clarin.dk Tools

clarin.dk projektet arbejdspakke 5.1

1.2.2011

Bart Jongejan, bartj@hum.ku.dk

Lene Offersgaard, leneo@hum.ku.dk

**Versionshistorie**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dato** | **Version** | **Logtekst** | **Ansvarlig** | **Status** |
| 20/12-2010 | 1.0 | Første udkast | Bart Jongejan | Første udkast |
| 21/12-2010 | 1.1 | Anders Conrads forslag indarbejdet | Bart Jongejan | Draft |
| 12/1-2011 | 1.2 | Resultat fra mødet med Jørg og Jakob og fra Tools-review indarbejdet | Bart Jongejan | Draft |
| 17/1-2011 | 1.3 | Kommentarer fra Anders, Jacob, Lene og Tomas indarbejdet | Bart Jongejan | Draft |
| 24/1-2011 | 1.4 | Mere om workflow-generering, nyt xml-eksempel i kap. 3. | Bart Jongejan | Prefinal |
| 1/2-2011 | 1.5 | Sidste kommentarer indarbejdet | Bart Jongejan | Final |

Indholdsfortegnelse

[Indledning 3](#_Toc284338221)

[1 Ændringer i arkitekturdokumentet 3](#_Toc284338222)

[2 Tilføjelser til arkitekturdokumentet 4](#_Toc284338223)

[2.1 Tools: funktionalitet og arkitektur 4](#_Toc284338224)

[2.2 Træk 4](#_Toc284338225)

[2.2.1 Facet 5](#_Toc284338226)

[2.2.2 Mål 5](#_Toc284338227)

[2.2.3 Dataformat, sprog og andre træk 6](#_Toc284338228)

[2.2.4 Trækværdispecialiseringer (stil) 7](#_Toc284338229)

[2.2.5 Administration af trækbegreber 7](#_Toc284338230)

[2.3 Registrering af værktøjer 7](#_Toc284338231)

[2.3.1 Metadata 7](#_Toc284338232)

[2.3.2 Registrering af Træk 8](#_Toc284338233)

[2.3.3 Registreringsforløb 8](#_Toc284338234)

[2.3.4 Registreringsforløb i pseudokode 9](#_Toc284338235)

[2.3.5 Registreringsmodulets URI 10](#_Toc284338236)

[2.3.6 Rettelse af oplysninger 15](#_Toc284338237)

[2.3.7 Sletning af værktøjsoplysninger eller inaktivering af et værktøj 17](#_Toc284338238)

[2.3.8 Værktøjsregistrering i detaljer: Backus–Naur Form 18](#_Toc284338239)

[2.4 Aktivering 19](#_Toc284338240)

[2.4.1 Aktivering i pseudokode 20](#_Toc284338241)

[2.4.2 Aktivering: URL 21](#_Toc284338242)

[2.5 Datasikkerhed 24](#_Toc284338243)

[3 Værktøjsintegrationsmanual 25](#_Toc284338244)

[3.1 On-line registrering af værktøjer 25](#_Toc284338245)

[3.2 Afgrænsning 25](#_Toc284338246)

[3.3 Registreringsformularen 26](#_Toc284338247)

[3.3.1 Generelle tools-metadata 26](#_Toc284338248)

[3.3.2 Registrering af integrerede værktøjer 26](#_Toc284338249)

[3.4 Forberedelse til integration af et nyt værktøj 27](#_Toc284338250)

[3.4.1 Webservice 27](#_Toc284338251)

[3.4.2 Resursetyper 27](#_Toc284338252)

[3.4.3 Parametre 27](#_Toc284338253)

[3.4.4 Parametre som request-streng 28](#_Toc284338254)

[3.4.5 Aktivering af webservice med POST- i stedet for GET-request 31](#_Toc284338255)

[3.4.6 HTTP status codes 33](#_Toc284338256)

[3.4.7 Afvikling 33](#_Toc284338257)

# Indledning

Denne manual beskriver nye udviklinger i Tools-modulet siden udkomst af ”DK-CLARIN: specifikation af teknisk infrastruktur” d. 4. maj 2010. Der lægges især vægt på integrationsprocessen for værktøjer i clarin.dk’s infrastruktur.

# Ændringer i arkitekturdokumentet

Nogle afsnit i “specifikation af teknisk infrastruktur” er ikke længere retvisende, for det meste da arkitekturen for aktivering af værktøjer nu inkluderer en ”workflowplanner” til fuldautomatisk beregning af hvordan enkelte værktøjsopgaver kan kædes sammen så at brugerens specifikation af det ønskede resultat bliver opfyldt når afvikling af opgaverne i workflowet er fuldført. Denne ændring medfører at de aktiverede værktøjer altid passer til de data de arbejder på, og at en separat valideringsfunktion derfor er overflødiggjort. Ændringen forudsætter at data og værktøjer er nøje og korrekt karakteriseret ved hjælp af metadata. Dette er ændringerne:

* Afsn. 2.6 “Værktøjer og services”. Følgende slettes: *Denne kompleksitet kan infrastrukturen ikke løfte og derfor vil infrastrukturen ikke tilbyde brugeren automatisk valg af værktøjer. I DK-CLARIN vil brugeren således selv skulle udvælge de værktøjer vedkommende ønsker at benytte, og brugeren må selv vha. værktøjernes dokumentation udvælge det rette værktøj og specificere de rette optioner.*

Begrundelse: det bliver muligt for brugeren at specificere det ønskede resultat, hvorefter infrastrukturen beregner måderne hvorpå resultatet kan nås, hvis det kan nås. Brugeren kan supplere sit ønske med angivelsen af et foretrukket værktøj, men det er ikke nødvendigt.

* Samme afsnit, følgende slettes: *Selvom de integrerede værktøjer kan tilgås direkte i den offentlige infrastruktur er der brug for et værktøjsvalidator-modul som tjekker om de parametre brugeren har specificeret som input til værktøjet opfylder værktøjets formelle krav. Dette modul kan web-grænsefladen kalde for at få en liste over et værktøjs krav til input-data og for at få en tilbagemelding til brugeren om parametrene opfylder de formelle krav. For hvert værktøj der integreres skal der således defineres en parameterListfunktion og en parameterValideringsfunktion.*

Begrundelse: Ved registrering af et integreret værktøj skal alle for værktøjet relevante input- og outputparametre og deres mulige værdier specificeres. Disse oplysninger bruger infrastrukturen til at beregne hvordan værktøjer kan sættes sammen for at opnå et mål.

* Afsn. 2.8.5 ”Bruge et integreret værktøj til at skabe en ny ressource i repositoriet”. Følgende sætning erstattes: *Der er ikke en automatisk metode til at udvælge de værktøjer der er mulighed for at benytte til en given ressource, men alle værktøjswebservices skal kunne liste de parametre der er nødvendige for at aktivere webservicen og tilbyde en validering af de parametre der overføres til værktøjet.*

Erstatning: *Der er en automatisk metode til at udvælge de værktøjer der er mulighed for at benytte til en given ressource.*

* Samme afsnit. Den sidste del af den sidste sætning slettes fra *ved at*: *Da værktøjerne er meget forskellige er det en udfordring at generalisere deres grænseflader, men det tilstræbes at workflow’et omkring specifikation af inputparametre for værktøjerne ensartes så meget som muligt {ved at det håndteres af et værktøjsvalidator-modul som specificerer og verificerer inputparametre over for brugeren}.*
* Samme delsætning slettes fra den sidste sætning i 2.8.7 ”Bruge et integreret værktøj til at udtrække/vise information”: *Specifikation af inputparametre for alle værktøjerne ensartes så meget som muligt {ved at det håndteres af et værktøjsvalidator-modul som specificerer og verificerer inputparametre over for brugeren}.*
* Afsn. 3.11 ”værktøjer”: SOAP understøttes ikke i eSciDoc. Derfor slettes de to sidste ord fra ”Integrerede værktøjer implementeres som webservices vha. REST *{eller SOAP}*.”

# Tilføjelser til arkitekturdokumentet

## Tools: funktionalitet og arkitektur

I afsnit 5.6 af arkitekturdokumentet udskydes en nærmere specifikation af funktionalitet og arkitektur til næste projektfase. Her følger den mere dybdegående beskrivelse.

Værktøjsmodulet har to overordnede funktionaliteter: registrering og aktivering af værktøjer.

Værktøjsmodulet kan håndtere registrering af både integrerede og ikke integrerede værktøjer. Integrerede værktøjer kan indgå i workflows som værktøjsmodulet beregner og efterfølgende afvikler. Ved afvikling af et workflow aktiverer værktøjsmodulet de planlagte værktøjer alt imens det holder øje med mellemresultater og først aktiverer et værktøj når dets input ligger klar. Hvis et værktøj kan downloades fra infrastrukturen eller aktiveres ved at følge et link, men ikke indgå i et workflow, er værktøjet ikke integreret.

Et integreret værktøj registreres ved at beskrive værktøjets inputdata og outputdata, som i øvrigt normalt vil være clarin.dk resurser. Når værktøjsmodulet skal beregne et workflow, afsøger den alle måder hvorpå de givne inputdata kan føres til de givne outputdata ved hjælp af en kæde af værktøjer med passende input- og outputkvalifikationer og generering af eventuelt nødvendige mellemresultater.

