Introduktion til Regin XML og Web services

KULTUR STYRELSEN

Center for Bibliotek, Medier og Digitalisering

17. juli 2012

Indholdsfortegnelse

1 Indledning

Målgruppe

Dette notat henvender sig primært til teknikere og it-leverandører, som skal implementere løsninger, der skal udveksle data med Regin, Kunstindeks Danmark og Museernes Samlinger. Der forudsættes derfor bekendtskab med relevante tekniske termer, men ikke en dyb indsigt i museernes praksis omkring registrering.

Kulturstyrelsens tjenester

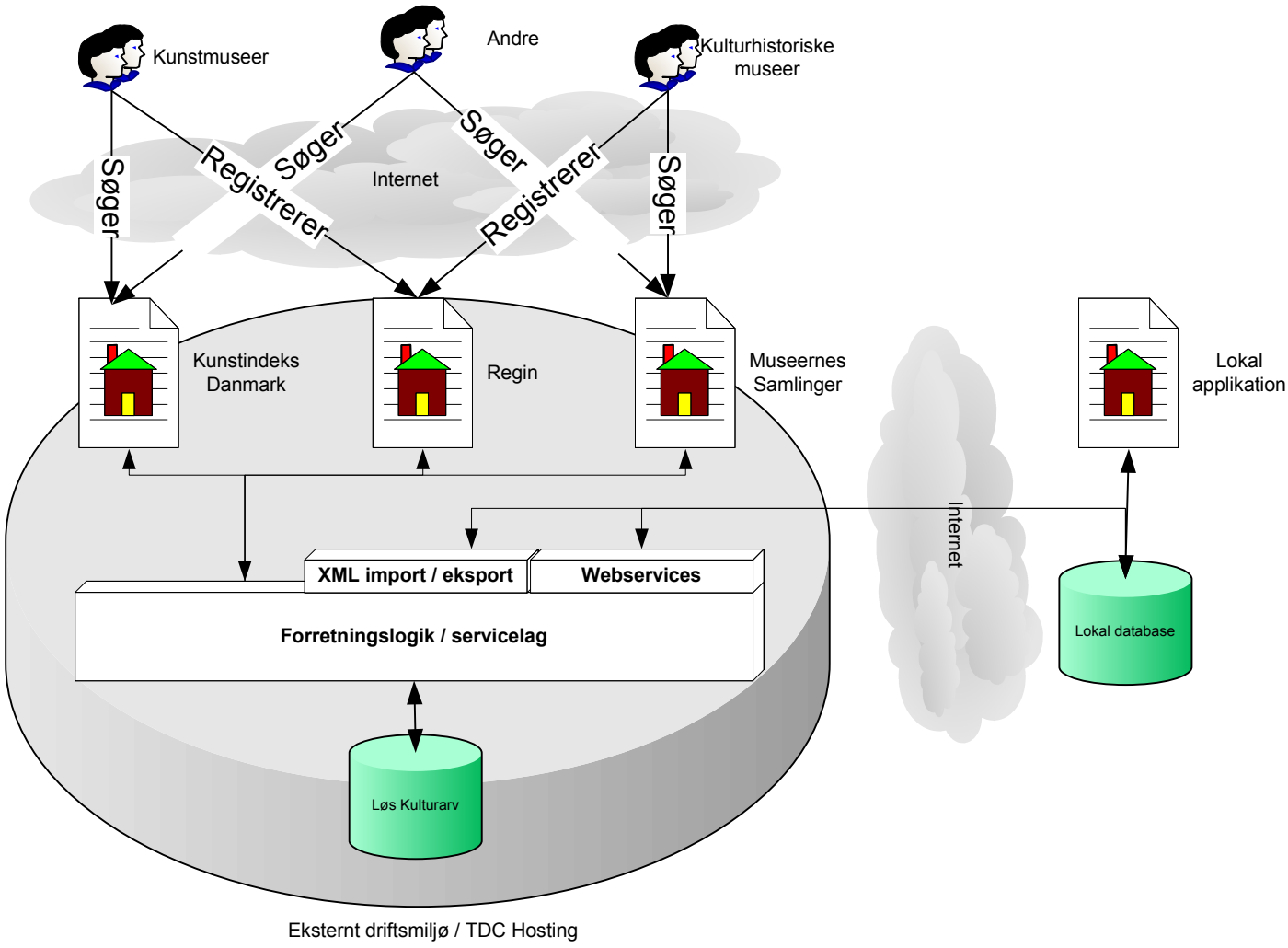
Kulturstyrelsen udvikler og driver de to centralregistre *Kunstindeks Danmark*[1] og *Museernes Samlinger*,[2] som samler information om kunstværker hhv. kulturhistoriske genstande og sager i danske statslige og statsanerkendte museers samlinger. Museerne kan indberette til og trække på oplysninger fra disse registre:

