

Patric Keisala

Utveckla ett C# bibliotek till

Spectrum Digital Solutions AB

Examensarbete 40 YH-poäng

FRONTEND WEBUTVECKLARE MALMÖ 2015

Datum: 2017-05-28

Handledare: Thomas Montgomery, Erik Tinebo

KYH Yrkeshögskola

www.kyh.se

Förord

Detta är ett examensarbete för Front End Web Development, KYH i Malmö motsvarar 40 YH-poäng (8 veckors studier). Examinatorn är Sohail Hashware och mina handledare är Emil Tinebo och Thomas Montgomery på Spectrum Digital Solutions AB Larochegatan 5 i Malmö. Spectrum Digital Solutions AB har försett mig med uppdraget att skapa ett bibliotek som dem är i behov av på sina kunders webbsidor. Tack Spectrum Digital Solutions AB, Emil och Thomas för all hjälp, vägledning och detta uppdrag som lärt mig väldigt mycket.

Sammanfattning

Tack vare Spectrum Digital Solutions AB fick jag möjlighet att arbeta i ett projekt som utmanar mig mer än att bara visa vad jag kan göra. Spctrm är ett konsultbolag som hyr ut personal till flertal större företag samt att de bygger och hjälper företag med olika IT lösningar. Spctrm hade behov utav ett bibliotek som kunde spara dem tid vid implementering av olika nyhetstjänster som Cision och Mynewsdesk som många av deras kunder idag har på sin hemsida, samt att det ska spara in på upprepande kod och fortsätta träna studenten i denna programmerings princip. Detta är byggt i C# .NET.

Abstraction

Thanks to Spectrum Digital Solutions AB, I had the opportunity to work in a project that challenges me more than just showing what I can do. Spctrm is a consulting company that leases staff to several major companies as well as building and helping companies with different IT solutions. Spctrm had the need of a library that could save them time in implementing various news services like Cision and Mynewsdesk, which many of their customers today have on their website, and that it will save on repeated code and continue training the student in this programming principle, This is built in C # .NET.

Innehållsförteckning

[1. Inledning](#_Toc479756228) 1

[Bakgrund](#_Toc479756229) 1

[Syfte och mål](#_Toc479756230) 1

[Problemformulering](#_Toc479756231) 1

[Avgränsningar](#_Toc479756232) 2

[2. Genomförandet](#_Toc479756233) 3

[Planering](#_Toc479756234) 3

[Metodik](#_Toc479756234) 4

[Genomförande](#_Toc479756234) 4

[3. Resultat](#_Toc479756235) 7

[4. Slutsatser](#_Toc479756236) 9

[5. Referenser](#_Toc479756237) 10

[7. Bilagor/Appendix](#_Toc479756238) 10

# 

# **Inledning**

## Bakgrund

Jag är tacksam att min uppdragsgivare Spectrum Digital Solutions AB kunde finna ett uppdrag till mig som gav mig en massa ny kunskap och nytänk. Spectrum Digital Solutions AB är ett konsultbolag som jobbar inom IT eller management såsom projektledning och kravhantering och de har ett stort antal kunder som de hyr ut personal till. Företaget har ett bra antal kunder som använder sig utav nyhets tjänster så som www.Cision.se och www.Mynewsdesk.com och dessa har man programmerat in på var och en hemsida utan att använda någon form av bibliotek, det tyckte dem vart dags att ändra på.

## Syfte och mål

Spctrm idag har ej använt sig av något bibliotek för att implementera nyhetstjänsterna på alla de hemsidor som är i behov av tjänsten, detta har inneburit att utvecklarna får lägga extra tid på detta arbete.

Biblioteket skall kunna spara in på detta arbete i framtiden för att ge dem tid att kunna fokusera extra tiden på andra delar.

Ett syfte är även att få bort all upprepande kod och arbete, samt för mig att utveckla ett arbetssätt för att förebygga detta och att jag får en större kunskap om objektorienterad programmering.

## Problemformulering

Spctrm gav mig nycklar till två nyhetstjänster Cision och Mynewsdesk, men i framtiden kan det kommas att byggas vidare med fler tjänster så det var ett par viktiga punkter som behövdes uppfyllas i projektet.