Vi ser derfor at vi har behov for et begrebsapparat til beskrivelse af resurser, og at vi også har behov for en praktisk måde at udvide begrebsapparatet på når der kommer nye typer af resurser på banen. Derfor er der udover registrering og aktivering en tredje funktionalitet i værktøjsmodulet: redigering af et antal lister der definerer ved hvilke egenskaber resurserne i clarin.dk kan kendetegnes. Disse lister optræder som picklister ved registrering og aktivering af integrerede værktøjer og skal kunne udvides efter behov når nye type resurser håndteres af infrastrukturen. Listerne kan eventuelt redigeres “i hånden” i en simpel teksteditor af en person med administratorrolle i infrastrukturen.

Egenskaber som, for værktøjmodulets vedkommende, i tilstrækkelig grad beskriver resurser, kan overordnet inddeles i *træk* og *stil*. *Træk* er de egenskaber som kræver overensstemmelse mellem værktøj og resurse, mens en *stil* er en valgfri specialisering af en trækværdi som først stiller sig i vejen for overensstemmelse mellem værktøj of resurse hvis både resurse og værktøj har angivelser for stil og disse angivelser ikke er enslydende.

I de følgende afsnit vil vi gå dybere ind på *træk* og *stil*.

## Træk

Et centralt begreb i Tools-modulet er *træk*. Tools-modulet baserer sin algoritme til beregning af workflows på de egenskaber inputtet har og de egenskaber det ønskede output skal have. Egenskaberne ved et input eller et output beskrives i et antal træk som for eksempel dataformat, sprog og facet.

### Facet

Hvor sprog og, i lidt mindre grad, dataformat er forholdsvis klare begreber, er det sidstnævnte træk, *facet*, mere abstrakt og kræver en nærmere forklaring. En resurses facetter er de intellektuelle anskuelsesmuligheder vi har på resursen. Hver gang der skabes en ny facet til en resurse, får beskueren en ny mulighed for intellektuelt at komme tættere ind på resursen.

Valget af navnet ”facet” er inspireret af diamantindustrien, som byder på et glimrende eksempel på hvordan destruktive processer, hvis de er udført med snilde, øger resursens, i dette tilfalde diamantens, attraktionsværdi. Andre eksempler, som er tættere på humanistiske fag, er den evige afvejning i arkæologien mellem opgravning eller bevarelse af gamle artefakter for efterslægten, og mellem digitalisering eller fysisk opbevaring af ”analogt” tekst-, billed- og lydmateriale.

I den digitale verden er den særlige omstændighed at man kan multiplicere en resurse uden at gøre skade på resursen. Derfor vil der, hvis man husker at kopiere det digitale udgangsmateriale før man skaber en ny facet på en resurse, altid være tale om skabelsen af en ny resurse for hver gang man skaber en ny facet. Produktet af skabelsesprocessen, som er en resurse, kan indgå i en ny skabelsesproces, og så videre. Derfor er det nærliggende at sætte lighedstegn mellem facet og resurse, men den forestilling er ikke helt dækkende, da der også kan skabes nye resurser ved processer som ikke er destruktive og derfor ikke skaber nye facetter – kopiering er et trivielt eksempel på en sådan proces.

En resurses facetter er resursens anskuelsesmuligheder, men udgør ikke resursens substans (hvis resursen er materiel) eller informationsindhold (hvis resursen er data). Ikke-identiske resurser kan have de samme træk, og facettrækket er ingen undtagelse.

For at afgrænse facetbegrebet er det nyttigt at lave følgende tankeøvelse. Hvis det teoretisk er muligt at bruge et automatisk værktøj til at skabe en resurse B fra en resurse A og at bruge et andet automatisk værktøj til at genskabe A fra resurse B, så har A og B nødvendigvis samme intellektuelle indhold, og har værktøjerne ikke skabt nye facetter til resurserne A og B. Hvis A og B ikke desto mindre har vidt forskellige egenskaber, kan forskellene ligge i dataformat, sprog eller andre træk, men ikke i facet-trækket.

En subjektiv vurdering om to resurser har samme facet-træk kan ikke altid undgås: hvis A ikke ”helt præcist” kan genskabes fra resurse B, men dog ”næsten”, kan man af praktiske grunde vælge at lukke øjnene for forskellene og tilskrive A og B samme facet. Som eksempler kan nævnes at oversættelse fra sprog X til sprog Y normalt ikke er 100 % reversibel, og at der normalt sker en irreversibel kvalitetsforringelse ved konvertering fra et videoformat til et andet.

Omvendt gælder det ikke at hvis A og B har samme facettræk, A kan blive til B og B til A. A og B’s faktiske indhold (substans eller informationsindhold) kan være forskellige, på trods af at de har samme facettræk.

### Mål

I en forskningsinfrastruktur som clarin.dk er der som regel en stor lighed mellem begreberne *mål* (eng. *goal*) og *facet*. Målet i mange forskningsprojekter er analysen af et givent udgangsmateriale. Selvom analysen af materialet kan beskadige materialet, skaber man en viden om materialet som man ikke havde før. Derfor svarer et mål i et analyseworkflow og alle delmål der fører til målet, til facetter. Selv de datalogiske resurser der betragtes som basisresurser i clarin.dk er facetter, da de er produkter af en forudgående bearbejdning og fortolkning, og blot giver os standardiserede anskuelsesmuligheder på noget der måske ikke længere eksisterer i sin helhed. Til en start kan vi skrive alle navne på basisresursetyperne vi håndterer i clarin.dk i listen af facet-træk:

Tekst

Video

Lyd

Billede

Afledte facetter for basistype tekst er:

Sætninger

Tokens

Ordklasser

Lemmaer

Søgeindeks

Frekvensliste

Også for lyd og video kan vi liste afledte facet-træk:

Fonemer

Prosodi

Hoved- og håndbevægelser

### Dataformat, sprog og andre træk

Hvis en resurse på reversibel vis kan transformeres til en anden resurse, er der tale om forskellige metoder til at udtrykke samme intellektuelle indhold. Udtryksmetoden er delvist et teknisk anliggende, og delvist et sproglig et. Til den tekniske del bliver der i clarin.dk anvendt et begrænset antal standardiserede formater for hver resursetype, mens der a priori ikke udelukkes nogen sprog, da sprog afhængigt af brugerens forskning kan være middel eller mål, og dermed både kan være formatagtig og facetagtig.

Udover facet, format og sprog kan listen af træk udvides efter behov, men i skrivende stund kendes der ikke nogen oplagte kandidater. Selvom listen af træk nemt kan udvides, bør den holdes så kort som muligt. Det skal undgås at forskellige træk adresserer beslægtede egenskaber: træk skal være ortogonale, hvilket vil sige at de kan variere uafhængigt af hinanden: en tekst (facet) kan være på tysk, dansk eller kinesisk (sprog) og være udformet som en flad tekstfil, en html-side, eller clarin.dk standardformat for basistekst (dataformat). Et andet krav til nye træk bør være at de kan anvendes på, hvis ikke alle, så i hvert fald en meget stor del af de resursetyper clarin.dk håndterer.

Man kan være i tvivl om værktøjer der nøjes med at formindske en resurse (fx ved sampling eller ved summarization) kan siges at resultere i en resurse med andre træk end den oprindelige resurse. Et argument for denne opfattelse er at processen er irreversibel, og at outputtet derfor må udgøre en ny facet. Modargumentet er at formindskelsesfaktoren kan ligge vilkårligt tæt på 1, og at outputtet i så fald i realiteten ikke åbner for en ny anskuelsesmulighed på resursen, og derfor ikke er en ny facet. Vi anbefaler derfor at indføre et ekstra træk der har til formål at redegøre for delmængder.

### Trækværdispecialiseringer (stil)

Ofte støder vi på forskellige, uforenelige opfattelser omtrent en facet. Hvor alle ubesværet kan håndtere en tekst den ene dag og en video den anden uden den ringeste nødvendighed for at forklare sig, kan det føre til en ophedet debat om man skal analysere tekstfragmentet *can’t* som *can* og *‘t* eller *ca* og *n’t*. Eller måske som *can* og *not*? Debatten har grobund i den omstændighed at de forskellige analyser gør krav på samme facet: de handler alle om tokens. For nu at give plads til uforenelige opfattelser introducerer vi specialiseringer.

En specialisering til en trækværdi (*tokens* er en trækværdi, *facet* er et træk) gør det muligt at forskellige “skoler” kan koeksistere i clarin.dk. Ikke kun trækværdier for trækket *facet* kan tænkes at skulle specialiseres, også værdier for *format* og *sprog* skal i nogle tilfælde nærmere kunne specificeres. For eksempel kan der være forskellige skoler indenfor formattrækværdien *video* (mov, avi, mp4, …) eller sprogtrækværdien *engelsk* (britisk, nordamerikansk, sydafrikansk, …).