1. ved at benytte *Regin*, en webapplikation, som samtidig udgør en grundlæggende *collections management* løsning for museet
2. ved at benytte en XML import/eksport funktion baseret på Dansk Museums Dokumentations Standard
3. ved at benytte Regin web services

Regin arkitekturen

Diagrammet på næste side illustrerer arkitekturen. Museale og andre brugere kan anvende de klienter og tjenester, som Kulturstyrelsen har udviklet. Lokale applikationer og databaser kan udveksle data med (og søge i) centralregistrene dynamisk via web services eller i batch via XML.

[1] http://www.kulturarv.dk/kid
[2] http://www.kulturarv.dk/mussam



Figur 1: Regin arkitekturen

2 Dansk Museums Dokumentations Standard

I centralregistrene registreres kunst- og kulturhistoriske metadata om blandt andet museumsgenstande og kunstværker. Til grund for registreringen ligger *Dansk Museums Dokumentations Standard* (DMDS), der er beskrevet med et XML schema, kaldet *DMDS schema*. På Kulturstyrelsens hjemmeside[3] kan man hente en række ressourcer, som på forskellige måder redegør for standarden:

- DMDS Schema
- Baggrunden for standarden
- En ideologisk/normativ begrundelse for kulturhistoriske museers registreringspraksis
- Regins brugermanualer, som detaljeret redegør for det praktiske arbejde med standarden således, som det udmønter sig i Regin

[3] http://www.kulturarv.dk/forvaltning/museumsdrift/vejledninger/registrering

3 Introduktion til Regin XML

Dokumentationen for Regins XML funktioner udgøres af:

- Dette dokument (*Introduktion til Regin XML og web services*)
- DMDS Schema, som anvendes ved import til og eksport fra Regin
- Kapitel 10 i Regin brugervejledningen, som blandt andet beskriver, hvordan man i praksis importerer og eksporterer XML i Regin

XML import og DiffTool

Til brug for import af XML i Regin, anvendes det såkaldte DiffTool, som kan hentes fra Kulturstyrelsens hjemmeside. DiffTool løser i virkeligheden flere opgaver:

1. Validerer XML filer og sikrer, at de respekterer DMDS Schema
2. Sammenligner to XML filer og beregner forskellen imellem dem (heraf navnet på værktøjet)
3. Uploader den resulterende differentiale XML fil og eventuelle vedhæftede filer til Regin serveren og sætter import jobbet i import køen

Hvorfor skal der sammenlignes XML filer og beregnes forskelle? Idéen er denne: De museer, som anvender XML til at indberette til centralregistrene med jævne mellemrum, foretager en eksport fra deres lokale *collections management system*. I princippet kunne man godt indsende hele registreringen hver gang, men det er besværligt at transmittere så mange data over nettet, og det belaster serveren unødigt med at behandle data, som allerede er indberettet én gang. Difftool har derfor som opgave at flytte de nødvendige beregninger til klientsiden, således at kun nye (eller slettede) poster overføres til centralregisteret. Det sker ved at lade Difftool sammenligne XML repræsentationen af museets registreringer nu og ved sidste indberetning. På denne måde aflastes serveren, og museet slipper for selv at skulle føre regnskab med hvilke poster, der allerede er indberettet.

4 Introduktion til Regin Web services

Dokumentationen for Regins web service lag udgøres af:

- Dette dokument (*Introduktion til Regin XML og web services*)
- Listen over tilgængelige services og metoder, som indeholder signaturer og WSDL beskrivelser for samtlige web services.[4]
- Regins datamodel, som beskriver objekter, felter og datatyper i den underliggende database.

[4] http://www.kulturarv.dk/regin/services/listServices

- Referenceeksempler på brug af web services. Eksemplerne er udarbejdet i Java.

De sidste to dele af dokumentationen kan rekvireres hos Kulturstyrelsen; se kontaktinformation sidst i notatet.

Regin web services benyttes lettest ved at anvende WSDL beskrivelserne sammen med et udviklingsmiljø, som kan generere lokale stubs til kald af tjenesterne. Den væsentligste udfordring er at lære den ret komplekse og til tider ideomatiske datamodel at kende. I det følgende gennemgås derfor kursorisk hovedprincipperne.

