5. Tjenester

Tjenesteoppsett

Tjenestene i matrikkelAPI-et er delt opp basert på type, med noen spesialtilfeller som unntak. Dette betyr at søk der vi ønsker å hente ut adresser ligger i AdresseService, søk der vi ønsker å hente ut matrikkelenheter ligger i MatrikkelenhetService og så videre.

Unntak er spesialtjenester som StoreService for å hente ut objekter, KodelisteService for å hente ut koder, RapportService for bestilling og uthenting av rapporter og NedlastningService for å laste ned store mengder data.

De alle fleste tjenestemetodene returnerer id-er for det man har søkt på. Man skal så bruke StoreService for å hente ut objekter.

Tjeneste klassifisering.

Vi forholder oss til en inndeling av tjenestene i: <<XService og XOppdateringService>> eksempel vis: AdresseService og AdresseOppdateringService.

- **XService** håndterer kall som ønsker å hente ut data fra gitt tjeneste.
- **XOppdateringService** håndterer kall som ønsker og oppdatere eller legge til data.

Typer tjenestemetoder

Tjenestemetodene på de vanlige tjenesteinterfacene kan deles inn i tre typer.

- Søk basert på en sammensatt modell
- Søk for å finne id for en ident
- Inversrelasjonssøk

Beskrivelse av de tre typene følger

Søk basert på sammensatt modell

Noen søk kan basere seg på input fra mange forskjellige deler av domenet. Det er for disse laget egne tjenestemoder som tar inn søkemodeller som parameter. Eksempel på dette er **MatrikkelenhetService** sin metode **findMatrikkelenheter**. Denne metoden tar inn en klasse med navn **MatrikkelenhetsokModel** som parameter og denne klassen gir oss mulighet til å

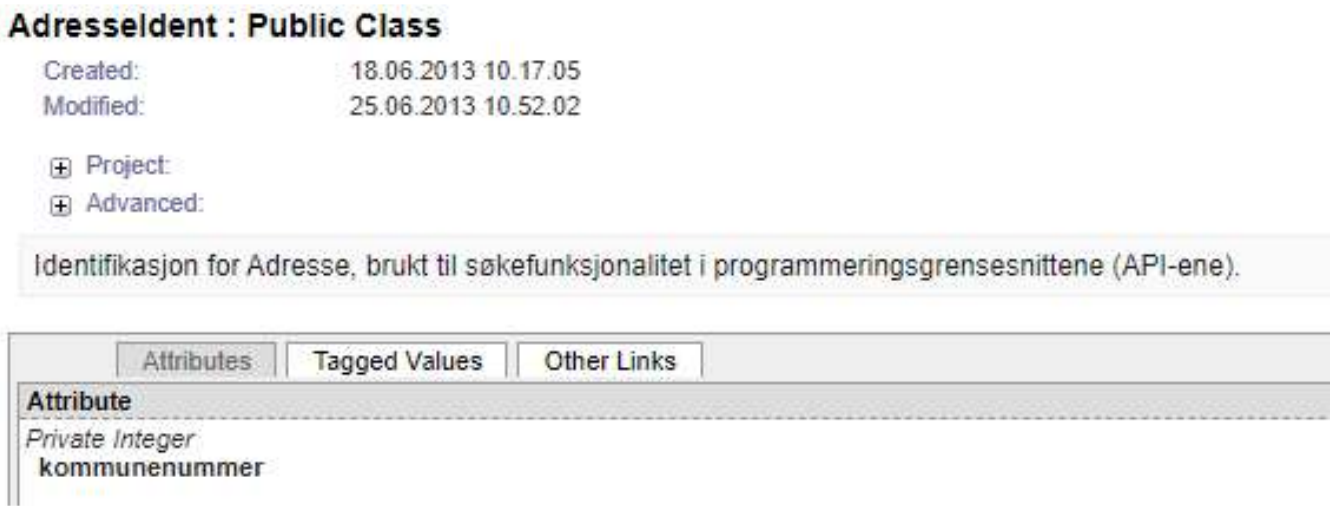
spesifisere alt fra matrikkelenhetfelter som gårdsnummer og bruksnummer, til koblingsfelter som krav om jordskifte og over til koblede objekter slik at vi kan angi adressenavn og husnummer for en adresse tilknyttet matrikkelenheten, eller etternavnet til en person som har eierforhold på matrikkelenheten. Siden tjenesten ligger på **MatrikkelenhetService** vet vi at det er **MatrikkelenhetId**-er som blir returnert fra søket.

