|  |  |  |
| --- | --- | --- |
|  |  | |
| Systemdokumentation HAJK | |
|  | |

2016-10-10

**Innehåll**

[Allmänt 2](#_Toc463860781)

[Förberedelser 2](#_Toc463860782)

[Systemdokumentation HAJK2 3](#_Toc463860783)

[Kompilera javascriptapplikationer 4](#_Toc463860784)

[Kompilera c# applikation 4](#_Toc463860785)

[Förbered för driftsättning 8](#_Toc463860786)

[Driftsättning 9](#_Toc463860787)

[Uppdatera applikation 17](#_Toc463860788)

[Uppdatera admin 17](#_Toc463860789)

[Uppdatera client 17](#_Toc463860790)

[Uppdatera backend 17](#_Toc463860791)

# Allmänt

Plattformen Hajk är en applikationsplattform som är till för att distribuera kartor på webben. Plattformen består av tre underliggande applikationer som har benämningen client, admin och backend. Client är den applikation som används av slutanvändaren, admin är den applikation som används at kartredaktören och backend är den applikation som har hand som bakomliggande tekniska operationer så som konfiguration och utskrift. Backend har möjlighet att komma åt filsystemet på den server där den körs och spara filer åt användaren för nedladdning.

Källkoden är uppbyggd av ett antal komponenter där två olika tekniker används. En för att skapa moderna webbapplikationer i HTML 5 och JavaScript enligt standarden EcmaScript 2015, samt en som bygger på kodspråket C# och ramverket .NET 4.5.2. Driftplattformen för den senare bör således vara .NET-kompatibel, vilket körs nativt på Windows genom Internet Information Services och på Linux genom att använda mono eller ASP.NET Core, detta är dock inget som testats ännu.

Backend kompileras enklast med Visual Studio vilken finns att tillgå i en community edition.

Admin och Client använder ramverket React och filformatet .jsx. Stilfiler använder less för dynamisk CSS. Dessa filer behöver alltså kompileras innan driftsättning. Dessutom är källkoden programmerad med EcmaScript 2015 och bör därför transpileras till EcmaScript 5 innan driftsättning där äldre browsers som Microsoft Internet Explorer förekommer. I framtiden kan detta steg hoppas över för att köra applikationen nativt, men innan dess behöver EcmaScript 2015 implementeras av alla webbläsare.  
Client och admin använder till detta byggsystemet grunt. Grunt kompilerar, transpilerar, packar och minifierar samtliga källkodsfiler för driftsättning.

# Systemskiss

Hajk2-admin-x.x.x.js

Hajk2-x.x.x.js

Client

Admin

Backend

App\_Data  
layers.json

map\_1.json

Upload

Temp

REST-API

JSON

File access

.NET 4.5

ES2015

ES2015

REST-API

File access

# Applikationsarkitektur client

# Tredjepartskomponenter

## Backend

PdfSharp - *Skriver ut till PDF*

SharpMap - *Skapa kartor*

EntityFramework - *Objektsrelationer till SQL*

EntityFramework SQL Server Compact

Microsoft SQL Server Compact - *Filbaserad databas*

Newtonsoft JSON - *Dataformatering*

log4net - *Loggning*

## Client

underscore - *tilläggsbibliotek*

open layers - *kartmotor*

react - *gränssnittsmotor*

react-dom - *gränssnitt för webb*

moment - *hanterar tid*

proj4 - *hanterar projektioner*

marked - *skapar markdown*

x2js - *översätter från xml till json och vice versa*

jsts – JavaScript Topology Suite – används av bufferverktyget

jquery - *bibliotek för manuella DOM-manipulationer (används så sparsamt det går)*

# Admin

*client - Hajk.x.x.x.min.js*

react - *gränssnittsmotor*

react-dom - *gränssnitt för webb*

# Förberedelser

Den dator som används för att kompilera källkoden behöver vara utrustad med följande programvaror: git[[1]](#footnote-1), nodejs[[2]](#footnote-2), npm, grunt[[3]](#footnote-3), jsdocs[[4]](#footnote-4), Visual Studio[[5]](#footnote-5).

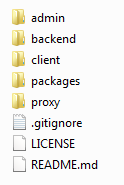
# Katalogstruktur för källkod

Källkoden beskrivs i dess grundläggande katalogstruktur.

Rotkatalogen har följande uppsättning av filer och mappar. Där de tre första innehåller de tre applikationer som beskrivs i avsnittet allmänt. Mappen packages innehåller referensfiler till Microsoft.NET och används av pakethanteraren NuGet. Denna mapp skall inte röras. Mappen proxy innehåller två stycken http-proxy-applikationer. Dessa är till för att genomföra http-anrop genom egen server om det finns behov av detta. Till exempel om användaren behöver genomföra korsdomänsanrop till andra kartservrar. Proxy-applikationerna behöver inte kompileras utan driftsätts direkt, mer om detta under avsnittet driftsättning.