5 Principper og eksempler

I DMDS repræsenteres al museal viden, som en eller flere *grafer*. Knuderne i graferne udgøres af objekter, opdelt i 29 klasser; se bilag 1.

De 29 klasser kan endvidere placeres i (delvis sammenfaldne) grupper:

1. Forældreløse klasser (se herunder): *Sag, Globalkunstner*
2. Entydigt tilknyttede klasser (klasser, som relaterer n:1 til andre): *Kunstner, Skib, Genstand, Kommunikation, FotoFilm, Magnetmedium, Rapport, Storformat, Arkivfond, Forsikring, Konservering, Arkivindhold, Position*
3. Løst tilknyttede klasser (klasser, som relaterer n:m til andre): *Værk, Genstand, FotoFilm, Magnetmedium, Rapport, Arkivfond, Aktør, Litteratur, Udstilling, Deponering*
4. Særlige klasser (klasser, som har "kvalificerede" n:m relationer til andre; se senere): *Aktør, Opbevaring, Stedregistrant*

Nogle klasser optræder altså i flere grupper. For eksempel relaterer en Genstand n:1 til Sag og Kunstner, men er løst knyttet (n:m) til Skib.

Kunstnere er et helt kapitel for sig. Normalt er det slet ikke nødvendigt at oprette nye kunstnere, da langt de fleste kunstnere findes i Kunstindeks Danmark i forvejen. *Under alle omstændigheder skal man sikre, at man ikke kommer til at oprette en allerede eksisterende kunstner.* Mere om dette i afsnit 5.6.

Selv om DMDS strengt taget beskriver en graf, har de fleste objekter i praksis en entydig forfader – organiseringen af objekter er m.a.o. *normalt* hierarkisk, især i de "øverste" dele af hierarkiet. Man kan derfor også tænke på datastrukturen som træer, hvor trækronerne er vokset delvis sammen. De mest normale undtagelser er opbevaringer og aktører, som knyttes til f.eks. flere genstande.

5.1 Forældreløse klasser

Forældreløse objekter (Sag, Globalkunstner) er de simpleste at oprette i XML:

```
<Export>
<Sag extId="Sag:82526">
<declareSagStatus>
<entryId>2</entryId>
<kode>ITV</kode>
<tekst>afventer initiativ</tekst>
</declareSagStatus>
<sagStatusId>2</sagStatusId>
<nummer>0012</nummer>
<betegnelse>Juleskikke i det moderne Danmark omkring år
2000</betegnelse>
<declarePeriode>
<entryId>74</entryId>
<kode>990</kode>
<tekst>1990-1999</tekst>
<startAar>1990</startAar>
<slutAar>1999</slutAar>
<beskrivelse>1990-1999 (1990 e.Kr. - 1999 e.Kr.)</beskrivelse>
<parentPeriodeId>69</parentPeriodeId>
</declarePeriode>
<periodeId>74</periodeId>
<sagstype>undersøgelsessag</sagstype>
<ansvarligExtId>MuseumAktoer:16</ansvarligExtId>
<internNote />
<eksternNote>Sagen opstået i forbindelse med forespørgsel fra
Danmarks Radio om alderen på den julepyntning af haver og huse
med lyskæder, som i disse år bliver mere og mere alminde-
lig.&#13;&#10;EW&#13;&#10;12. december 2002</eksternNote>
</Sag>
…
</Export>
```

Figur 2: Repræsentation af en Sag i XML

Nogle bemærkninger:

Det er XML; husk at *escape* særlige tegn i tekststrenge (f.eks. CR/LF i ekstern note i eksemplet).

Nogle af objektets attributter (f.eks. sagens status ovenfor) bliver repræsenteret ved flere XML elementer, i et særligt "declare" element navngivet `<declareSagStatus>` el.lign. Denne struktur repræsenterer værdier, som er slået op i en *stamtabel* – en hjælpetabel, som styrer indholdet af felter med enumererede komplekse typer. Disse hjælpetabeller indeholder typisk et id, en kode og en tekstbeskrivelse af værdien; declare elementet nævner alle tre atributter ved værdien for at gøre det let at forstå betydningen af værdien i.f.m. eksport. Elementet er ikke obligatorisk; normalt anvendes kun id'et i forbindelse med *import* i Regin, mens det ofte kun er teksten,

der vises i Regin. I eksemplet ovenfor er sagens status ”afventer initiativ” – svarende til koden ”ITV” og id 2.