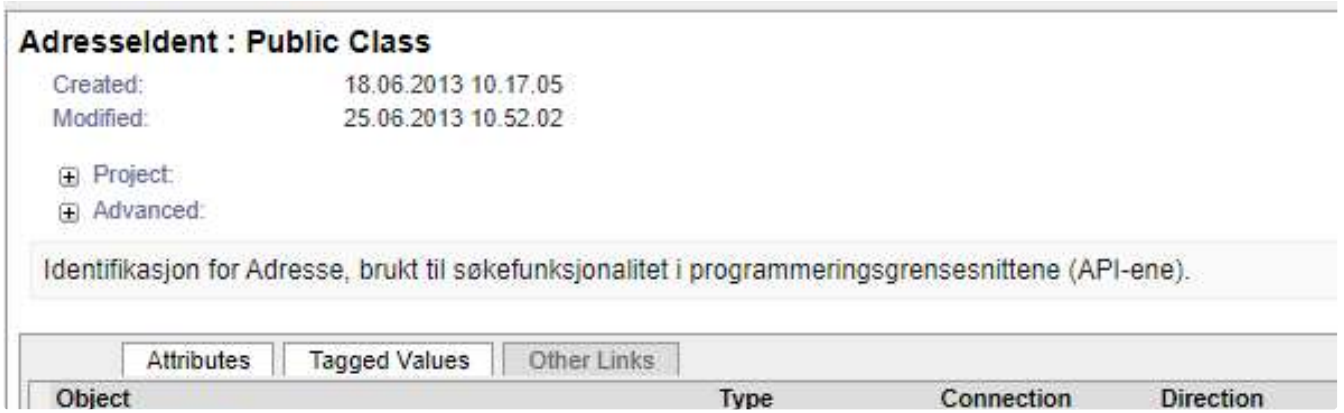
Det er laget slike sammensatte søk for adresser, bygg, matrikkelenheter og person i API-et.

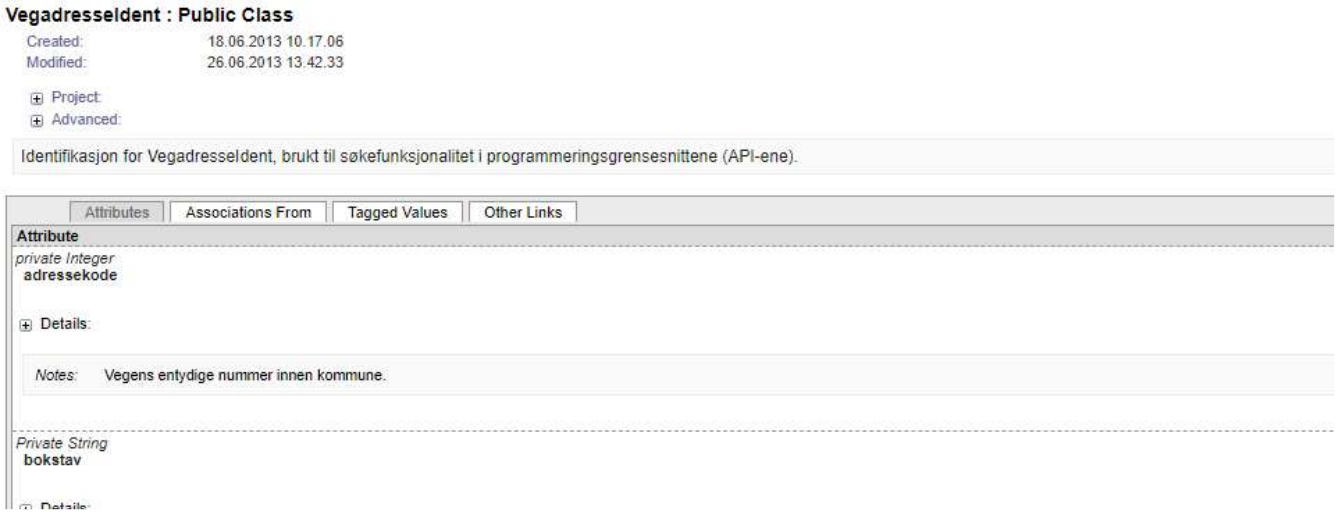
Logiske identifikatorer

Under følger en tabell som viser hva som benyttes som logiske identifikatorer for et objekt. Dette kan brukes dersom man for eksempel skal finne id-en til et objekt for så å finne den komplette boblen via StoreService (se kapitlet om StoreService under). Utlistingen under er ment for å gi en rask oversikt over hvilke felt som benyttes for å bygge opp de logiske identifikatorene. For en mer komplett oversikt over hva de forskjellige feltene er henvises det til implementasjonsmodellen hvor man kan navigere seg inn på hver enkelt identifikator. Ofte er hoved-identifikatoren en abstrakt klasse, da finner man implementasjons-klassen ved å velge Other Links.

