Dette gøres med javascript funktioner som denne:  
 $('#homeLink').click(function (event) {

event.preventDefault();

var url = $(this).attr('href');

$('#content').load(url);

});

Og så skal alle actionLinks tilføjes en passende id:  
 <li>@Html.ActionLink("Home", "Index", "Home", null, new { id = "homeLink" })</li>

<li><a href="/Home/Menu" id="menuLink">Menu</a></li> @\*Just to show… \*@

<li>@Html.ActionLink("Music", "Index", "Music", null, new { id = "musicLink" })</li>

<li>@Html.ActionLink("Jobs", "Index", "Jobs", null, new { id = "jobsLink" })</li>

Alt dette er lige efter “bogen”, men der opstår et problem. Når man skifter til music-siden med et ajax-kald, så bliver de javascript funktioner, der skal initiere music-siden ikke kørt. Dette er jeg nødt til selv at kode, hvilket jeg gør ved at tilføje en callback-funktion til ajax-kaldet load:  
$('#musicLink').click(function (event) {

event.preventDefault();

var url = $(this).attr('href');

$('#content').load(url, function() {

$.getScript("/Scripts/music-lab1.js");

});

});

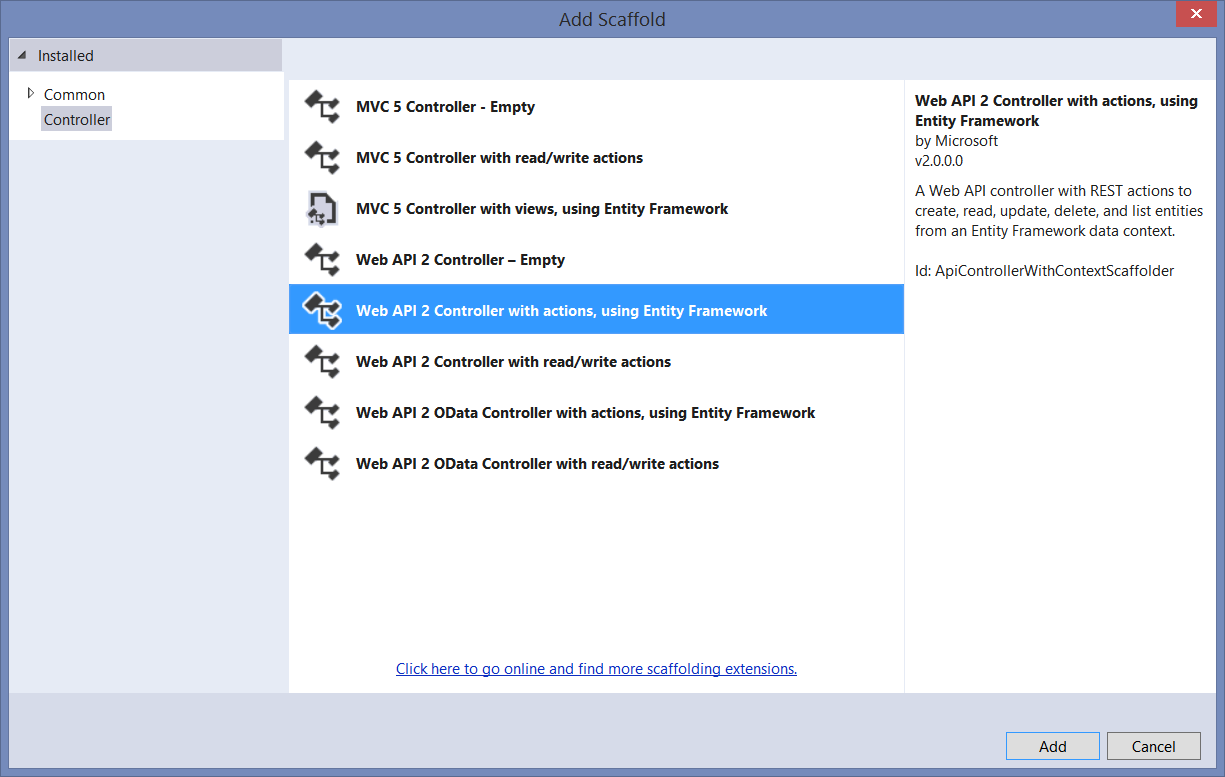
Jobs siden kræver også særbehandling. Når man bruger ASP.Net MVC’s hjælpefunktioner til at opbygge en form så bruges jQuery.Validate til at validere bruger input ude i browseren. Når siden loades via en almindelig http-get (fuld side) så gøres dette via:  
@section scripts {  
 @Scripts.Render("~/bundles/jqueryval")

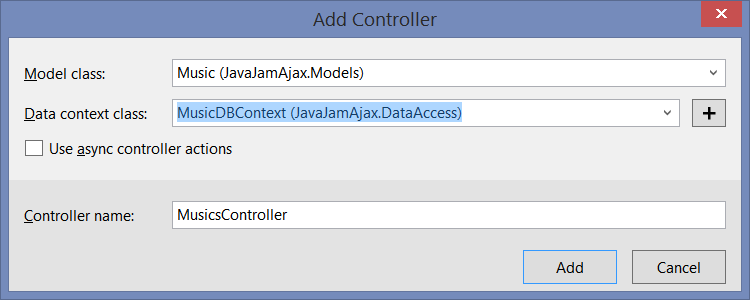
}

Men når siden vises via et Ajax-kald (XmlHttpRequest), så skal jQuery.Validation filerne loades manuelt. Derfor er der 2 udgaver af Jobs siden (Index.schtml og JobsAjax.cshtml).

### Lab2

Laver først en modelklasse Music og en EF Context-klaase MusicDBContext. Opretter en Web.API-controller med navnet Musics.





Jeg har valg kun at lægge path til billederne ind i databasen. Til at teste mit web.api (MusicsController) bruger jeg en Crome-app (extension) ved navm Postman. Den er genial til at test et web.api, og jeg bruger den også til at lægge nogle data ind i databasen (via controlleren). Se hvorledes jeg bruger Postman i videoen.

Næste trin er at omskrive musiksiden til at hente dataene fra mit web.api, og så opbygge siden i klienten. Dette kan gøres på 2 forskellige måden: via en html tekststreng eller via Javascript kode som f.eks. document.createElement("H1");. Jeg vælger den første, da den kræver mindst kode, og jeg bedre kan genbruge den html jeg har fra lab1.