Alle Regin objekter i en DMDS XML fil er forsynet med en attribut ved navn *extId*. ExtId’er har følgende kvaliteter:

- De identificerer objektet entydigt i XML filen – og hvis man laver sin egen XML fil til import i Regin, skal man forsyne alle objekter med et unikt extId.
- Ikke alle objekter i Regin har et extId – men hvis man eksporterer et objekt uden extId som XML, tildeler Regin dem et.
- Hvis man importerer et objekt med et bestemt extId og derefter eksporterer det, har det stadig det samme extId. Tilsvarende kan man via XML adressere og overskrive eksisterende objekter i Regin ved at referere til deres extId.

Uanset om man importerer eller eksporterer DMDS XML, anvender man *altid* `<export>` elementet. `<import>` elementet beregnes af DiffTool og udgør den differentiale XML ved sammenligning af to XML (eksport) filer. Man skal derfor aldrig selv benytte `<import>` elementet, men altid placere sin XML i et `<export>` element.

Web service eksempel
Samme sag, som ovenfor er beskrevet som XML, kunne oprettes via Regin web services således.

```
SagServiceStub.Sag sag = new SagServiceStub.Sag();
sag.setBetegnelse("Juleskikke i det moderne Danmark omkring år
          2000");
sag.setNummer("0012");
sag.setPeriodeId(74);
sag.setSagStatusId(2);
sag.setSagsType("undersøgelsessag");
sag.setAnsvarligId("MuseumAktoer:16");

// Etc.

SagServiceStub.Create createParam = new SagServi-
ceStub.Create();
createParam.setToken(currentToken);
createParam.setWsform(sag);

SagServiceStub sagService = new SagServiceStub();
SagServiceStub.WSStatus sagStatus = sagService.
          create(createParam).get_return();
```

```
System.out.println("Status creating sag: " +
          sagStatus.getFaultCode() + ", " +
          sagStatus.getFaultMessage() + ", generatedSagId:" +
          sagStatus.getGeneratedId());
```

Figur 3: Oprettelse af en sag via web services

(Dette og følgende eksempler er udarbejdet i Java.) `SagServiceStub` genereres af udviklingsmiljøet ud fra WSDL definitioner. Eksemplet viser ikke den forudgående login operation, som har resulteret i `currentToken`, et login token, som autoriserer operationen.

Fejlkoder, som Regin web services returnerer, er beskrevet i bilag 3.

5.2 Stamtabeller

Mange felter i objekterne er af enumererede typer, hvis værdier styres af såkaldte stamtabeller. Stamtabelværdier og deres koder, som man skal kende for at læse og skrive i de enkelte objekter, kan slås op dynamisk:

```
/**
 * @param currentToken
 * Metoden læser indholdet af Periode stamtabellen
 */
private static void getHelpTable(String currentToken) throws
          IOException, MalformedURLException {
URL periodeUrl = new URL(
          "http://www.kulturarv.dk/regin/stamtabeller/
          h_Periode.xml?token=" + currentToken);
BufferedReader br = new BufferedReader(
          new InputStreamReader(periodeUrl.openStream()));
String inputLine;
while ((inputLine = br.readLine()) != null) {
  System.out.println(inputLine);
}
br.close();
}
```

Figur 4: Periode stamtabellen læses efter login

Opslaget sker på den viste URL:
http://www.kulturarv.dk/regin/stamtabeller/
$NAVN.xml, hvor $NAVN erstattes af navnet på tabellen. Navnene på tabellerne findes i bilag 2 til denne vejledning.

Ønsker man ikke at benytte web services til at slå disse værdier op, kan de alternativt hentes fra Kulturstyrelsens hjemmeside[5] i MySQL SQL format.

Generelt anbefales det at cache stamtabeller lokalt af hensyn til performance, da de sjældent ændres.

[5] http://www.kulturarv.dk/forvaltning/museumsdrift/vejledninger/registrering; se "Regin Stamtabeller"

5.3 Entydigt tilknyttede klasser

Entydigt tilknyttede klasser håndteres også ret enkelt i både XML og web services.

XML eksempel

```
<Genstand parentExtId="Sag:82526" extId="Genstand:12345" pa-
rentType="Sag">
<nummer>0012X0001</nummer>
<betegnelse>telefon nr 1</betegnelse>
<declareGenstandErhvervelse>
<entryId>9</entryId>
<kode>GAV</kode>
<tekst>gave</tekst>
</declareGenstandErhvervelse>
<genstandErhvervelseId>9</genstandErhvervelseId>
…
</Genstand>
```

Figur 5: Et entydigt tilknyttet objekt i XML; en genstand under en sag

En række underelementer ved genstanden er udeladt i eksemplet ovenfor, men bemærk, at <Genstand> elementet har to attributter, som <Sag> ikke har:

- parentType, som her er sat til "Sag" for at markere, at genstanden er knyttet til en sag – den kunne i stedet have været knyttet til en kunstner.
- parentExtId, som er extId'et for den sag, som genstanden hører til.

