Hämtande av data ifrån eNova API

Data hämtas via get-anrop till resuser i API:et. Resueser är i första hand olika objectstyper i Enova, t.ex. produkter, kunder, ordrar osv. Ett enkelt anrop kan se ut såhär:

**/api/products**

Anropet ovan kommer att retunera ett JSON med produkter i systemet.

# Filtrerings-Parametrar

För att finjustera vilken data som retuneras från systemet kan användaren skicka med olika parametrar. Olika resuser kan ha olika parametrar, men några är gemensamma. Dessa listas nedan.

### Sort

Sortering på en viss property i resursen. T.ex: **/api/products?sort=name** sorterar produkterna i bokstavsordning. Skriv desc efter för att sortera i omvänd ordning: **/api/products?sort=name desc.** Det går även att sortera på fler properties (thenby), t.ex. **/api/products?sort=name,identifier.**

### Filter

För att välja ut vissa objekt framför andra kan ett filter skickas med. T.ex. **/api/products?filter=price>50** retunerar enbart de produkter som har ett pris över 50. Operatorerna **>, <, >=, <=, =** och **\*** går att använda. **/api/products?filter=name=ost\*** retunerar t.ex. alla produkter vars namn börjar på ost.

((fyll på med mer info om filter när det är spikat))

### Paging

Ofta finns fler objekt i systemet än vad som är lämpligt att retunera i ett anrop. I sådana fall retunerar systemet ett begränsat antal, och ger användare möjlighet att stega vidare för att se resten. För att ändra hur många objekt som skall retuneras skrivs t.ex.: **/api/products?size=30** för att retunera 30st i taget. För att stega vidare och se fler objekt skrivs t.ex. **/api/products?page=1, /api/products?page=2** osv.

### Properties

För att specificera exakt vilken information från resursen som skall retuneras, går det att skriva såhär: **/api/products?properties=identifier,name,price.** De properties som listas retuneras i samma ordning.

### Kombinera parametrar

Samtliga parametrar går att kombinera för en väldefinierad fråga, t.ex. **/api/products?size=5&filter=Enabled=true&sort=name&properties=identifier,name** vilket retunerar identifier och namn på de 5 första produkterna som har värdet Enabled=true, efter sortering på namn.

### Hämta unikt objekt

För att hämta ett specifikt objekt kallat **abc123** skrivs t.ex: **/api/products/abc123.** Här går det även att specificera vilka properties som ska retuneras, se ovan.

# Fördefinierat anrop

Ofta hämtas objekt på samma sätt upprepade gånger. T.ex. kanske en produktdetaljssida alltid visar samma slags information från ett visst antal produkter. Därför finns möjligheten att definiera hur svaret skall se ut i förväg och sedan referera till den konfigurationen via namn. Om det finns en konfiguration som heter **detail** för produktdetaljsidan anropas den såhär: **/api/products?location=defail** . Det går även att lägga till andra parametrar manuellt efter namnet för att skriva över en specifik del av konfigurationen.

((skriv mer om hur det ska konfas))

# Anpassad resurs

Om än större behov av kundanpassning behövs, finns möjligheten att skapa en ny resurs med andra anrop och parametrar. T.ex. kan ett kundprojekt definiera **/api/myproducts** som följer andra regler än basresursen **products.** Se nedan.

# Marknads-Parametrar

API:et stödjer även att hämta objekt med olika språk och valutor. Detta kan göras på ett antal olika sätt.

### Skicka med ett kund-id

Genom att skicka med parametern **/customer=abc123** så kommer svaret presenteras i kunden abc123:s språk och valuta, om de är specificerade på kunden.

### Skicka med admin-information

Specificera vilket admin-konto som skall användas genom att skriva **/admin=apa&pass=banan** och om uppgifterna stämmer används språk och valuta som är inställt på administratören.

### Specificera en fördefinierad marknad

Ge en viss marknad ett namn och referera till dess inställningar genom att skriva **/market=sv**. Detta kan också skrivas innan resursnamnet, t.ex. **/api/sv/products**, **/api/eu/products**, **/api/whatevermarketisconfigured/products**

### Specificera parametrar direkt

Skriv t.ex. **/language=sv&currency=euro** för att få exakt den konfiguration som passar för anropet.

### Prioritering

Samtliga parametrar ovan kan kombineras och skickas med i samma anrop. I så fall är prioriteringen såhär: market < admin < customer < language. Dvs att specificera vilken kund som gäller får högre prioritet än att specificera en marknad, så om marknaden säger att språket skall vara svenska men kunden är inställd på engelska så gäller engelska. Om kunden inte har någon inställning för språk gäller istället inställningen för marknaden. Specifika parametrar som language/currency har alltid företräde.

Anpassning av API

EnovaAPI går att anpassa på ett antal olika sätt. Dels går det att ersätta services för att ändra funktionalitet, dels går det att bygga helt nya resurser, t.ex. **/api/myproducts.**

## Addin-folder

Anpassningar till API:et byggs som en eller flera dller och läggs i en katalog **addin** i roten av projektet (på samma nivå som approot i en publicerad webb). Därifrån kommer dllerna att läsas in vid uppstart. Alla klasser i dllerna vars namn slutar med Mapper, Model eller Service kommer automatiskt att registreras i Autofac. För att själv ta kontroll över registreringen kan man implementera IEnovaApiModule. I en sådan kan man sätta prioritet för att bestämma i vilken ordning autofac-modulerna skall köras, vilket är viktigt eftersom sista registrering av en klass/interface är den som gäller. Basmodulen har en prioritet på 0 och en högre siffra på prioritet i egen modul innebär därför att modulen körs **efter** basmodulen.

# Ersätt/lägg till funktionalitet

För att ersätta en service implementerar man relevant interface, t.ex. IProductService och skriver sina egna metoder. Alla services som kan ersättas ligger i Wipcore.Enova.Api.Interfaces.

Det går även att implementera **IPropertyMapper** för att lägga till mappning för egna properties. T.ex. kan man lägga till en klass som hanterar property ”myproductdetail” och sedan fråga efter denna property genom att lägga till den i ett get-anrop, t.ex. **api/products?properties=myproductdetail.** Notera att det inte går att använda egen property i sortering eller filtrering.

# Egen resurs

För att bygga sin egen resurs skapar man en dll med nya controllers och lägger i addin. Då Api:t bygger på ASP.NET Core, måste den nya controllern skapas i ett ASP.NET 5 templates, för att det ska fungera korrekt.  
Enklast är att sen använda attribut för att specificera routes. T.ex.

[Route("api/[controller]")]

[Route("api/{market}/[controller]")]

public class CustomProductController : Controller