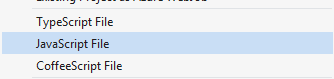
# Del2: WebService Frontend

I denna del kommer vi att skapa klienten för applikation med hjälp av HTML, javscript, jquery och bootstrap. Följande funktioner kommer att implementeras:

* Lista alla filmer
* Visa detaljerna för en film
* Lägga till en ny film

Vi kommer att använda jQuery för att göra anrop till vårt API.

Det första vi gör är att lägga till en javascriptfil för vår applikation. Detta gör vi genom att högerklicka på mappen scripts, välj add och sedan JavaScript File. Döp denna till app!



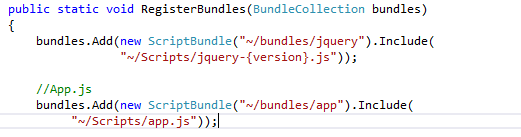
Vi börjar med att göra en checker för att se så att den laddas korrekt på vår hemsida. I din app.js, skriv

console.log(”testing”);

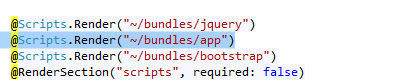
och spara filen.

Vi måste nu inkludera den i vårt projekt så att den laddas hem av browsern när vi går in på vår url.

Scrolla upp till mappen App\_Start och öppna filen BundleConfig.cs. I denna fil vill vi skapa en ny bundle med vårt skript. Vi gör detta i RegisterBundles-funktionen och lägger till den under Jquerybundlen med följande kod.



Vi kan nu gå in i mappen Views, sedan shared och välja \_Layout.cshtml. Här länkar vi nu in vår nya scriptbundle!



Starta nu MVCapplikationen. När sidan har laddat klart, tryck på f12 på tangentbordet alternativt högerklicka på hemsidan, granska komponent och sedan gå till fliken console. Om texten ”testing” dyker upp fungerar allting, annars testa att ladda om och om det fortfarande inte fungerar se så att du har följt stegen rätt. Det är även viktigt att vi laddar in vår bundle efter jquery då vårt script kommer att använda sig av jquery.

Vi ska nu göra en superduperenkel förfrågan mot vårt api med hjälp av jquery och dess GET-metod. Vi kan använda jquerys ajaxanrop men för GET och POST finns det simplare funktioner.

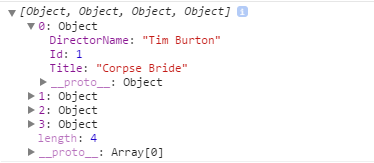
Dokumentation om jquery.get hittar du på <http://api.jquery.com/jquery.get/>

I din js-fil, ta bort din console.log och byt ut en mot:



Det första argumentet är en URI till ditt API för att göra requests. Du hittar alla sådana under API-fliken på din egen hemsida. Den andra parametern är en anonym funktion som kommer att köras när svaret kommer tillbaka från servern med data.

Kolla i konsolen i din webbläsare och öppna svaret.

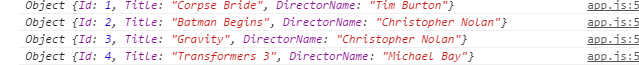


Här ser du det svar du får från servern med din GET-request. Detta är i JSONformat och svaret är ett Javascriptobject.  
Vi kan loopa över denna med antingen jquerys .each eller en vanlig for-loop i ren javascript. Jag kommer att använda javascriptloopen eftersom jag anser att det är klarare vad som händer med den. Lägg nedanstående kod i den anonyma funktion som är ett callback till $.get vi skrev innan.



Vi skapar en variabel movie som representerar en nyckel i objektet movies. För varje movienyckel vi hittar (som en foreach i c#) refererar vi dens plats i movies-objektet genom att peka på dess index som i en array.

Detta kommer att ge oss följande resultat i konsolen:



När vi skickar en GET-request till servern med /api/movies som argument gör vi alltså en förfrågan mot vårt API som sedan returnerar det MovieDTO vi skapade i gårdagens uppgift. Denna data kan vi nu använda för att berika vår sida med innehåll från databasen!

Vi kan bryta ner den förenklade get requesten till en ren ajax-request för att se hur den verkligen funkar. Vi kan kommentera ut den kod vi tidigare skrivit i app.js och definiera en funktion som heter GetData.



I denna funktion använder vi jquerys ajaxmetod för att bygga en GET - request. Nedanför ser en enkel uppbyggnad för ett anrop med ajax och vilken information vi behöver. POST och PUT behöver även en data-nyckel med det objektet vi vill sätta in. Ett exempel på det kommer längre ner.



För vår GET request vill vi hämta alla filmer vilket innebär att vi vill anropa APIt med URI api/Movies



Eftersom vi ska HÄMTA alla filmer vill vi att vår type är GET.



Och om det funkar vill vi köra en funktion. I exemplet endan har jag deklarerat en anonym funktion som kommer att vara ett callback om vårt api returnerar att operationen gick bra!