Filen gitignore innehåller en beskrivning av de filer som inte skall versioneras av git.

LICENCE-filen innehåller en text som beskriver aktuell licens, för närvarande CC0 1.0 Universal. README.md är en markdownfil med enklare dokumentation om systemet.

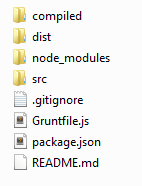


Mappen admin består av följande filer (efter första kompileringen). Innan kompilering kan utföras så måste beroenden laddas hem. Detta görs med kommando:

cd c:\[sökväg till rotkatalog]\admin

npm install

Detta kommando laddar hem alla beroenden och skapar upp lägger de i mappen node\_modules.



För att kompilera .jsx-filer mm så används programvaran grunt. Denna behöver installeras på datorn innan första användningstillfället. Kör kommando:

npm install -g grunt-cli

# Kompilera javascriptapplikationer

Adminapplikationen är beroende av client applikationen, så för att spara resurser delar de på gemensamma komponenter. Därför behöver client-applikationen vara kompilerad innan admin-applikationen kompileras. Beroenden till client applikationen laddas hem på samma sätt som till admin-applikationen, detta steg kan ta några minuter:

cd c:\[sökväg till rotkatalog]\client

npm install

För att kompilera de båda applikationerna kör följande kommandon:

cd c:\[sökväg till rotkatalog]\client

grunt dependencies

grunt build

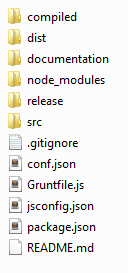
grunt release

cd..