Disse attributter går igen i andre entydigt tilknyttede objekter. Klasser, som kun relaterer til én anden klasse (Forsikring, Arkivindhold, Position), har dog ikke behov for parentType attributten. I stedet for parentExtId og parentType attributter har de derfor blot et underelement med en reference til det objekt, de relaterer til.

Web service eksempel

Den samme genstand kunne oprettes via web services således:

```
GenstandServiceStub.Genstand genstand =
          new GenstandServiceStub.Genstand();
genstand.setNummer("0012X0001");
genstand.setBetegnelse("telefon nr 1");
// Etc …

GenstandServiceStub.Create createGenstandParam =
          new GenstandServiceStub.Create();
createGenstandParam.setToken(currentToken);
```

```
createGenstandParam.setWsform(genstand);
createGenstandParam.setParentType("Sag");
createGenstandParam.setParentId(sagStatus.getGeneratedId());

GenstandServiceStub genstandService = new GenstandService-
Stub();
GenstandServiceStub.WSStatus genstandStatus =
          genstandService.create(createGenstandParam).
          get_return();

System.out.println("Status creating sag: " +
          sagStatus.getFaultCode() + ", " +
          sagStatus.getFaultMessage() + ", generatedSagId:" +
          sagStatus.getGeneratedId());
```

Figur 6: Oprettelse af en genstand under en sag via web services

I eksemplet benyttes `sagStatus.getGeneratedId()` til at referere til den sag, som blev oprettet i det tidligere eksempel.

5.4 Løst tilknyttede klasser

Løst tilknyttede klasser kan typisk relatere til flere objekter af forskellige klasser – det er derfor nødvendigt med en mere generel mekanisme til at knytte disse sammen. I XML er løst tilknyttede objekter derfor forsynet med et særligt `<link>` element.

```
<Vaerk extId="KidVaerk:25168I25502">
<nummer>0044</nummer>
<vaerkKIDStatus xsi:nil="true" />
<vaerkArtType xsi:nil="true" />
<!-- Etc … -->
<link objectExtId="Kunstner:8016" objectType="Kunstner" />
<link objectExtId="Kunstner:24206" objectType="Kunstner" />
</Vaerk>
```

Figur 7: Et værk med to ophavsmænd i XML

I web services benyttes et ”Attach” objekt og attach() metode til at knytte eksisterende objekter til hinanden:

```
DeponeringServiceStub.Attach attach =
          new DeponeringServiceStub.Attach();
attach.setObjectId(getDeponeringId());
attach.setParentId(genstandId);
attach.setParentType(parentType);
attach.setToken(Login.loginToken);

DeponeringServiceStub.WSStatus status =
          deponeringService.attach(attach).get_return();
```

Figur 8: En eksisterende deponering knyttes til en genstand via web service

5.5 Særlige klasser

Særlige klasser udgøres af klasser, som relaterer n:m til andre, ligesom de løst tilknyttede klasser. Men relationen er *kvalificeret* med en eller flere oplysninger. Man kan derfor i virkeligheden opfatte disse relationer som selvstændige klasser, men ideologisk tillader DMDS (og Regin) ikke, at objekter af disse klasser eksisterer selvstændigt – de findes kun for at knytte andre objekter sammen:

Relationen imellem Aktør og andre klasser kvalificeres af aktørens rolle.
Relationen imellem Opbevaring og andre klasser kvalificeres af opbevaringens type.
Relationen imellem Stedregistrant og andre klasser kvalificeres af stedets type.

XML eksempel

```
<Aktoer extId="Aktoer:35">
<navn>Jensen, Jens</navn>
<stilling>Lærer</stilling>
<bynavn>Århus</bynavn>
<nationalitet />
<internNote xsi:nil="true" />
<eksternNote xsi:nil="true" />
</Aktoer>

<ObjektAktoer ObjektAktoerType="DeponeringAktoer" parentEx-
tId="Deponering:2" extId="DeponeringAktoer:4239">
<aktoerExtId>Aktoer:35</aktoerExtId>
<declareAktoerRolle>
<entryId>163</entryId>
<kode>LNE</kode>
<tekst>låner</tekst>
</declareAktoerRolle>
<aktoerRolleId>163</aktoerRolleId>
<internNote xsi:nil="true" />
<eksternNote xsi:nil="true" />
</ObjektAktoer>
```

Figur 9: En aktør og dens (”låner”) relation til en eksisterende deponering oprettes i XML

I eksemplet har Jens Jensen, en lærer i Århus, lånt en genstand af museet (selve genstanden og dens uddeponering er ikke vist).