* **Det skall vara simpelt att integrera i ett projekt.**

Inte bara ska det vara lätt att hämta datan från biblioteket, det skall även vara lätt att använda sig av datan. Cision, Mynewsdesk och de andra kommande har en del fält som de har gemensamt så som ex. Id, Url, Images, dessa fält ville man skulle ha ett gemensamt namn och struktur.

* **Det skall vara simpelt att utöka biblioteket.**

Det största problemet i att utöka projektet var Deserializeringen. De olika nyhetskällorna har olika strukturer och namn och det gör att Deserializeringen inte går att kombinera, utan man får skapa ett eget objekt för var nyhetskälla. Detta är den svåraste delen att få rätt när man skall utöka projektet så en dokumentation och exempel behövdes fås fram.

* **Det ska lösa problemet med repeterande kod.**

Projektet var planerat så att utvecklarna skulle slippa skriva om koden i varje projekt som de ville implementera nyhetsflödet i. Jag valde dock även att ta detta ett extra steg och se till att få så mycket jag kan i biblioteket att vara återanvändbar kod, vilket även var en del av varför detta projekt valdes, att få fortsätta att jobba på ett sätt så att man motverkar onödig upprepad kod.

## Avgränsningar

På grund av tidsbrist kunde inte projektet visas integrerat med ett JavaScript ramverk.

Däremot om man kör C# .NET med MVC har man möjlighet att enbart ha det som backend och få klienten som vilket ramverk man än vill, vilket möjliggör att använda projektet.

Man kan även med AJAX kalla på en C# method och på så sätt få tillbaka data, jag har implementerat JSON Encoder för just detta syfte om de väljer ett Javascript ramverk i framtiden.

Någon effektkarta skapades ej i början, då kraven var så simpla och det var inget fokus på gränssnittet. Gränssnittet byggdes bara som hastigast i slutet när allting var klart om där fanns tid till det, det absolut viktigaste var biblioteket. Gränssnittet skulle bara vara en liten fingervisning åt utvecklarna.

# **Genomförandet**

## Planering

Vecka 11 : Förstudie av källorna, testning och OOP, sätta upp projektmiljöer

Vecka 11 - 12 : Planering av projektet och påbörja diagram (Klass-, Sekvens-, Tillståndsdiagram)

Vecka 13 - 16: Uppbyggnad av bibliotek samt test app, dokumentation byggs på under utveckling, Testning utförs före eller efter beroende på vilka testmetoder för studien finner.

Vecka 17 : Rapportskrivning och analyser.

Till start fanns ett påbörjat bibliotek som hjälpte mig att göra tester mot och lära sig av. Under första veckan testade jag hur lång tid biblioteket skulle ta att ladda, och därifrån kunna komma på vad som skulle behöva göras. Till en början granskade jag svarstiden hos Cision, vilket hade en medeltid på 3 sekunder, och om man som det gamla biblioteket var uppbyggt valt att hämta alla poster, så fick man räkna med minst 23 sekunder upp till en minut, bara för datorn, vilket jag anser är för lång tid att vänta. Sen kollade jag in hur projektet var uppbyggt och vad som kunde förbättras. Sen tänkte jag framtidssäkra, det projektet hanterade enbart XML data, vad händer om man har behov av JSON, att hämta samt som utdata.

Förstudien gav följande planering.

* **Uppmuntra att utvecklarna bygger sidan så att det inte blir så långa svarstider.**

Genom att inte implementera en loop för att hämta all data och senare på klient sidan visa hur man enkelt kan dela upp detta på sidor. Om utvecklaren sedan vill hämta all data, så bör det inte vara svårt att bygga in detta i en loop.

* **Includera frivillig Caching för att drastiskt kunna reducera svarstiden mer.**

För de sidor som inte lägger till nya nyheter så ofta kan jag reducera svarstiden till under 100ms.

* **Includera hantering av JSON data.**
* **Bygga om projektet så att det hela är byggt med återanvändbar kod.**

## Metodik

## Valet av programmeringsspråket C# är för att utvecklarna för det mesta implementerar detta i .NET vilket är deras specialitet, men det är inget hinder, detta gav mig chansen att programmera mer i OOP vilket inte rördes vid så mycket under skoltid men som man väl kan finna i PHP, wordpress exempelvis har en del av sin kod uppbyggd i OOP PHP. Det tillåter mig även att programmera program och native applikationer om jag skulle ha behov av det i framtiden. Däremot har detta bibliotek även byggts upp med det funktionella programmeringstänket då detta är hur jag vanligtvis programmerar, vilket resulterade i en större del bestående av statiska metoder byggt för att returnera ett visst arbete.