Eksempel for AdresseIdent (som er av typen Vegadresse eller Matrikkeladresse):




Adresseident : Public Class
Created: 18.06.2013 10.17.05
Modified: 25.06.2013 10.52.02
Project:
Advanced:
Identifikasjon for Adresse, brukt til søkefunksjonalitet i programmeringsgrensesnittene (API-ene).
Attributes
Tagged Values
Other Links
Object
Type
Connection
Direction



Requesten blir da som følger (merk at vi må angi hva slags adresseIdent vi benytter i dette tilfellet) i f eks findAdresseIdForIdent (denne hadde ingen bokstav):

```
<adr:adresseIdent
  xsi:type="adr1:VegadresseIdent"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <adr1:kommuneIdent>
    <kom:kommunenummer>3031</kom:kommunenummer>
  </adr1:kommuneIdent>
  <adr1:adressekode>14200</adr1:adressekode>
  <adr1:nummer>109</adr1:nummer>
</adr:adresseIdent>

```

Logisk identifikator	Hvilke felt brukes for å bygge opp den logiske identifikatoren
AdresseIdent	Identifikatoren inneholder kommunenummer. Resterende opplysninger som utgjør identifikatoren avhenger om adressen er av typen Vegadresse eller Matrikkeladresse. Oversikt over opplysninger finnes i VegadresseIdent og MatrikkeladresseIdent
BrukerIdent	Brukernavnet
ForretningIdent	Løpenummeret til forretningen
KommuneIdent	Kommunenummer
MatrikkeladresseIdent	Gardsnummer, bruksnummer, festenummer samt undernummer
MatrikkelenhetIdent	KommuneIdent, gardsnummer, bruksnummer, festenummer og seksjonsnummer
PersonIdent	Nummer, fødselsnummer eller organisasjonsnummer avhengig av om personen er av typen AnnenPerson, FysiskPerson eller JuridiskPerson. Merk at alle disse ligger i feltet nummer på Person-objektet.
SefrakIdent	Kommunenummer, registreringskretsnummer og et husløpenummer
VegadresseIdent	Adressekode, nummer og bokstav

Søk for å finne id for en ident

Vi kan ikke forvente at brukere av API-et vet hva våre unike id-er er for de forskjellige objektene våre så det er laget tjenester som tar inn en logisk identifikator (se kapitlet over for hvordan de bygges opp) for et objekt og returnerer id-objektet som representerer denne i systemet. Disse tjenestene finnes for alle boblene som har logiske identifikatorer, men ikke for absolutt alle bobler. De boblene som ikke har noen logisk identifikator (som for eksempel en teig, en flate for en matrikkelenhet) vil ikke ha tjenester for å oversette fra logiske identifikatorer til id-er.

Det er laget både entalls og flertalls-versjoner av tjenestemetodene slik at man kan hente ut id-er for mange identer om gangen hvis man ønsker det.

Klasser som har disse tjenestene er eksempelvis Adresse, Veg, Bygg, Forretning, Kommune og Matrikkelenhet.

Inversrelasjonssøk

I modellen kan man alltid navigere via id-koblingene som er laget, men det kan også være tilfeller der man ønsker å gå motsatt vei. Det er i disse tilfellene laget tjenester for å søke opp dette. Eksempelvis kan man i **AdresseService** søke opp alle adresser for en eller flere veger eller alle adresser for en eller flere bygg. Disse koblingene finnes ikke direkte i modellen nødvendigvis men det er mulig å navigere i modellen for å finne objektene.

Spesielle tjenester

StoreService

Denne tjenesten tilbyr metoder for å hente ut en eller flere bobler basert på deres id. Denne er primærmåten å navigere gjennom boblemodellen og for å hente ut boblene.

RapportService

Matrikkelen tilbyr en del predefinerte rapporter og denne tjenesten brukes for å bestille og hente disse. Bruksmønsteret er at man benytter en metode for å bestille en rapport. Man får da tilbake et rapport-objekt som peker til et Jobb-objekt som man kan hente ut for å få status for bestillingen. Når rapporten er ferdig kan man hente denne ut over HTTPS som en filnedlasting.