cd admin

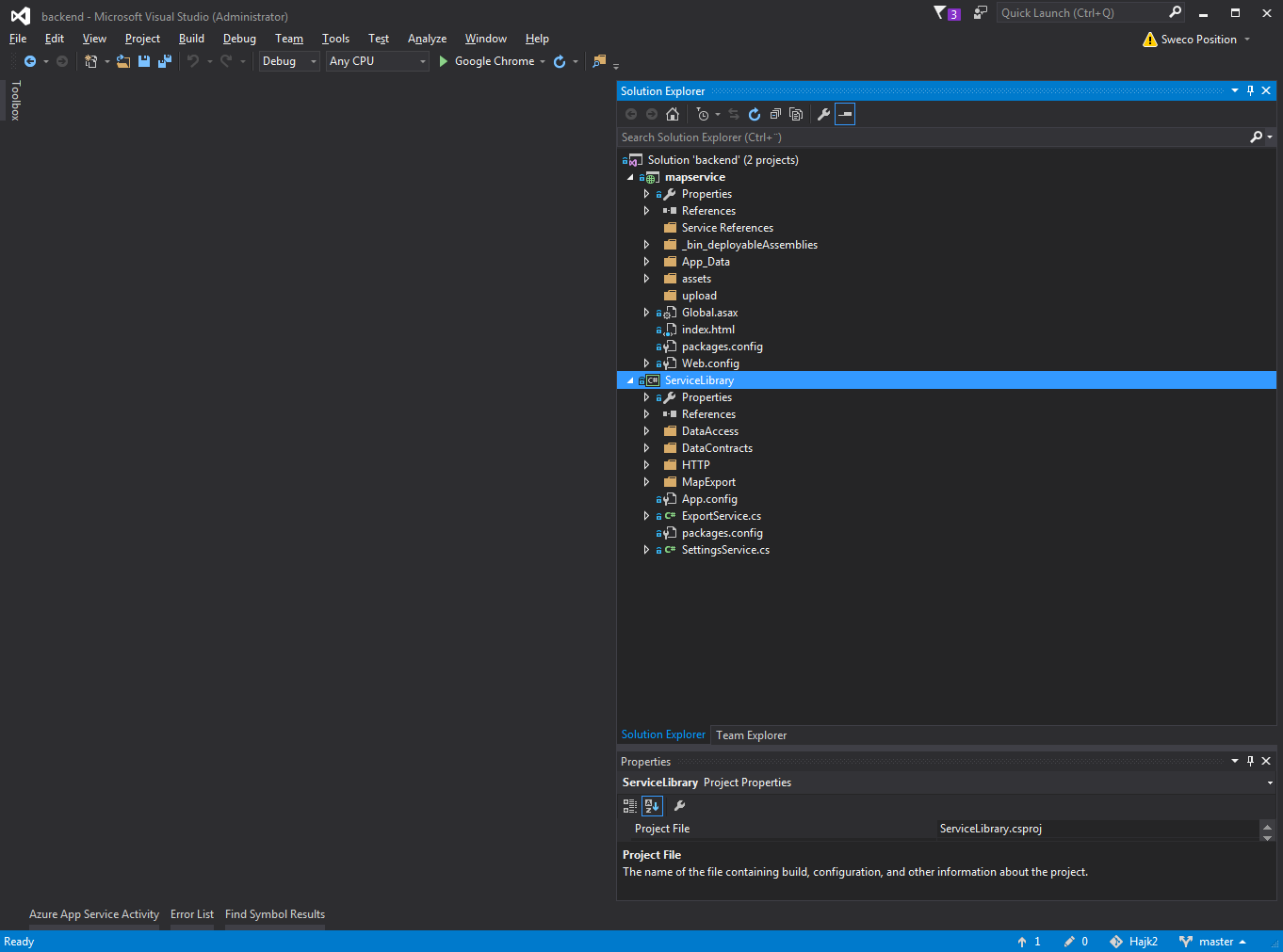
grunt

Kommandot grunt dependencies behöver endast köras första gången applikationen byggs eller när ett nytt externt bibliotek läggs till. Anledningen till detta är att det tar lång till att bygga alla beroenden med browserify om varje require-statement till externa resurser läggs till för varje fil. För att snabba upp processen så skapas en fil där alla globala variabler till externa bibliotek finns. När ovanstående kommandon körts så skapas för client-applikationen mappar dist och release. Dessa innehåller filer som är redo för driftsättning i testmiljö respektive produktionsmiljö.



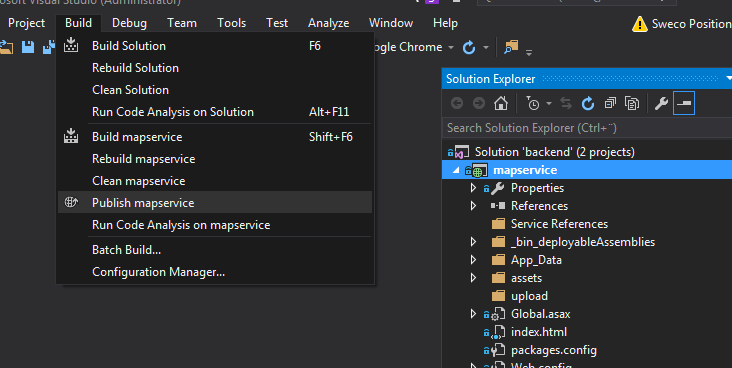
# Kompilera c# applikation

Använd utforskaren och navigera till c:\[sökväg till rotkatalog]\backend och dubbelklicka på backend.sln. Då startas Visual Studio och följande skärm visas:

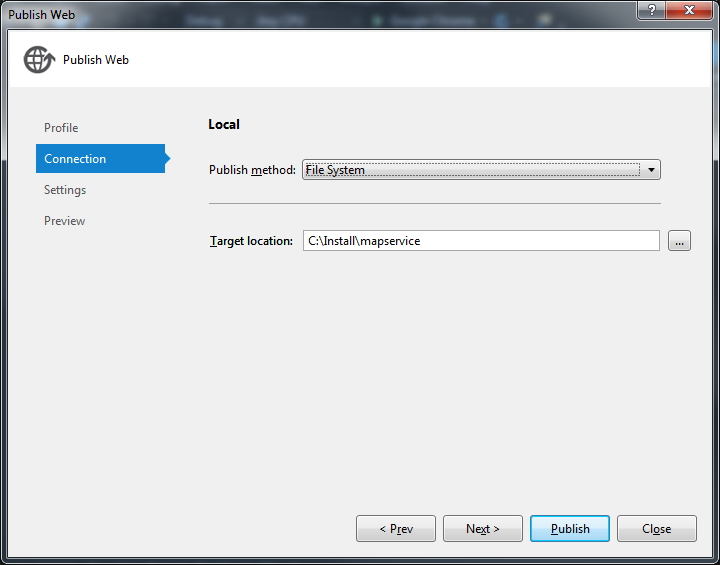


Håll inne ctrl och tryck sedan på w, följt av s för att öppna Solution Explorer. Från denna vy kan projektets källkodsfiler öppnas och granskas. Innan första kompilering görs så måste beroenden laddas hem via NuGet. Detta görs automatiskt genom ställa markören på mapservice, enligt bilden, och därefter trycka på F6 för att genomföra en kompilering.

För att paketera applikationen för driftsättning så markera Build i huvudmenyn och klicka på Publish mapservice.



Markera därefter fliken Connection och välj som publiceringsmetod (Publish method) File System och som mål den mapp där de kompilerade filerna skall placeras:



Tryck därefter på Publish för att kompilera och skriva ut alla filer.