## Jag valde att inkludera ett publikt bibliotek som heter Newtonsoft.JSON för att hantera Json datan, till skillnad från XML paketet som Microsoft har hand om. Detta är det enda externa paket eller Nuget som det kallas inom C#, fungerar på liknande sätt som Npm eller yarn. Detta valdes för att det var det mest populära paket, däremot var dokumentationen på paketet minimalt och skapade problem och lång utvecklingstid.

Valet av MVC i .NET är just att det var det packat i visual studio jag hade något förståelse för då jag fann att detta liknade Angular 1 till en viss del. Eftersom gränssnittet inte var högsta prioritet så valde jag att ta det säkra kortet.

## Genomförande

Under första veckan började jag med att kolla upp vad för data jag fick in från Cision, hur strukturen var och hur lång tid det tog att få datan.

Jag testade medeltiden på svarstiden från Cision tjänsten och fick den till runt tre sekunder, sedan gjordes det tester mot det gamla projektet och fick fram att vid användning av det. Utan caching och vid hämtning av alla poster, kunde hämtningen ta från 23 sekunder till en minut, vilket jag ville försöka motverka genom att visa hur man kan bygga hemsidan på istället.

Jag kollade sedan upp hur första projektet hanterade jobbet och vad som kunde förbättras, som jag nämnde i Problemformulering så ville jag bygga det mer för att ta in återanvändbar kod principen. Jag då valde även att bygga det helt från scratch, vilket även skulle hjälpa mig att lära mig C# för detta projekt, samt att det är lättare än att modifiera och ändra om allt.

Jag ville även jobba på ett sätt så att jag kunde blanda in tester i projektet. Visual Studio för C# har inbyggda funktioner man kan använda genom att välja Nytt projekt -> Test -> Unit Test Projekt, så jag byggde en testmetod för alla sätt man kan använda biblioteket på, och inte för varje metod i programmet, det klarar det inbyggda debug verktyget av att kolla. Med tanke på att C# är så strikt med typer och vad man får returnera, så var det rätt lätt att modifiera tills testerna inte fallerade längre, och alla tänkbara sätt som man kan tänka sig att programmet kan gå fel på, hanteras på rätt sätt. Däremot som C# nybörjare är det svårare att bygga upp metoderna i början.

I slutet av första veckan fick jag tillgång till en livesite som hade implementerat Cision, däremot använde de en nyckel som ger json data vilket var lite oroande. Cision är uppbyggt på det sätt att du har en nyckel för XML data och en för JSON. Jag märkte även av att spctrm använde sig av javascript ramverk för att publicera dessa nyheter på sidan sen, så då ansåg jag att det var det bästa att implementera en JSON hantering i projektet. Då var först tänkt att man jag vill kunna hämta data med xml nyckel eller json, när det gäller cision, så jag hittade JSON.NET skapat av Newtonsoft. Jag försökte även senare i projektet kombinera XML och JSON Deserializeringen, båda fungerar så att du får sätta upp klasser som är uppbyggda efter hur datan struktur du får, Dessa klasser kan man se i Cision -> Xml Serialization och Mynewsdesk -> JsonSerialization.

Denna idén skrotade jag senare då det var för komplicerat, det hade gått fortare att bygga upp de för sig då man enbart gör det en gång om man bygger på biblioteket. Ifall de ville implementera Cisions JSON nyckel, så lär de använda AJAX och då e det mycket bättre att strunta i biblioteket då man hade laddat in onödig kod, och direkt hämtat datan från Cision.

Jag valde dock att bygga mynewsdesk med jsonserializeringen då denna nyhetstjänst tillåter att man hämtar xml eller json med samma nyckel, största delen av detta beslut är för jag vill själv kunna hantera json data i framtida projekt när jag jobbar mot ett javascript ramverk, då är det bra att ha denna kunskap. Sen ville jag även att utvecklarna har två val vid utökning av biblioteket att kunna använda json.net som jag byggt upp och dokumenterat i Spctrm\_News -> documentation.txt. Dåda verktygen har sina fördelar, Microsoft XML är äldre, bättre dokumenterat, bättre utvecklat enligt min mening än json.net, däremot kan du inte encoda till json vilket json.net gör väldigt smidigt. Denna funktion fann jag viktig för att uppmuntra att de bygger framtida sidor i nått form av javascript ramverk, då .NET är server script. Jag önskar att jag funnit mer tid till att gräva mig in i MVC.