Eksempel

Følgende eksempel viser hvordan vi kan bestille og hente ut en rapport. Bruker rapporten "Samlet rapport for matrikkelenhet" som eksempel.

```
1  //Finner først matrikkelenheten vi vil ha rapport for
2  MatrikkelenhetService matrikkelenhetService;
3  MatrikkelenhetId matrikkelenhetId = matrikkelenhetService.findMatrikkelenhetIdForIdent(new MatrikkelenhetIdent(...));
4
5  //Bestiller så rapporten
6  RapportService rapportService;
7  OfflineMatrikkelRapport r = rapportService.createSamletRapportForMatrikkelenhet(matrikkelenhetId, ExportTypeId.PDF, true)
8
9  //Henter rapporten ut og finner URL for filen.
10 MatrikkelRapport rapport = rapportService.hentRapport(r.getJobbId()); //Dette kallet feiler hvis rapporten ikke er klar.
11
12 String urlForRapport = rapport.getURL(); //Kan så bruke denne for å hente rapporten over https
```

KodelisteService

API-et tilbyr en tjeneste for å hente ut alt av koder i systemet. Dette kan da brukes for å cache opp kodeverdier i lokal klient. Da det i utgangspunktet kan komme til nye kodeverdier når som helst bør man i klient benytte seg av denne tjenesten ofte for å være sikker på at man ikke ender opp med å få bobler fra StoreService som refererer til kodeverdier man ikke vet om.

Metoden getKodelister på tjenesten tar inn et tidspunkt, men dette bør ikke benyttes med mindre man vet at man ikke ønsker kodelister på nåværende tidspunkt. Dette krever spesielle rettigheter så man bør nok benytte seg av "nåtid" i de fleste tilfeller.

NedlastningService

Det kan være tilfeller der man ønsker å hente ut store datamengder. NedlastningService tilbyr to metoder for å kunne gjøre dette. De to metodene har samme signatur men forskjellige returtyper. Den ene metoden gir tilbake id-er og den andre gir oss boblene. Det er opp til brukeren å bestemme hvilke av disse som er mest hensiktsmessige. Hvis man allerede har en lokal kopi av matrikkelen og vil sjekke om man har alle id-er vil det være unødvendig å hente ut alle objekter. Under følger eksempel på bruk av tjenesten

Eksempel

```
1  NedlastningService service;
2
3  String filter = "{kommunefilter: [\"1201\"]}";
4  List<VegId> alleIds = new ArrayList<>();
5  List<VegId> idsEtterId;
6  VegId sisteId = null;
7  do {
8      idsEtterId = service.findIdsEtterId(sisteId, Veg.class, filter, 10000);
9      alleIds.addAll(idsEtterId);
10     sisteId = alleIds.get(alleIds.size() - 1);
11 } while(idsEtterId.size() > 0);
12
13 return alleIds;
```

Eksemplet henter ut alle veger i kommune med kommunenummer "1201". Se beskrivelsen av tjenesten for mer informasjon rundt de forskjellige parametrene.

Tjenesten kan man da bruke for å hente ut data for en eller flere kommuner (eller hele landet) og man kan så koble dette sammen med endringsloggstjenester for å bygge opp en lokal kopi av matrikkeldata.

EndringsloggService

EndringsloggService kan brukes for å hente en instans av Endringer, som igjen inneholder en liste av Endring. En Endring representerer en Nyoppretting, Oppdatering, Sletting eller en Typeendring for ett bobleobjekt. Legg merke til at en endring inneholder ikke hva som eventuelt har blitt endret i bobleobjektet, om man oppgir ReturnerBobler.Alltid som parameter til endringsloggstjenesten vil man få tilbake boblen i nåværende tillstand(dvs. ikke tilstanden boblen hadde da endringen ble utført).

Brukstilfelle for denne tjenesten kan være å holde ett system synkronisert med matrikkelen. For å etablere en lokal kopi vil man bruke endringsloggtjenesten sammen med nedlastningservice beskrevet over. Man kan da hente ut siste endring fra endringsloggservice først, før man så benytter nedlastningservice for å hente ut alle objekter for typene man vil ha i lokal kopi. Etter at man har fått ut alle objekter kan man lese endringer som har foregått i løpet av perioden man lastet ned data.