# Förbered för driftsättning

De kompilerade filerna finns på olika platser på filsystemet:

(1) c:\[sökväg till rotkatalog]\client\release

(2) c:\[sökväg till rotkatalog]\admin\dist

(3) c:\Install\mapservice.

Skapa en mapp på valfri plats (här c:\Install) och kopiera över filer från ovanstående sökväg så att målmappen följer nedanstående struktur:

(1) c:\Install\client

(2) c:\Install\admin

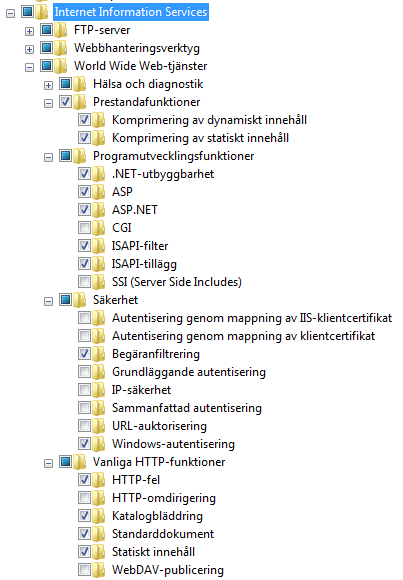
(3) c:\Install\mapservice

Vid varje release så publiceras dessa binärer enligt samma struktur på github för nedladdning.

<https://github.com/hajkmap/hajk/releases>

Innan driftsättning måste vissa förutsättningar förberedas på server.

IIS måste vara installerad med rätt inställningar. Följande bild beskriver rekommenderade inställningar:



# Driftsättning

Skapa en mapp på servern där applikationen skall hostas.

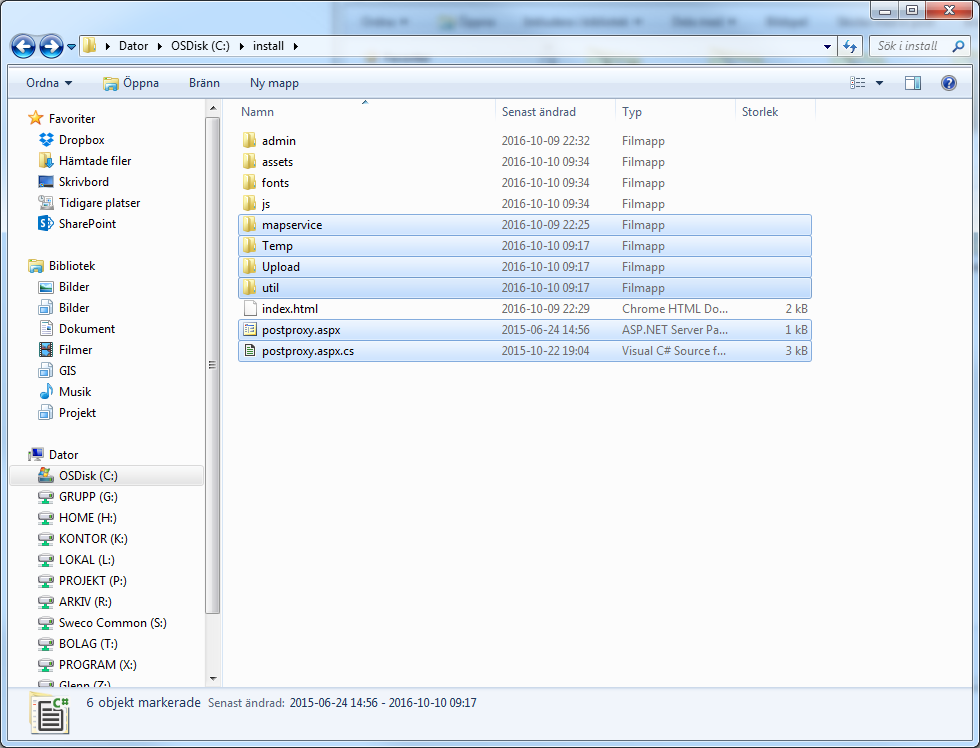
Placera de tre mapparna admin, client och mapservice i mappen. Öppna mappen client och kopiera innehåller, gå tillbaka ett steg och klistra in innehåller direkt i install-mappen. Detta så att filen index.html och mapparna assets, fonts och js är det som utgör grundapplikationen.

Skapa två mappar, en vid namn Upload och en vid namn Temp och ge användaren IIS\_IUSRS (alternativt den avnändare som är registrerad att göra den App\_Pool som siten använder, standard är IIS\_IUSRS) skrivrättigheter. I Temp-mappen kommer temporära filer som användaren skapar att lagras. För Temp-mappen så kan det vara lämpligt att schemalägga veckovisa rensningar. I Upload-mappen lagras filer som användaren laddar upp via administratörsgränssnittet.