ObjektAktoeren har en ny attribut, ObjektAktoerType, som angiver de klasser, den binder sammen.

Bemærk, at aktøren og dens relation til et andet objekt (her en deponering) oprettes med to forskellige elementer: aktøren selv og en ”ObjektAk-

toer", som repræsenterer relationen og den kvalificerende oplysning om aktørens rolle.[6]

Også via web services sker oprettelsen altså i to tempi:

```
StedregistrantServiceStub.Stedregistrant nyStedregistrant =
          new StedregistrantServiceStub.Stedregistrant();
nyStedregistrant.setEksternNote("Ekstern note");
nyStedregistrant.setStedregistrantKlassifikationId(6);
nyStedregistrant.setStedregistrantGruppeId(6);
StedregistrantServiceStub.Create stedregistrantCreateParam =
          new StedregistrantServiceStub.Create();
stedregistrantCreateParam.setToken(currentToken);
stedregistrantCreateParam.setWsform(nyStedregistrant);
StedregistrantServiceStub stedregistrantService =
          new StedregistrantServiceStub();
StedregistrantServiceStub.WSStatus saveStedregistrantStatus =
          stedregistrantService.
          create(stedregistrantCreateParam).
          get_return();
System.out.println("Save stedregistrant status: " +
          saveStedregistrantStatus.getFaultMessage() + ", " +
          saveStedregistrantStatus.getGeneratedId());

ObjektStedregistrantServiceStub.ObjektStedregistrant
          nyObjektStedregistrant =
          new ObjektStedregistrantServiceStub.
          ObjektStedregistrant();
nyObjektStedregistrant.setStedregistrantTypeId(1);

ObjektStedregistrantServiceStub.Create
          createObjektStedregistrantParam =
          new ObjektStedregistrantServiceStub.Create();
createObjektStedregistrantParam.setParentType("Genstand");
createObjektStedregistrantParam.setParentId(2041570);
createObjektStedregistrantParam.setToken(currentToken);
createObjektStedregistrantParam.
          setConnectorId(saveStedregistrantStatus.
          getGeneratedId());
createObjektStedregistrantParam.
          setWsform(nyObjektStedregistrant);

ObjektStedregistrantServiceStub objektStedregistrantService =
          new ObjektStedregistrantServiceStub();
ObjektStedregistrantServiceStub.WSStatus
          objektStedregistrantCreateStatus =
          objektStedregistrantService.
          cre-
ate(createObjektStedregistrantParam).get_return();

System.out.println("Create Objektstedregistrant status: " +
          objektStedregistrantCreateStatus.getFaultMessage() +
          ", " + objektStedregistrantCreateStatus.
```

[6] Det fremgår også af eksemplet, at relationen yderligere kvalificeres af en intern og ekstern note; disse felter findes på alle objekter i datamodellen, men de er ikke altid eksponeret i Regins brugerflade og dermed synlige for museet.

```
            getGeneratedId());
```

I eksemplet oprettes en Stedregistrant, som derefter bindes til en eksisterende Genstand (med id 2041570) ved hjælp af en ObjektStedregistrant. Det fremgår af ObjektStedregistrantens type, at der er tale om et fundsted (~ StedregistrantTypeId 1).

5.6 Kunstnere

Kunstnere er som sagt et helt kapitel for sig, og der er faldgrubber. Når man skal indføre data i Kunstindeks Danmark, er det derfor vigtigt straks at notere denne gyldne regel:

*Man må først oprette en kunstner, når man er **helt** sikker på, at han/hun ikke findes i forvejen*.

Kunstindeks Danmark har i forvejen næsten 14,000 registreringer, som repræsenterer danske kunstnere – så sandsynligheden for at kunstneren findes i forvejen er stor.

Men hvordan gør man?

Kunstnerregistreringen er i Kunstindeks Danmark delt i to klasser:

- En *GlobalKunstner* indeholder stamoplysninger for kunstneren, som er fælles for alle museer – kunstnerens navn, fødselsår, dødsår, etc. Det er de oplysninger, som står i boksen "stamoplysninger" på Kunstnerskærmbilledet i Regin.