### Administration af trækbegreber

De i clarin.dk kendte træknavne (*facet*, *format*, *sprog*, …), trækværdinavne (*tokens*, *video*, *tysk*) og navne på specialiseringer (*Penn Treebank*, *AVI*, *Mittelhochdeutsch*) skal alle skrives ind i nogle tabeller som Tools-modulet har adgang til. Disse tabeller bliver præsenteret som picklister når et værktøj skal registreres. Ved at vælge de navne der bedst beskriver input og output for et værktøj fortæller værktøjsudbyderen hvad der kræves af inputtet for at det passer til værktøjet, og hvad der kommer ud som output. Omvendt er det en forudsætning for Tools-modulets gnidningsløse funktioneren at en resurse røber de træk der karakteriserer resursen ved hjælp af resursens metadata.

## Registrering af værktøjer

Registrering af værktøjer vil foregå i en interaktiv proces, hvor værktøjsudbyderen via en formular på infrastrukturens websted specificerer søgbare metadata. I en mere teknisk orienteret del af formularen er det muligt at indtaste de trækoplysninger Tools-modulet skal have kendskab til for at beregne hvordan værktøjsopgaver kan sættes sammen i workflows. Den del skal kun udfyldes for værktøjer der skal integreres i infrastrukturen.

### Metadata

Generelle søgbare metadata er:

*ToolID*

*Version*

*Title*

*Publisher* copyright owner

*Content Provider* den institution, der registrerer værktøjet

*Creator* de(n) person(er), el. det projekt der har lavet værktøjet

*Rights*

*InfoAbout*

*ServiceURL* for integrerede værktøjer

*ExternalURI* for eksterne værktøjer

*ConformsTo*

*Description*

*Facets*

*DataFormats*

*Languages*

Ikke søgbare metadata er:

*XMLparms*

*PostData*

*Inactive*

XMLparms, PostData og Inactive er Bool’ske felter. *XMLparms* angiver at værktøjet ønsker at modtage parametrene i XML-format i stedet for som form-data. *PostData* angiver at værktøjet ønsker at modtage al input i en POST-request fra infrastrukturen i stedet for selv at sende en GET-request til infrastrukturen. Disse parametre forudsætter at ConformsTo er POST og ikke GET. *Inactive* fortæller at værktøjet midlertidigt er ude af drift på *ServiceURL*.

Hvis et værktøj er integreret, er Facets, DataFormats og Languages ustrukturerede gentagelser af de trækoplysninger som står i den teknisk orienterede del af formularen. Hvis et værktøj ikke er integreret, er disse tre felter den eneste kilde til viden om hvilken slags resurser værktøjet kan producere, og hvilke formater og sprog understøttes.

Alle metadatafelter skal udfyldes af værktøjsudbyderen. Undtaget er dog *version*, der er optionel, samt *serviceURL* og *ExternalURI* der kun udfyldes når det er relevant.

### Registrering af Træk

Den tekniske registrering af et værktøj går ud på en nøje beskrivelse af inputtet til værktøjet og af outputtet som værktøjet producerer. Både input og output beskrives ved at beskrive de træk som input og output har. Hvis et træk ikke er relevant, skal det ikke beskrives, men mindst ét træk skal beskrives.

Man kan registrere flere *inkarnationer* af et værktøj. For eksempel kan man definere en inkarnation af en lemmatiser for engelsk og dansk som kan tage POS-tags som del af inputtet, mens en anden inkarnation, der tager sig af tysk, fransk og nederlandsk, ikke kan håndtere POS-tags. Alle inkarnationer af et værktøj har de generelle metadata som brugere kan søge i, tilfælles. Hvis man kommer til at definere flere inkarnationer end nødvendigt (hvis to inkarnationer er ens eller kun er forskellige i ét træk), bliver disse automatisk sammenføjet af registreringsmodulet.

### Registreringsforløb

Registrering kan ske trinvis eller i én omgang. Ved trinvis registrering svarer registreringsmodulet med en opdateret formular, som indeholder alle data som værktøjsudbyderen allerede har indtastet, eventuelt suppleret med nye indtastningsfelter som, givet de indtastede data, er blevet relevante. Her er fire eksempler på hvordan registreringsformularen dynamisk ændres.

1. Hvis værktøjsudbyderen har indtastet navnet på et i forvejen registreret værktøj og beder om at få opdateret formularen, svarer registreringsmodulet med en formular indeholdende alle registrerede oplysninger vedrørende værktøjet. Derefter kan værktøjsudbyderen ændre og gemme oplysninger efter behov.
2. Hvis værktøjsudbyderen allerede har tilkendegivet at værktøjet håndterer dansk input, men at der er flere sprog værktøjet kan håndtere, svarer registreringsmodulet med en nyt formular som oplyser at dansk er inputsproget og med et nyt udfyldningsfelt hvor værktøjsudbyderen kan udfylde et andet sprog. Dette kan gentages indtil alle sprog er udfyldt.
3. Hvis værktøjsudbyderen har oplyst at værktøjet tager input som belyser ”pos-tagger”-facetten af resursen, er der ved opdatering af formularen nu også et felt som beder om at specificere tag-settet (”POS-tagging style”). Sidstnævnte ændring i formularen er altså en funktion af *værdien* af en tidligere indtastning. Havde værktøjsudbyderen oplyst at inputtet havde facetten ”tokenizer”, ville formularen efter opdatering have vist et ”tokenizer style” inputfelt, og ikke et ”pos-tagger style” inputfelt.
4. Sæt at værktøjsudbyderen allerede har oplyst at inputtet kan være dansk, og at outputtet i så fald også er dansk. Hvis værktøjet også kan håndtere tysk, kan sproget ”tysk” ikke bare tilføjes til listen af inputsprog, for det ville implicere at der skete en oversættelse fra tysk (inputsprog) til dansk (outputsprog). Dette er næppe meningen, medmindre værktøjet er et oversættelsesprogram. I stedet skal værktøjsudbyderen oplyse at der er flere input-/output-par for sprogtrækket. Efter opdatering af formularen er der nu et nyt par felter hvor værktøjsudbyderen kan oplyse at inputsproget kan være tysk, og at outputsproget i det tilfælde også er tysk.

Under alle faser i en dialog med registreringsmodulet bliver værktøjsudbyderens valg udtrykt som html formdata i en URI, og intet andet. Et svar fra registreringsmodulet er et nyt xhtml-form, og er dermed både XML data og HTML som eventuelt kan vises direkte til brugeren. XML-dataene består for en stor del af optionslister som opremser alle værdier som er valgbare og gyldige under de givne forhold. Kun i den del af formularen som henvender sig til værktøjer generelt – både integrerede og ikke integrerede – er felterne åbne for fri tekstinput. For visse felter, fx ToolID, sker en validering når man forsøger at gemme indholdet af formularen, og en fejlbesked genereres hvis et felt ikke er udfyldt korrekt. Den valgte løsning gør det muligt at registrere et nyt værktøj i ét enkelt kald til registreringsmodulet. Det kræver bare at URI’en hvormed registreringsmodulet aktiveres, er fyldestgørende og korrekt.

Eksisterende værktøjsdefinitioner kan redigeres. ToolID-værdien identificerer værktøjet utvetydigt, hvorefter man har mulighed for at redigere én inkarnation ad gangen. Det er op til værktøjsudbyderen at opdatere værktøjsdefinitioner når der sker ændringer i værktøjets funktionalitet