Gå till mappen mapservice/App\_Data och ge rättigheter på samma sätt.

Skapa en mapp som heter util, placera däri innehållet som återfinns i Hajk2\proxy\mvc. Detta är en proxy för http-get anrop som kan användas av klienten.

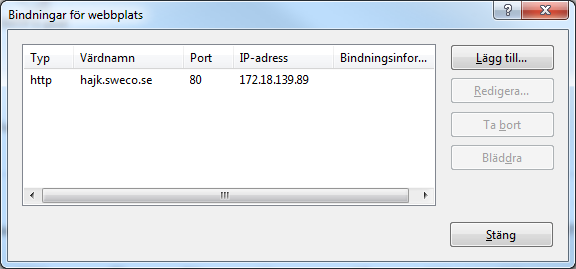
Lägg även till filerna postproxy.aspx och postproxy.aspx.cs som hittas i Hajk2\proxy\aspnet i roten på katalogen. Detta är en proxy för http-post anrop som används av klienten.



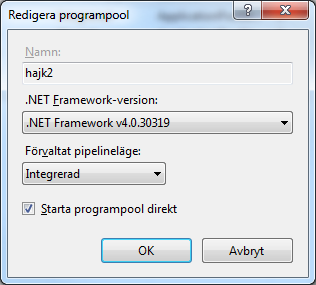
Starta programmet Internet Information Services (IIS)-hanteraren.

Beroende på om servern är dedikerad till att köra endast webbkartan eller om den är till för att hosta flera applikationer så skapa en ny webbplats, alternativt använd Default Web Site. Denna instruktion gäller för skapandet av en ny webbplats i en delad miljö, men tillvägagångssättet är detsamma om Defualt Web Site används.

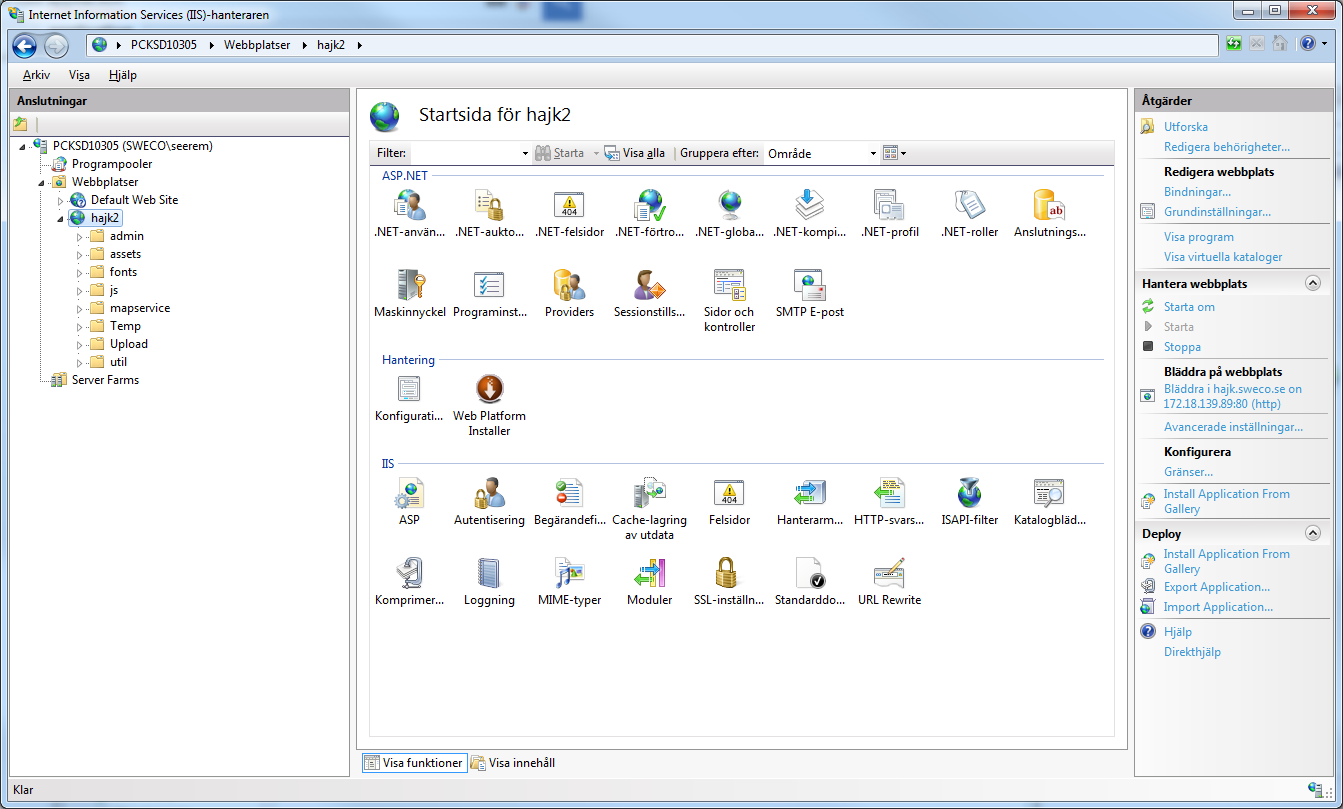
Fyll i bindningsinformation på liknande sätt:



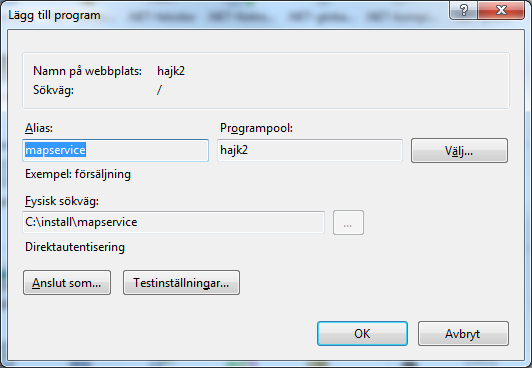
OBS!: Den programpool som skapas måste som lägs köra .NET Framework 4.



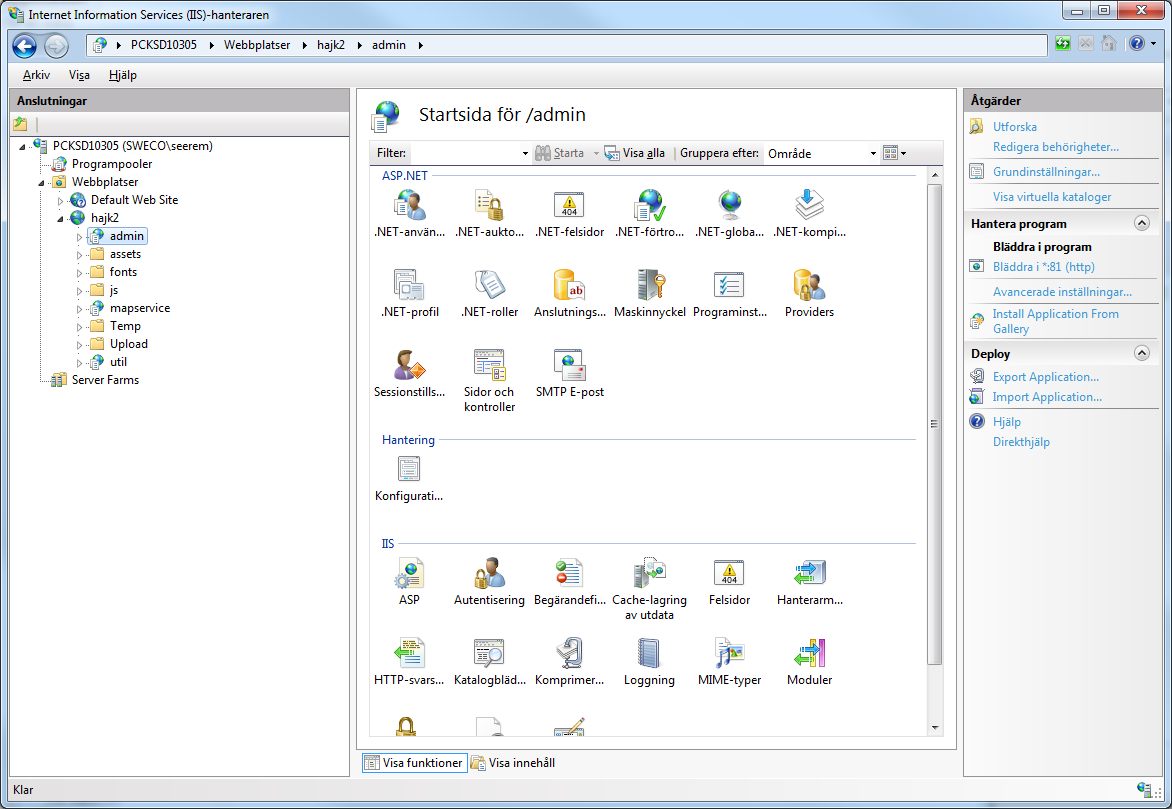
När detta steg är gjort så återfinns katalogstrukturen under vald site. Mapparna admin, mapservice och util behöver registreras som .NET-applikationer.