- En (lokal) *Kunstner* repræsenterer kunstnerregistreringen på det konkrete museum, dvs. museets interne noter om kunstneren. Men det er også til objekter af denne klasse, at man tilknytter værker, udstillinger, litteratur, kommunikationer, genstande m.m., som museet har registreret om kunstneren. De underentiteter, som optræder på Kunstnerskærmbilledet er altså knyttet til Kunstner entiteten – og værker kan *ikke* knyttes til en Globalkunstner.

Man kan derfor mere præcist sige: En kunstner må i Kunstindeks Danmark være repræsenteret af et og kun et GlobalKunstner objekt; og af et og kun et Kunstner objekter for hvert museum, som har værker af kunstneren.

Når man opretter et værk, skal man derfor følge denne algoritme:

Opret et Værk objekt

Tjek om der findes et Kunstner objekt, som repræsenterer kunstneren.
Hvis ja [kunstneren er tidligere registreret på museet]:
Knyt værket til denne Kunstner.
Hvis nej:
Tjek om der findes en GlobalKunstner, som repræsenterer kunstneren.
Hvis ja [kunstneren findes i KID, men er ikke registreret på museet]:
Opret en (lokal) Kunstner
Knyt den til GlobalKunstneren
Knyt værket til Kunstneren.
Hvis nej [kunstneren findes ikke i KID]:
Opret en GlobalKunstner
Opret en (lokal) Kunstner
Knyt dem sammen
Knyt værket til Kunstneren.

(Iterér over kunstneroprettelsen, hvis et værk kan have flere kunstnere tilknyttet.)

GlobalKunstnere relaterer 1:n til Kunstnere, så Kunstnere oprettes og tilknyttes ligesom andre entydigt tilknyttede klasser (jf. afsnit 5.3). Kunstnere relaterer n:m til Værker, så Værker oprettes og tilknyttes ligesom andre løst tilknyttede klasser.

Hvordan spørger man efter eksisterende Kunstnere og GlobalKunstnere?

Når man benytter web services, kan man søge efter eksisterende Kunstnere og GlobalKunstnere på samme måde som alle andre objekter. I den forbindelse er det særligt værd at bemærke mulighederne i Regins indbyggede søgemaskine, Lucene, som anvendes både ved søgninger igennem webklienten og web services. Lucene parseren[7] giver mulighed for at benytte blandt andet ”fuzzy searches” (~ søgninger) til at imødegå den meget væsentlige udfordring med stavemåde ved søgninger efter eksisterende kunstnere. Dermed bliver det muligt f.eks. at bede brugeren tage stilling til en liste over mulige matches, såfremt den præcise søgning ikke giver noget resultat.

Når man benytter XML, må man foretage en XML eksport fra museet for at få en liste over eksisterende Kunstnere (og deres extId’er) på museet. En liste over alle GlobalKunstnere kan hentes i et særligt XML tabel format fra

[7] Se Lucene parserens dokumentation på http://lucene.apache.org/java/docs/queryparsersyntax.html

XML eksport siderne i Regin ved tryk på knappen ”GlobalKunstner”. Den øverste række i tabellen viser feltnavnene.

6 Hvad nu?

1. Læs resten af dokumentationen. Både DMDS schema og web service referenceeksemplerne er annoterede.
2. Skriv til os med dine spørgsmål (se kontaktinfo herunder) – vi kan ikke svare på alt på stående fod, men der er som regel en løsning på de problemer, du løber ind i.
3. Hvis du har brug for at prøve kræfter med Regin XML og web services uden frygt for at komme til at skade museets eksisterende registrering, kan du også skrive til os – så kan vi evt. efter nærmere aftale oprette et ”test museum” til dig.

7 XML eller web services – hvad skal jeg bruge?

Hvis du er i tvivl om hvilken metode, du skal bruge til at udveksle data med centralregistrene, kan du overveje følgende tommelfingerregler.

Brug XML når:

- du skal overføre store mængder af data sjældent
- du gerne vil have et udtræk af dine data i registeret i et menneskelæsbart format
- du har valgt at bruge Regin til at styre museets samling, og du én gang for alle skal konvertere data i din lokale database
- du foretrækker at indberette tilføjelser til samlingen manuelt med jævne mellemrum

Brug web services når:

- du skal overføre små mængder af data ofte
- du ønsker dynamisk at trække oplysninger fra centralregisteret til nye præsentationer på internettet, f.eks for at trække oplysninger fra registeret til en hjemmeside
- du udvikler dine egne applikationer, som skal integreres tæt med Regin (f.eks. et særligt konserveringsmodul)
- du foretrækker at indberette alle ændringer i museets samling løbende og automatisk

8 Hvordan får jeg adgang til Regin XML og web services?

For at få adgang til Regin XML og web services, skal du have en Regin konto med de rigtige privilegier på museet. Det er museets lokale Regin administrator (også kendt som den "ansvarshavende registrator"), som opretter denne konto og tildeler privilegierne.