### Registreringsforløb i pseudokode

**Do /\*** *Specify general data for tools\*/*

Enter searchable metadata

**Until** ToolID is unique

**If** Tool must function as an integrated tool

**Do /\****Specify one or more incarnationsof the tool\*/*

**Do /\****Specify one or more features\*/*

Select a feature to specify

**Do** */\* Specify possible input/output combinations*

*for the selected feature\*/*

**Do /\****Specify input and corresponding output\*/*

**Do /\****Specify input\*/*

Enter input feature value

**While** need to subspecify

Enter subspec of input feature value

**Until** all input feature values specified

**Do /\****Specify output\*/*

Enter output feature value

**While** need to subspecify

Enter subspec of output feature value

**Until** all output feature values specified

**Until** input and corresponding output specified

**Until** all possible input/output combinations specified

**Until** all features relevant for the tool are specified

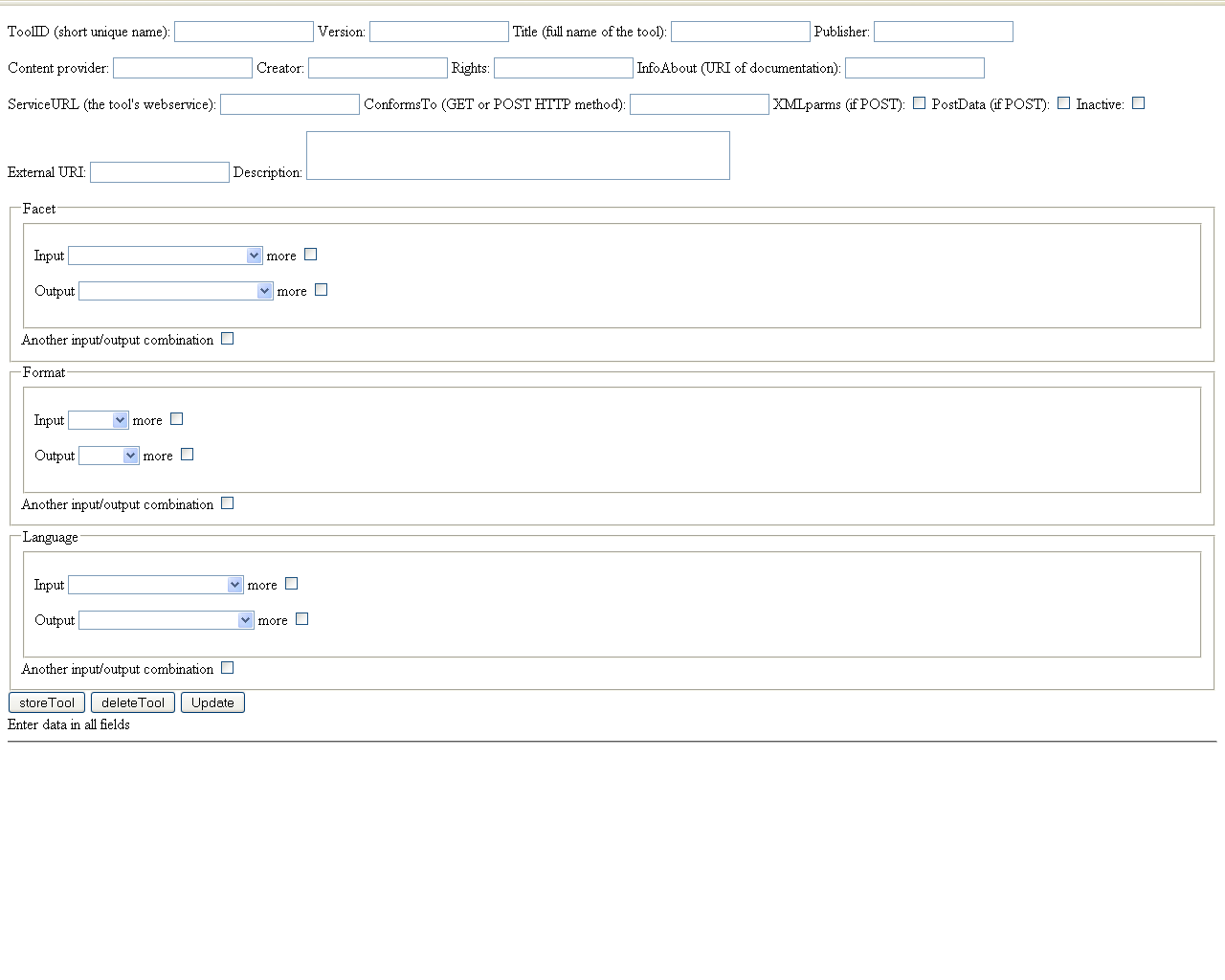
**Until** all incarnations of the tool are specified

### Registreringsmodulets URI

Før har vi beskrevet registreringsforløbet som udfyldning af en HTML formular. I det følgende gennemgås et eksempel på registrering af et værktøj.

Bag kulisserne styres registreringsmodulet ved hjælp af http-parametre.

Hvis registreringsmodulet aktiveres uden parametre, svarer modulet med en tom, ret spartansk HTML-formular (med forbehold for manglende felter):



Formularen består af en øvre del med tekstfelter, en midterdel med listebokse og tjekbokse og en nedre del med knapper: *storeTool*, *deleteTool* og *Update*.

Trykker vi på *Update* knappen, sendes et GET-request til registreringsmodulet med denne URI:

http://infra.clarin.dk/tools/register?name=&vrsion=&fllNm=&pblshr=&cntPrvdr=&crtr=&rghts=&docuri=&url=&cnfrms=&xuri=&dscrptn=&Facet1.1Input=&Facet1.1Output=&Format1.1Input=&Format1.1Output=&Language1.1Input=&Language1.1Output=&bsubmit=Update

Den øvre del af formularen er afspejlet i disse parametre:

name=&vrsion=&fllNm=&pblshr=&cntPrvdr=&crtr=&rghts=&docuri=&url=&cnfrms=&xuri=&dscrptn=

Den midterste del af formularen er afspejlet i disse parametre:

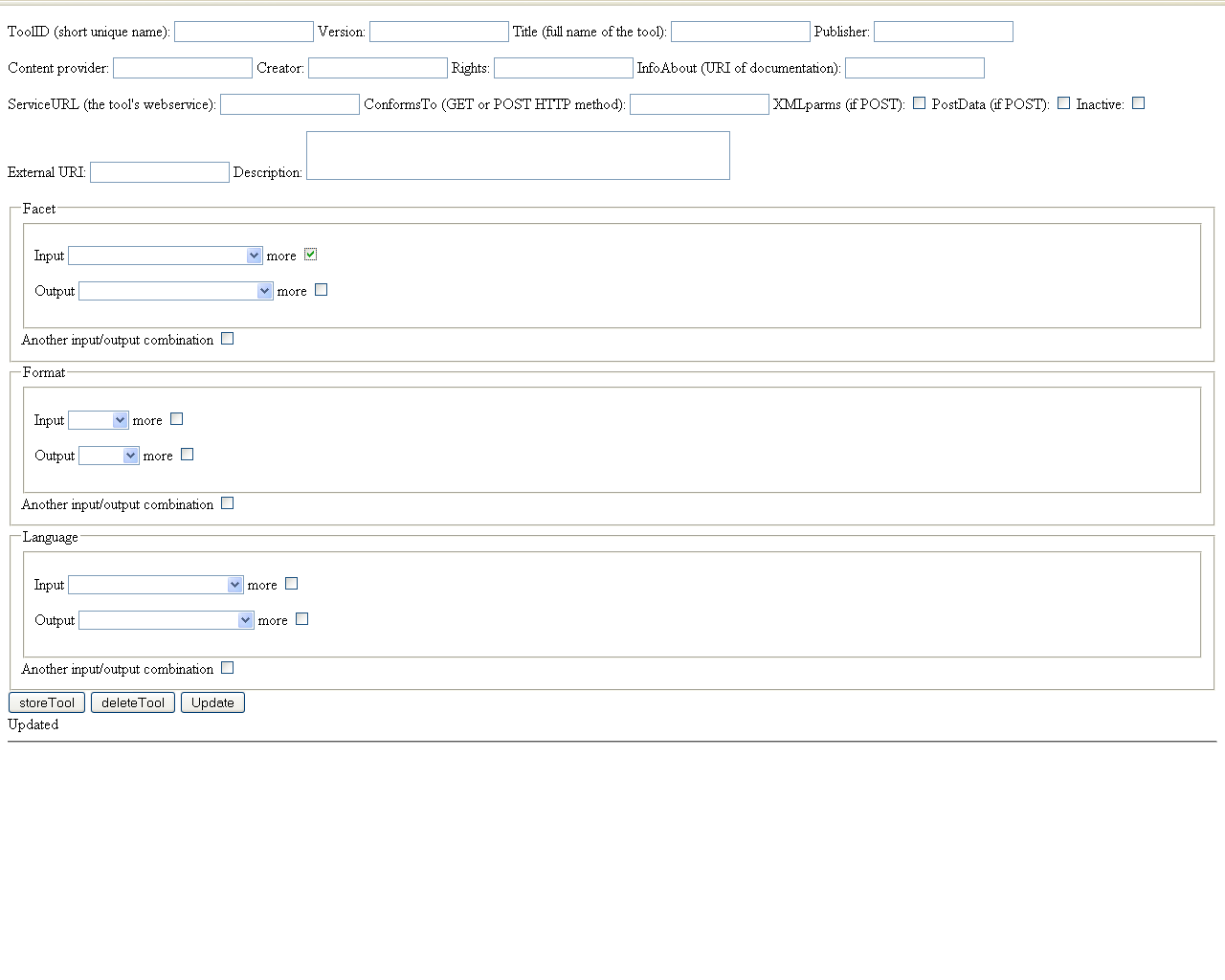
Facet1.1Output=&Format1.1Input=&Format1.1Output=&Language1.1Input=&Language1.1Output=

Den nederste del af formularen er udtrykt i denne parameter:

bsubmit=Update

Parametrenes værdier står tomme, undtagen værdien Update for bsubmit parameteren. Dette svarer til formularens tomme tilstand.