Tanken var att påbörja alla diagram till projektet så det var fokusen i starten av andra veckan. Jag läste på hur jag skulle gå tillväga med de inbyggda programmen, för att få fram sekvens och tillståndsdiagram behöver jag installera VS ultimate 2013 då de nyare gratis versionerna enbart hade UML inbyggt. Detta gick inte alls, hade ingen som helst aning om vad jag behövde i programmet eller vart jag skulle börja, så denna ideen sköt jag fram tills efter biblioteket var klart och då fick jag fram samtliga diagram vilket ni finner bland bilagorna.

Jag påbörjade byggandet av biblioteket dag två av andra veckan. Genom att göra om det från början såg jag till att lära upp mig själv på ett sätt jag vet jag lär mig på samt att jag då direkt kunde bygga upp det på ett sätt som jag ville se det hela på, vilket är att få så mycket som möjligt i återanvändbar kod.

Under uppbyggnadens gång kollade jag vidare på flertal saker, och testade olika tekniker, i början försökte jag bygga det hela som jag hade gjort i PHP eller Javascript, Arrays och object fick jag inte att bete sig som jag är van vid så det var otroligt frustrerande i början, arrays och objekt som jag känner de nu är otroligt strikta, hur du skapar arrays är såhär, (vilken typ) int[] numbers; detta är en single-dimension array, och för att ha fler dimension array får du ange det med [ , ] eller om du vill ha en array med arrays [] [], mer om detta finns under ”Arrays Tutorial” på microsofts hemsida1. Kan säga att jag idag inte vet om det är möjligt att göra som i javascript eller PHP, men något jag planerar att få lära i framtiden, men som tur är kom jag runt det och har inte behov av detta längre.

Fick tips att kolla upp ”Constraints on Type Parameters”2 och ”Generic Methods”3, vilket är helt underbart att kunna efter att man suttit fast nån dag, kort sagt om dessa är att du kan bygga mer dynamiska metoder genom att när du kallar på metoden då ange vilken typ den ska jobba med då c# är strikt med att jobba med typer, detta kan du sen använda var du är behöver i metoden, ett exempel är i News.cs metoden xml\_deserialize. Dessa lägger jag till i referenser mest för det att om någon hamnar i C# i framtiden så är detta fantastisk kunskap.

Efter all programmering för Cision var klart, inkluderat alla gemensamma metoder i News.cs jobbade jag med serialiseringen, detta var den stora delen av det hela, och som tog mest tid, försöket att kombinera alla till en tog lång tid och resulterade i att jag skrotade det hela som beskrivet tidigare på förra sidan. Detta arbete med de olika serialiseringarna pågick ett par veckor ändå till slutet av projektet, med försöket att kombinera, lära, dokumentera och få ut rätt resultat.

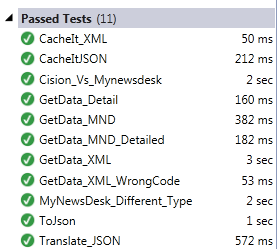
Någon vecka innan allt va klart med satte jag mig ner med MVC, vilket står för ”Model, View and Controller” vilket jag vill säga är liknande Angular 1.

Detta var för att kunna se att jag verkligen fick alla resultat som jag ville och att jag kunde använda det på rätt sätt, jag la ingen tid på styling eller så utan enbart la till det minimala för att visa vad som händer och vad man kan göra med biblioteket.

# **Resultat**

Bibliotekets krav är färdiga, det är simpelt att integrera, lätt att bygga ut och det tar hand om återkommande kod. Kodmässigt har jag reducerat rejält från det förra projektet samt gjort metoder som enbart var byggt för en nyhetstjänst att nu kunna användas av samtliga nyhetstjänster, detta har även gjort det extremt enkelt att bygga på biblioteket med fler nyhetskällor, utöver kraven har jag även lagt till stöd för JSON data, både för encoding och decoding, påbörjat ett gränssnitt, dokumentation samt testfall till utvecklarna som kan användas som exempel. Om det visar sig att utvecklarna har behov av en loop för att hämta all data direkt ser jag inga problem att integrera det, jag valde att inte göra detta för att uppmuntra att utvecklarna tänker på svarstiderna från tjänsterna.