- For at eksportere XML fra Regin skal man have "Læse (login)" og "XML Eksport" privilegierne – XML eksport sker fra Regin, så man er nødt til at være logget ind her.
- For at importere XML til Regin skal man have "Læse (login)", "XML Import", "Registrator" og – afhængigt af hvilke data, man skal indføre – desuden "Sagsansvarlig" privilegierne.
- For at læse via Regin web services skal man have "Læse (login)" og "Web service Læse" privilegierne.
- For at skrive til Regin via web services skal man have "Læse (login)", "Web service Skrive", "Registrator" og – afhængigt af hvilke data, man skal indføre – desuden "Sagsansvarlig" privilegierne.

Hvis den lokale administrator er i tvivl om fremgangsmåden, er han eller hun selvfølgelig velkommen til at kontakte Kulturstyrelsen for vejledning.

9 Kontakt, spørgsmål og svar

Har du yderligere spørgsmål om Regin XML eller web services, eller har du kommentarer til denne vejledning, kan du skrive til regin-support@kulturstyrelsen.dk.

Bilag 1: Klasser
som her er nævnt i den rækkefølge, DMDS schema kræver af objekter i XML filer:

1. Sag
2. GlobalKunstner
3. Værk
4. Kunstner
5. Skib
6. Genstand
7. Kommunikation
8. FotoFilm
9. Magnetmedium
10. Rapport
11. Storformat
12. Arkivfond
13. Aktør
14. Litteratur
15. Udstilling
16. Forsikring
17. Konservering
18. Opbevaring
19. Objektopbevaring
20. Deponering
21. Arkivindhold
22. Stedregistrant
23. Position
24. LokalitetDanmark
25. LokalitetUdland
26. Klassifikation
27. Datering
28. MIMEtypeobjekt
29. Adresse

Bilag 2: Stamtabelnavne
Listen herunder indeholder navnene på alle stamtabeller i Regin.

h_AdresseType
h_AktoerRolle
h_AlternativtNavnType
h_ArkivindholdIndhold
h_ArkivindholdType
h_Autenticitet
h_DateringMetode
h_DateringType
h_DeponeringType
h_Ejerlav
h_FotoFilmFormat
h_FotoFilmType
h_GenstandArtType
h_GenstandBetegnelse
h_GenstandErhvervelse
h_GenstandFundmetode
h_HenvisningArt
h_Kommunenummer
h_KommunikationType
h_LitteraturDecimalklassifikation
h_LokalitetAnlaegsregistrant
h_LokalitetType
h_MagnetmediumType
h_Materiale
h_Nationalitet
h_OCMGrundlag
h_OCMKildetype
h_OCMKode
h_OCMKvalitet
h_OWCkode
h_OpbevaringType
h_Periode: denne stamtabel er forældet og erstattet af:
h_Periode2
h_PositionPositivretning
h_PositionType
h_Postadresse
h_Prioritering
h_Proveniens
h_RapportType
h_Repraesenteret
h_SagStatus
h_SkibForhudning
h_SkibKonstruktion
h_SkibTilknytning
h_SkibType

h_Sted
h_StedregistrantGruppe
h_StedregistrantKlassifikation
h_StedregistrantType
h_StedregistrantUndergruppe
h_StorformatFormat
h_StorformatType
h_Topografinummer
h_UdstillingType
h_VaerkErhvervelsesmaade
h_VaerkKIDStatus
h_VaerkTitelType
h_Vaerktype

Bilag 3: Web service fejlkoder

Fejlkode	Tekst	*Note*
2000	OK	
4000	Bad request	
4001	Unauthorized	*Similar to 4003 Forbidden, but specifically for use when authentication is possible but has failed or not yet been provided (No token provided).*
4003	Forbidden	
4004	Not found	
5000	Internal server error	
5003	Service unavailable	
Application specific faultcodes		
6000	Object not found	
6001	Field not found	
6002	Illegal context	
6003	Nummer not unique	
6004	Following field(s) contains an illegal value	
6010	File too big	
6020	GlobalKunstner is created by another museum	
6022	GlobalKunstner in use by other museums	
6024	Kunstner in use by Vaerk objects	
6026	GlobalKunstner linked to Weilbach	
6030	Trying to change a field value which the user does not have the rights to do	
6040	*Image error*	*An unspecified problem with upload of an image or other MIME type object.*