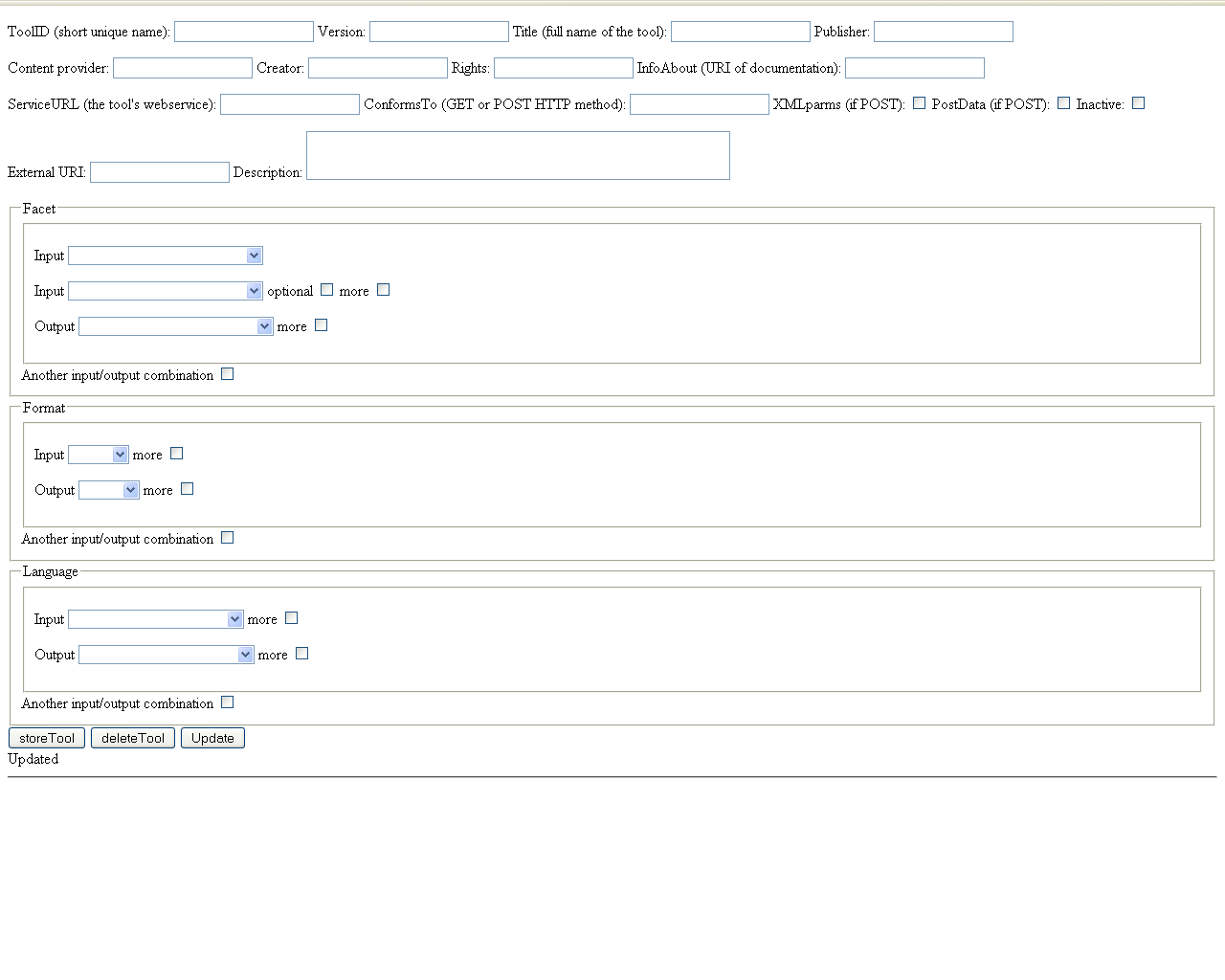
Den midterste del af formularen er dynamisk. Sætter vi et hak i den første tjekboks:



og trykker på Update-knappen, udvides den midterste del af parameterlisten til:

Facet1.1Input=&Facet1.1InputMore=on&Facet1.1Output=&Format1.1Input=&Format1.1Output=&Language1.1Input=&Language1.1Output=

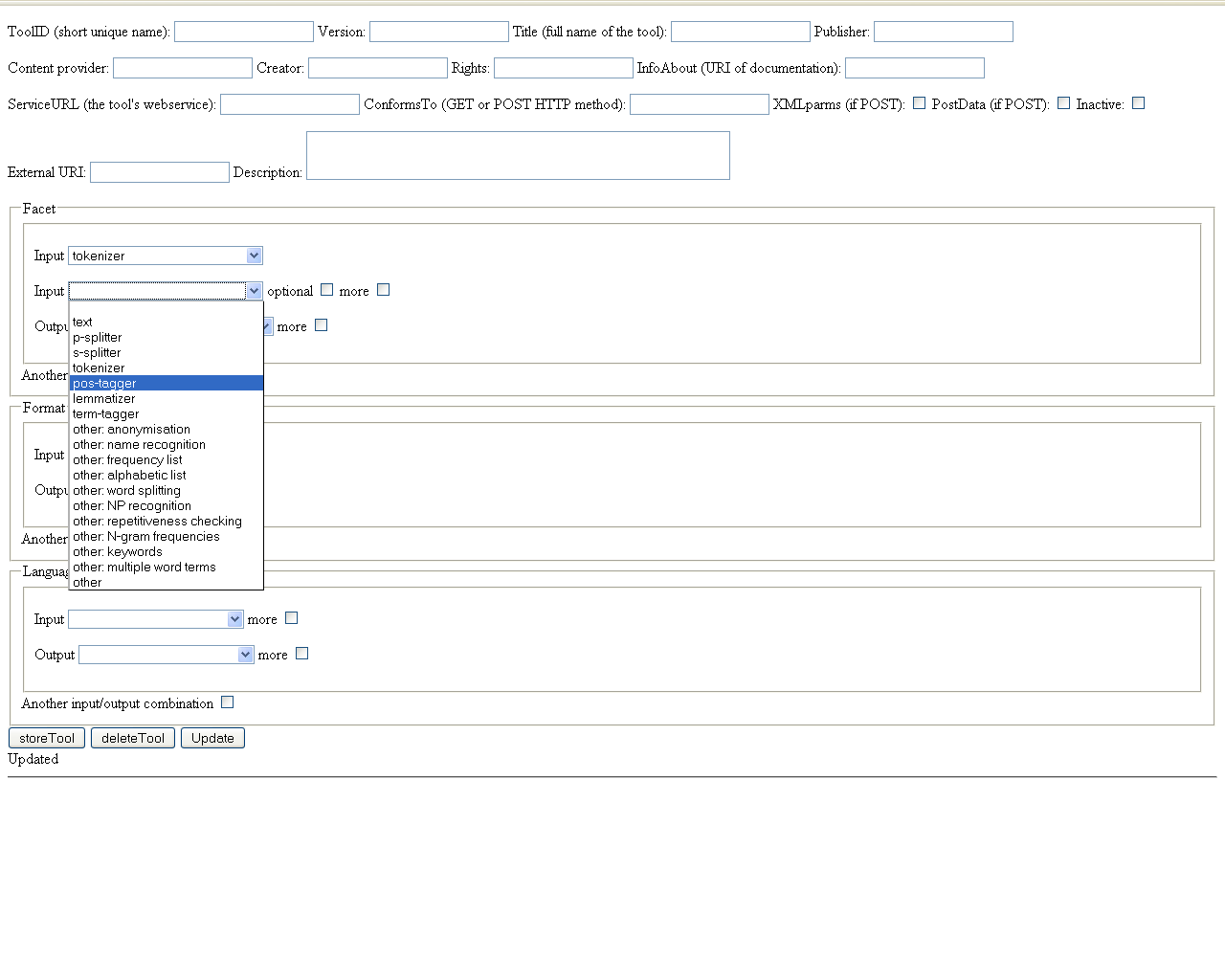
Hakket i *more*-boksen udtrykkes i parameteren Facet1.1InputMore=on. Registreringsmodulet svarer med en ny formular med to nye felter:



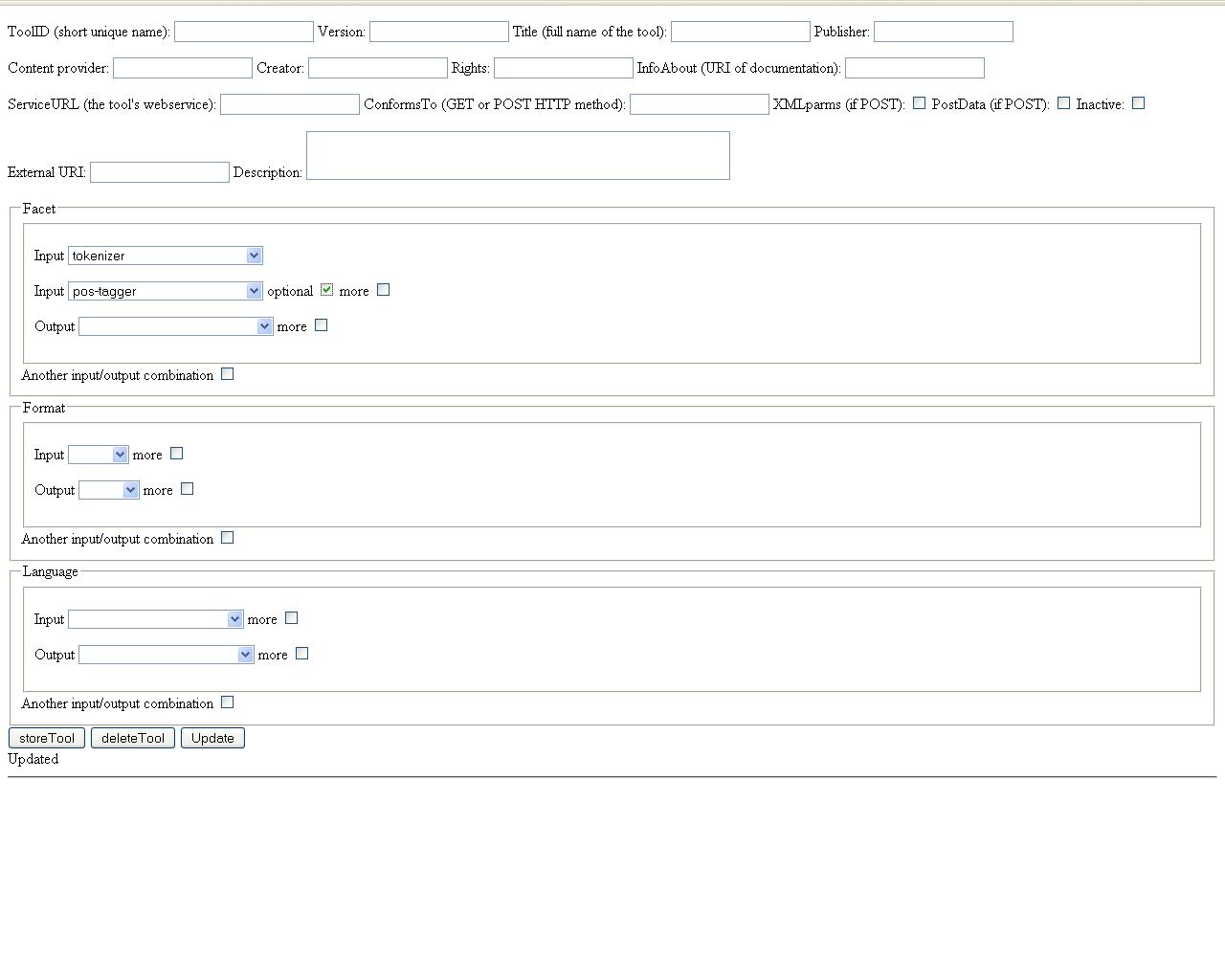
Ny er en ekstra listeboks for facet-input og en tjekboks ved navn *optional*. *more*-boksen som akkompagnerede den første listeboks for facet-input, er nu flyttet til den nye listeboks. Betydningen af de to listebokse er at værktøjet kan tage to forskellige inputs med hver deres facet. Det første input er altid tvunget, men værktøjsudbyderen kan vælge at lade den anden, tredje, osv. input være valgfrie. Her er et eksempel.