Alla testfall i Figur 1.1 Visar att samtliga testfall ligger på en bra svarstid, jag gick snabbt efter svar jag hittade på ett forum post Quora4 men försökte få det så lågt jag kunde.

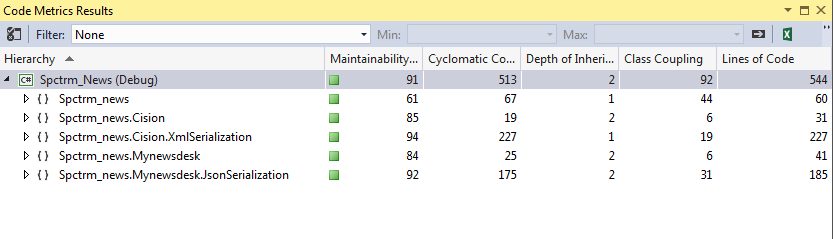


Figur 1.1

De testfall över sekunden är just de testfall som använder sig utan Cisions data. Men det är på grund av deras server. Däremot om man använder sig av cache kommer man vanligen under 100 ms vilket är föredraget att använda sig av om möjlighet finns.

Det finns även ett annat verktyg i visual studio som kontrollerar hur komplext ens program är och hur lätt det är att underhålla som kallas Code Metrics, vilket man finner under Analyze –> Calculate Code Metrics, Figur 2.2,

Detta verktyg som ger en bättre inblick om vad som behövs testas mer eller möjligen byggas om om det är riktigt illa, men för oss ger det bra resultat.



Figur 2.2

Ju högre Maintainability index och ju lägre Cyclomatic Complexity desto bättre.

Detta visar oss att programmet är relativt lätt att underhålla.

Hade önskat mer resultat med MVC gränssnittet, men det absolut viktigaste är med.

Det visar hur man kan använda biblioteket och hur man renderar det. Det visar hur Queryn till biblioteket ser ut, innehåller en dokumentation och visar hur man kan skapa sidor utan att behöva ladda in all data direkt, vilket var ett av mina egna mål för att uppmuntra att tänka på svarstider.

Det var tänkt att visa samma med ett JavaScript ramverk, men det fanns inte tid alls till det.

# **Slutsatser**

Att få arbeta med C# har verkligen öppnat vägen till nya kunskaper och jag känner att jag kommer ha väl nytta av detta. Inte bara att jag nu är mer kunnig inom objektorienterad programmering, att jag nu är betydligt mer kräsen när det gäller att återanvända kod och att C# öppnat vägar till annat än bara hemsidor. Utan till mer att jag nu har kunskap att även kunna arbeta mer med t.ex. native appar i JAVA, eller Unity, windows program och det känns bra.

Jag blev även mer glad i att utveckla som jag gjorde genom att använda testmetoder. Det inbyggda debug programmet hjälpte extremt mycket. Även då jag upptäckte analysverktygen i slutet är det en del jag kommer använda mig mer av.

Önskar att jag haft mer tid till allt för att ordentligt gå igenom allt. Hade t.ex. inte satt alla metoder som statiska om jag bara tagit en extra titt, men det var tidigt man lade in det och sen kollade man inte extra för det fungerade ju.

Thomas Montgomery har verkligen ställt upp och det skall han verkligen ha ett tack för. Han har sett till att jag haft en LIA plats och hittat ett projekt med betydelse som gett mig otroligt mycket kunskap under dessa månader.

# **Referenser**

1. Microsoft Guide, Arrays Tutorial

<https://msdn.microsoft.com/en-us/library/aa288453(v=vs.71).aspx>

2. Microsoft Guide, Constraints on Type Parameters

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/constraints-on-type-parameters>

3. Microsoft Guide, Generic Methods

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/generic-methods>

4. Quora topic: How long will the average user wait for a webpage to load before they abandon?

<https://www.quora.com/How-long-will-the-average-user-wait-for-a-webpage-to-load-before-they-abandon>

# **Bilagor/Appendix**

Bilaga 1. UML Diagram, För full UML diagram med Serialiserings object, kolla i projektet ULM\_full.png

Bilaga 2. Tillståndsdiagram

Bilaga 3. Sekvensdiagram