Eksempel som går igjennom alle endringer

```
EndringsloggService endringsloggService;
StoreService storeService;

Map<BubbleId<?>, BubbleObject> boblerForHenting = new HashMap<>();
Set<BubbleId<?>> leggesTil = new LinkedHashSet<>();
Set<BubbleId<?>> oppdateres = new LinkedHashSet<>();
Set<BubbleId<?>> slettes = new LinkedHashSet<>();

String filter = "{kommunefilter: [\"1201\"]}";
MatrikkelEndringId forsteEndringsId = null; // bruk f.eks. id fra siste endring hentet med nedlastingstjenesten
Endringer<MatrikkelEndring<?, ?>> endringer;
do {
   endringer = endringsloggService.findEndringer(
         forsteEndringsId,             // returner endringsobjekter fra og med oppgitt endringsID.
         MatrikkelBubbleObject.class,  // returner endringer for alle typer i matrikkelen
         filter,                       // begrens til endringer for en gitt kommune
         ReturnerBobler.Aldri,         // ikke returner bobleobjektene endringen gjelder for(dvs. kun endringsobjektet)
         10000);                       // returner maksimalt 10000 endring

   // Samle opp id for alle bobleobjekter som er lagt til eller oppdatert, slik at de kan hentes i ett store service kall
   for (MatrikkelEndring<?, ?> endring : endringer.getEndringList()) {
      switch (endring.getEndringstype()) {
         case Nyoppretting:
            leggesTil.add(endring.getEndretBubbleId());
            boblerForHenting.put(endring.getEndretBubbleId(), null);
            break;
         case Typeendring:
         case Oppdatering:
            oppdateres.add(endring.getEndretBubbleId());
            boblerForHenting.put(endring.getEndretBubbleId(), null);
            break;
         case Sletting:
            slettes.add(endring.getEndretBubbleId());
            leggesTil.remove(endring.getEndretBubbleId());
```

```
            oppdateres.remove(endring.getEndretBubbleId());
            break;
        }
    }

    // hent alle nye og oppdaterte bobler med store service
    for (BubbleObject bubbleObject : storeService.getObjectsIgnoreMissing(boblerForHenting.keySet())) {
        boblerForHenting.put(bubbleObject.getId(), bubbleObject);
    }
} while (!endringer.isAlleEndringerFunnet());

for (BubbleId<?> bubbleId : leggesTil) {
    BubbleObject bubbleObject = boblerForHenting.get(bubbleId);
    // legg til bubbleObject til system som skal synkroniseres
}

for (BubbleId<?> bubbleId : oppdateres) {
    BubbleObject bubbleObject = boblerForHenting.get(bubbleId);
    // oppdater bubbleObject i system som skal synkroniseres
}

for (BubbleId<?> bubbleId : slettes) {
    // slett bubbleId fra system som skal synkroniseres
}
```

Feilhåndtering

Dersom det er en feil som faktisk prosesseres på tjeneren skal dette komme tilbake i responsen som en relativt fornuftig feilmelding. Noen eksempler dersom man har søkt på noe som ikke finnes:

```
AdresseService:
<faultcode>S:Server</faultcode>
<faultstring>Fant ingen forekomst ved søk på krets med kretsnr:359, kretstypeKode.id=4097 og kommune.id=KommuneId{value=100000301, snapshotVersion=SnapshotVersion{timestamp=CURRE
    <detail>
        <ns3:ServiceException ...>
            <ns3:stackTraceText>no.statkart.skif.exception.FinderException: Fant ingen forekomst ved søk på krets med kretsnr:359, kretstypeKode.id=4097 og kommune.id=KommuneId{va

StoreService:
```

```
<faultcode>S:Server</faultcode>
<faultstring>[KommuneId{value=300712, snapshotVersion=SnapshotVersion{timestamp=CURRENT}}]</faultstring>
    <detail>
        <ns32:ServiceException ...>
            <ns32:category>:ServiceException:ApplicationException:FinderException:</ns32:category>
            <ns32:stackTraceText>no.statkart.skif.exception.ObjectNotFoundException: [KommuneId{value=300712, snapshotVersion=SnapshotVersion{timestamp=CURRENT}}]


```

En kjent feil er dersom man skulle fått en fornuftig feilmelding (ala det over), men man samtidig har ugyldig systemVersion får man ikke en gyldig feilmelding tilbake. Så om man får en generell feilmelding på dette formatet under kan det være lurt å sjekke at matrikkelContext-objektet er riktig satt oppt:

```
      <S:Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>S:Server</faultcode>
         <faultstring>Error mapping from no.statkart.skif.exception.FinderException to java.lang.Throwable</faultstring>
      </S:Fault>
```