Lad os udfylde de to bokse for facet input, og lad os markere den sidste af de to som optionel.

Den ene boks:



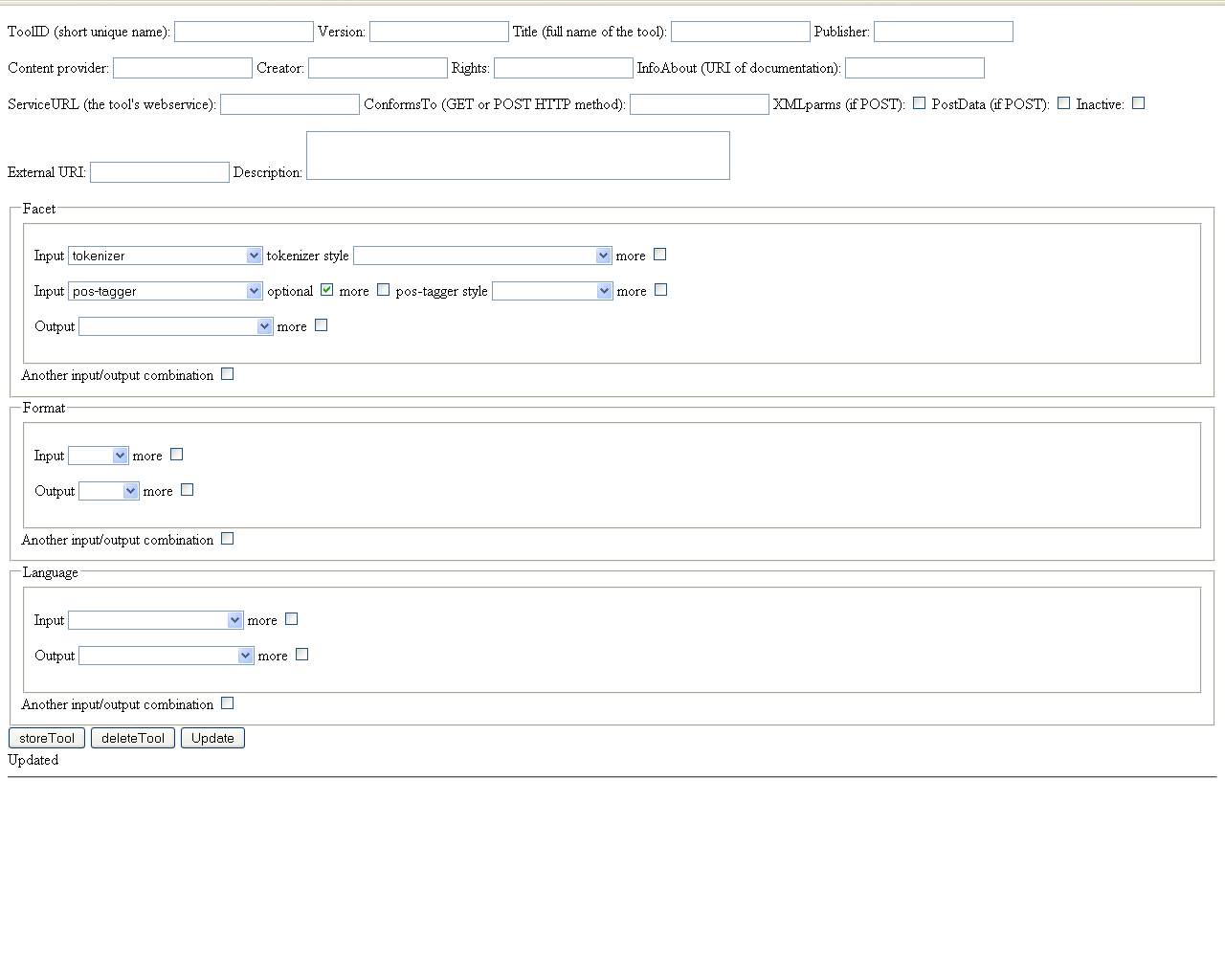
Den anden boks:



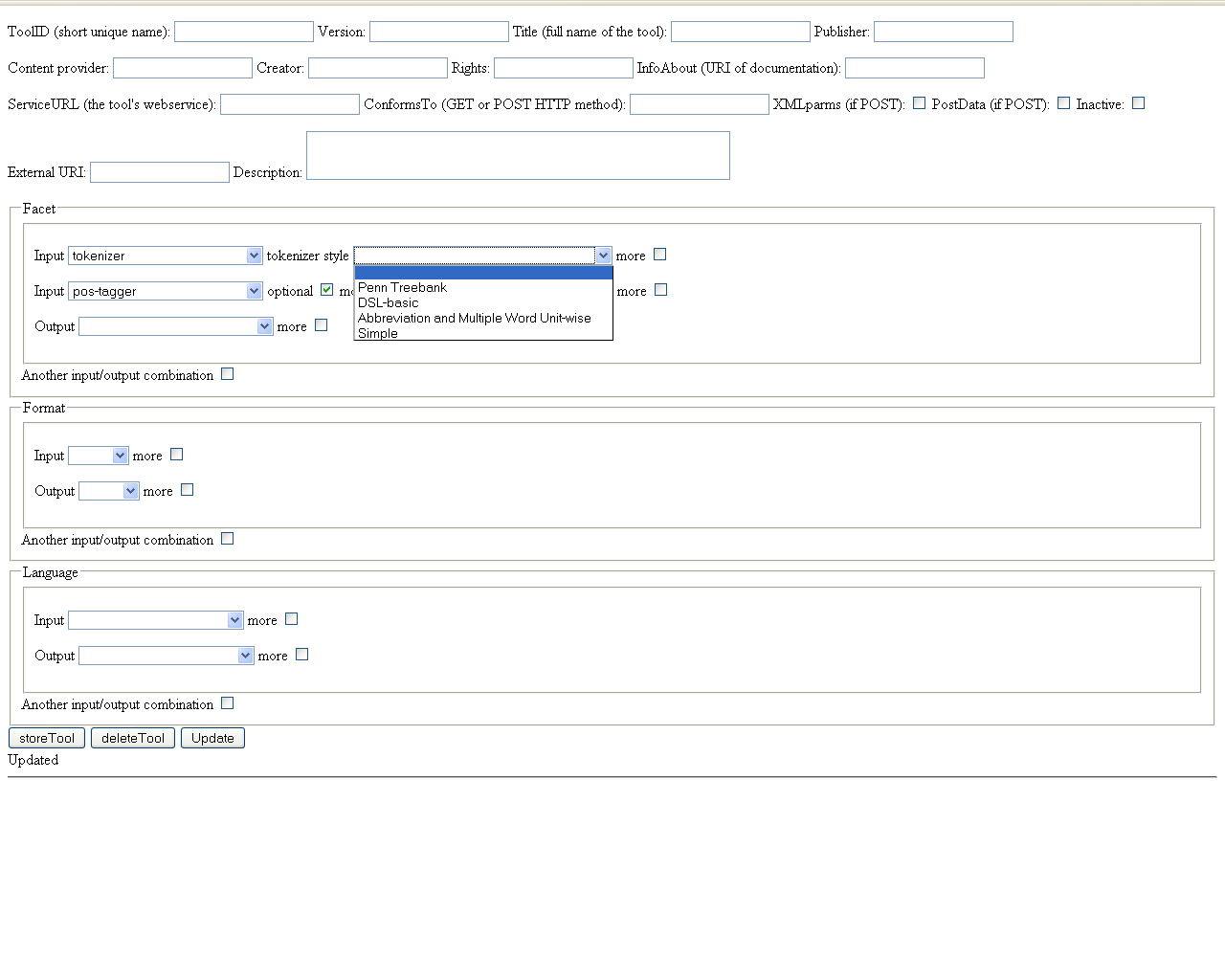
Ved tryk på Update-knappen sendes disse midterdelparametre til registreringsmodulet:

Facet1.1Input=tok&Facet1.1InputMore=on&Facet1.2Input=pos&Facet1.2InputOptional=on&Facet1.1Output=&Format1.1Input=&Format1.1Output=&Language1.1Input=&Language1.1Output=

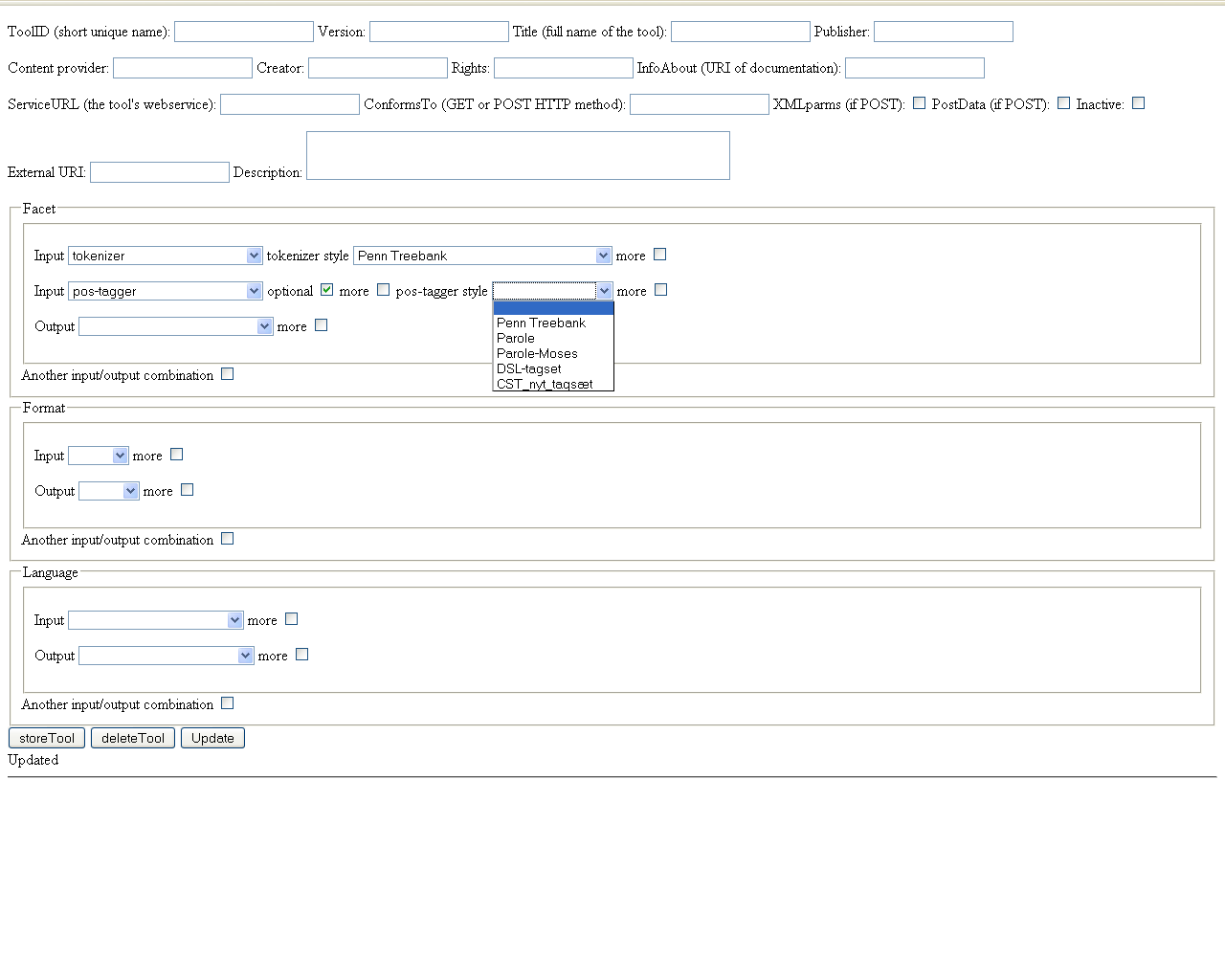
Registreringsmodulet svarer med en formular som indeholder fire nye felter, da det har opdaget at de to inputfacetter begge kan specificeres nærmere. Da der ikke er nogen grund til at begrænse antallet af mulige specificeringer, tilføjes også en *more* tjekboks til hver ny listeboks:



Listeboksen ved navn *tokenizer style* er opstået fordi værdien *tokenizer* blev valgt for den første, ikke optionelle input, og giver mulighed for at vælge mellem et antal forskellige måder at definere et token på:

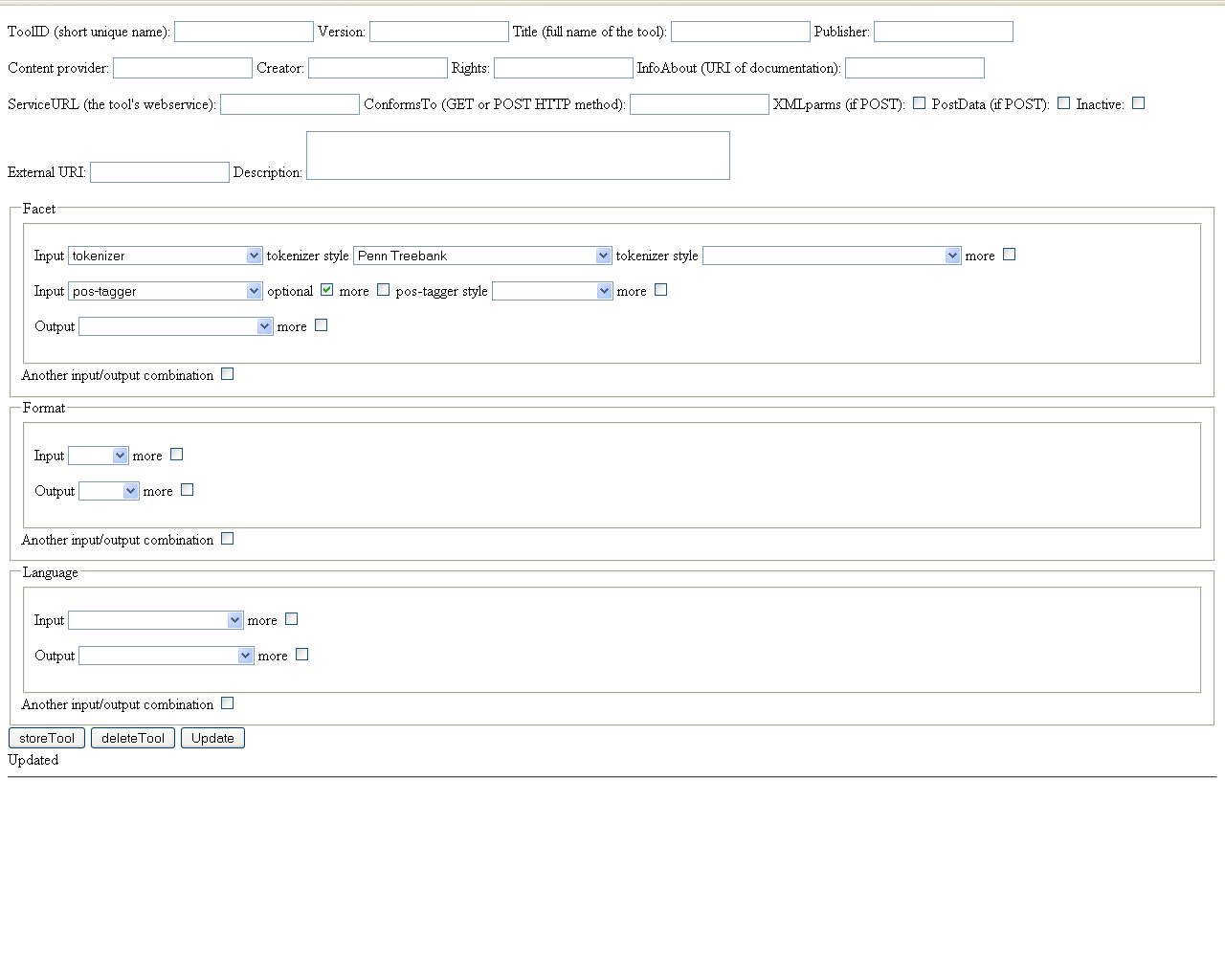


Ligeledes giver pos-tagger style listeboksen mulighed for at vælge mellem et antal tagsets som infrastrukturen understøtter:



Man er ikke forpligtet til at vælge en specificering, det er kun et tilbud som registreringsmodulet giver værktøjsudbyderen. Hvis man undlader at specificere noget der kan specificeres, giver man udtryk for at den specificering ikke er kritisk og at alle mulige specificeringer er tilladelige.