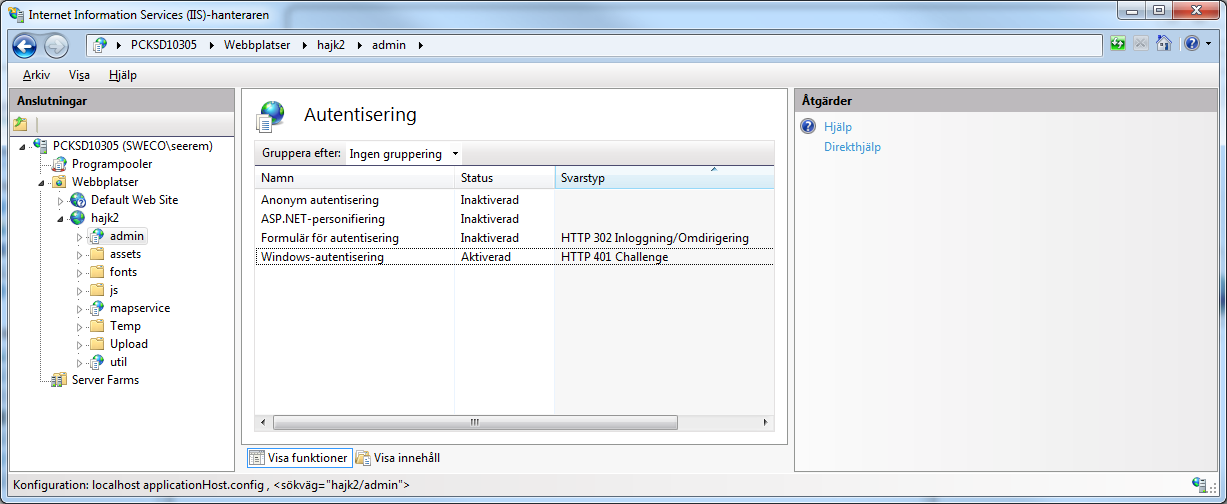
Detta görs genom att högerklicka på respektive mapp och välja ”Konvertera till program”, godkänn inställningarna genom att trycka på OK.



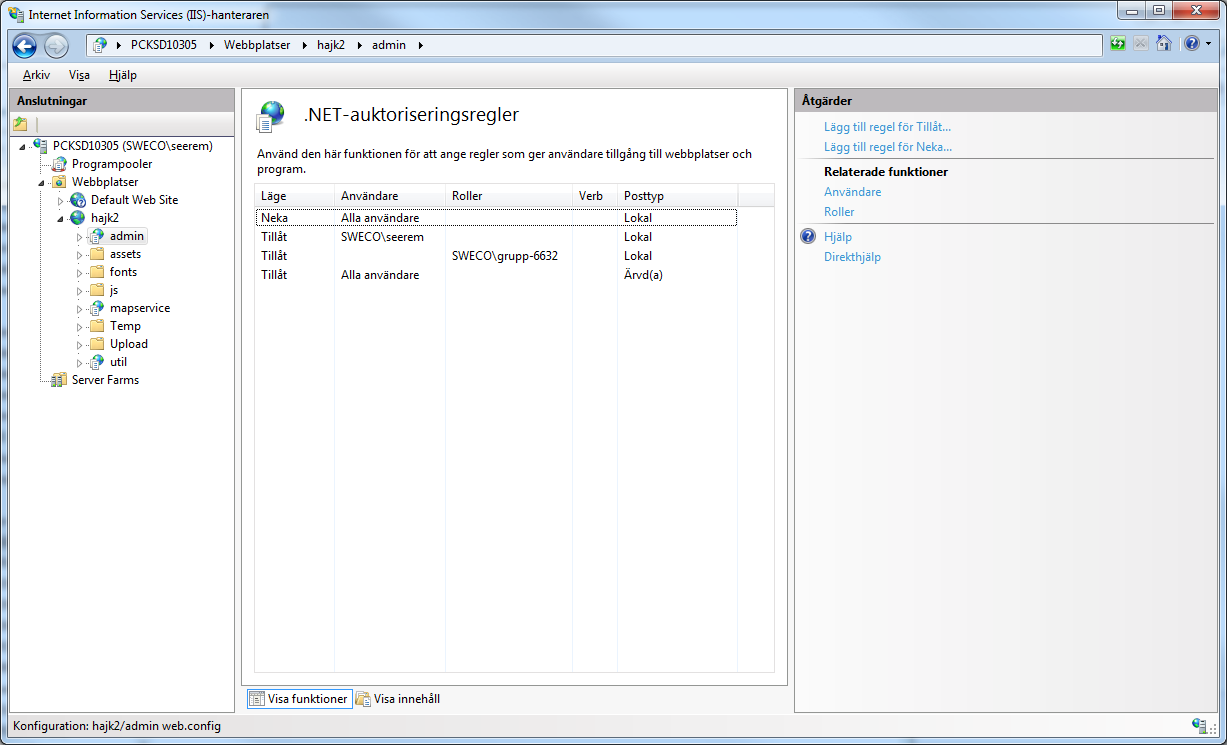
När detta är gjort för de tre applikationerna så bör de se ut på följande sätt:



För admin-applikationen rekommenderas att använda Windowsautentisering. Aktivera detta genom att klicka på IIS / Autentisering och ange följande inställningar:



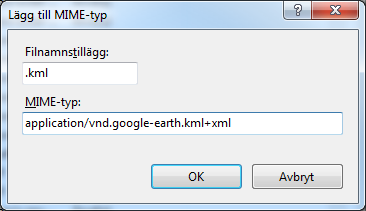
Klicka därefter på tillbaka och öppna ASP.NET \ .NET-auktoriseringsregler och ställ in regler för vilka användare/användargrupper som skall få tillgång till applikationen.



Markera slutligen webbsiten (hajk2) och tryck på MIME-typer.

Tryck på Lägg till..

Ange **.**kml och application/vnd.google-earth.kml+xml.



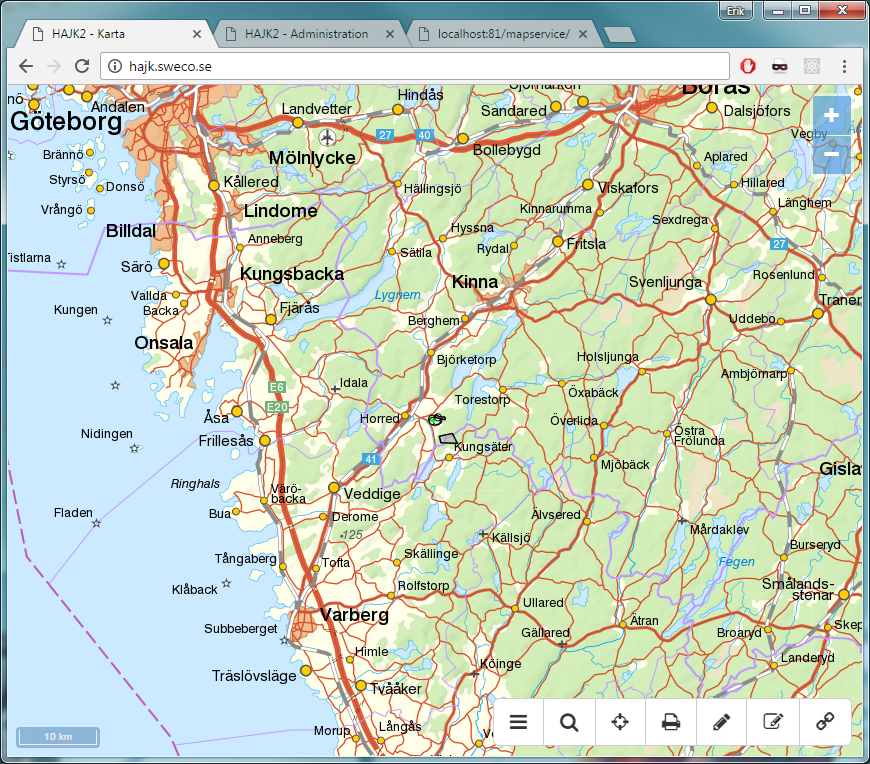
Lägg även till följnade mime-typer:

.json som application/json

.woff som application/x-font-woff

.woff2 som application/x-font-woff2

Nu är applikationerna installerade och skall se ut enligt följande:



# Uppdatera applikation

Ange följande kommando för att uppdatera med den senaste koden från GitHub:

git pull

Kompilera applikationen enligt avsnitten för det i den här dokumentationen.

Färdigkompilerad kod för större releaser finns att tillgå på GitHub: <https://github.com/hajkmap/hajk/releases>

## Uppdatera admin

Byt ut alla filer i mappen admin förutom config.json och Web.config. Dessa är lokala inställningsfiler.

## Uppdatera client

Öppna mappen client i ett fönster och roten på applikationen i ett annat.

Markera i mappen client mapparna, js, assets och fonts, välj kopiera, klistra in i roten på applikationen, välj ersätt alla filer.

Markera filen index.html och välj kopiera, klistra in i roten på applikationen, välj ersätt befintlig fil.

## Uppdatera backend

Byt ut alla filer i mappen backend förutom Web.config. Detta är en lokal inställningsfil.

1. https://git-scm.com/download/win [↑](#footnote-ref-1)
2. https://nodejs.org/en/ [↑](#footnote-ref-2)
3. http://gruntjs.com/getting-started [↑](#footnote-ref-3)
4. https://github.com/jsdoc3/jsdoc [↑](#footnote-ref-4)
5. https://www.visualstudio.com/vs/community/ [↑](#footnote-ref-5)