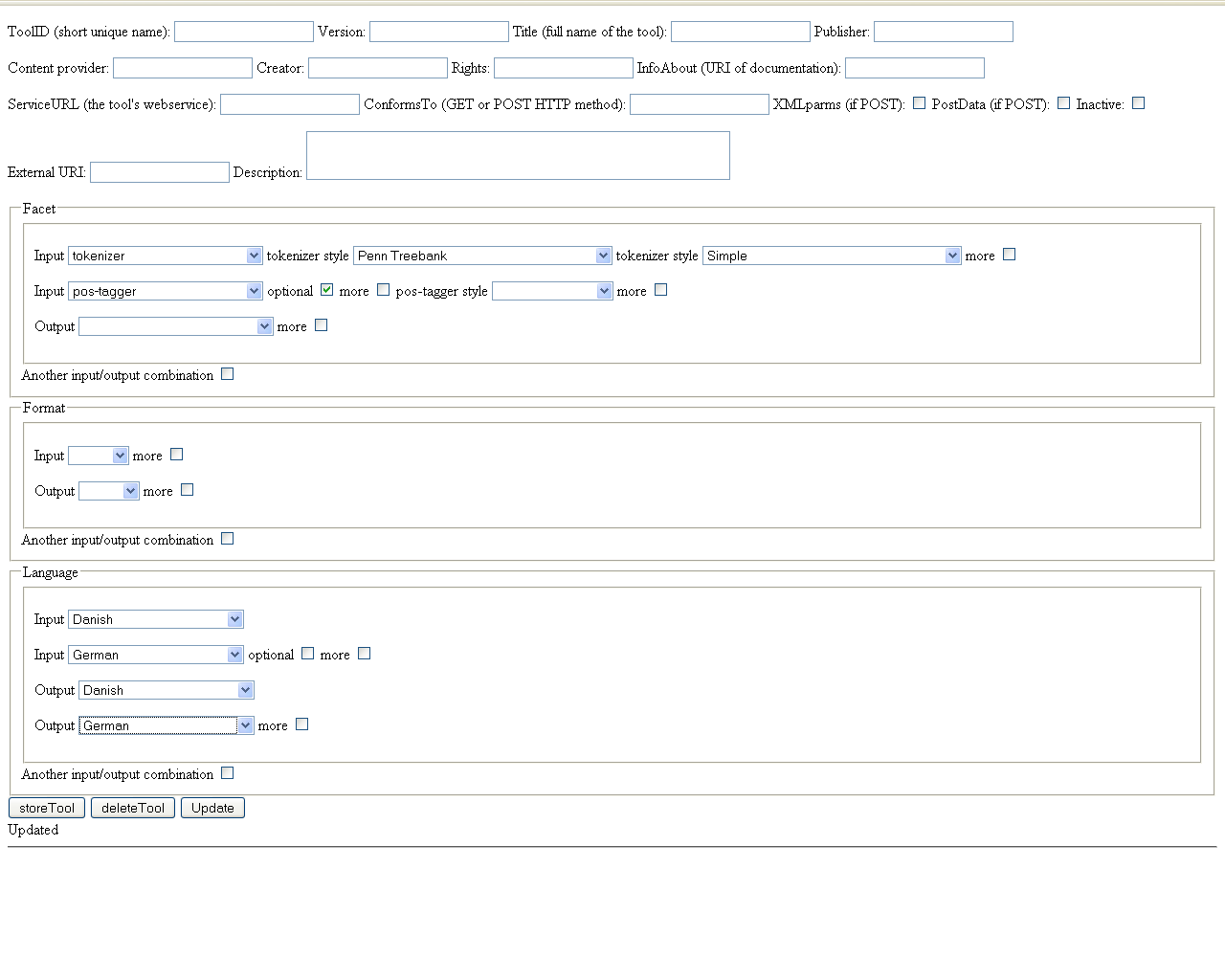
Hvis værktøjsudbyderen vil udtrykke at tokenizer-style enten kan være *Penn Treebank* eller *Simple*, benyttes *more*-tjekboksen til højre for *tokenizer style* listeboksen og bedes registreringsmodulet om en opdateret version af formularen:



Efter at have valgt *Simple* i den nys opståede *tokenizer style* listeboks og efter tryk på Update-knappen, ser parametrene for facet således ud:

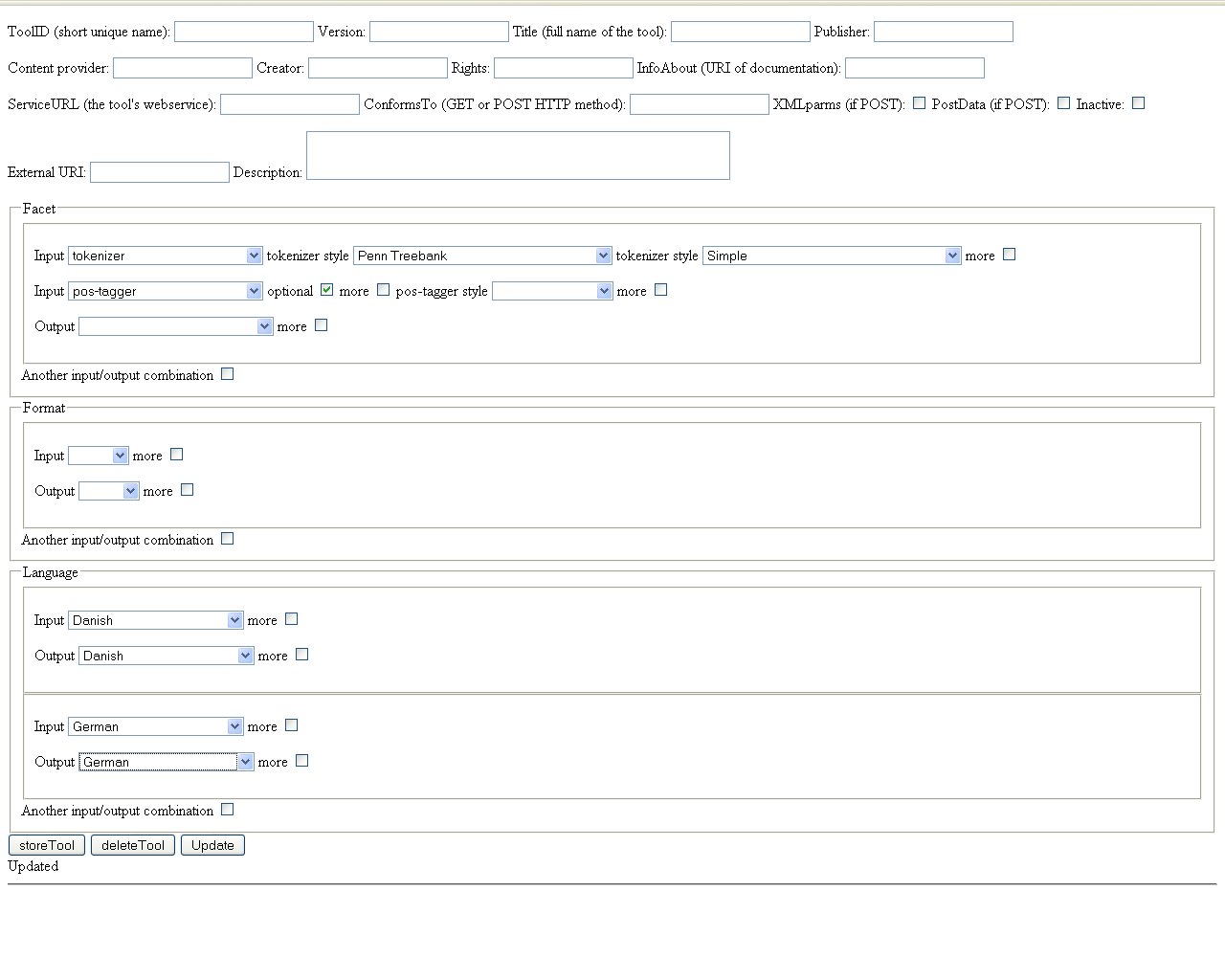
Facet1.1Input=tok&Facet1.1InputMore=on&Facet1.1.1InputSpc=PT&Facet1.1.1InputSpcMore=on&Facet1.1.2InputSpc=simple&Facet1.2Input=pos&Facet1.2InputOptional=on&Facet1.2.1InputSpc=&Facet1.1Output=&Format1.1Input=&Format1.1Output=&Language1.1Input=&Language1.1Output=

Hvis man skal udtrykke at et værktøj kan håndtere både dansk og tysk input og output, er der to meget forskellige muligheder. Dette er den ene mulighed:



Med disse oplysninger påstår værktøjsudbyderen at værktøjet kan tage input som indeholder både dansk og tysk tekst, fx en dansk tekst med tyske citater. Outputtet vil også være en blanding af dansk og tysk.

Dette er den anden mulighed:

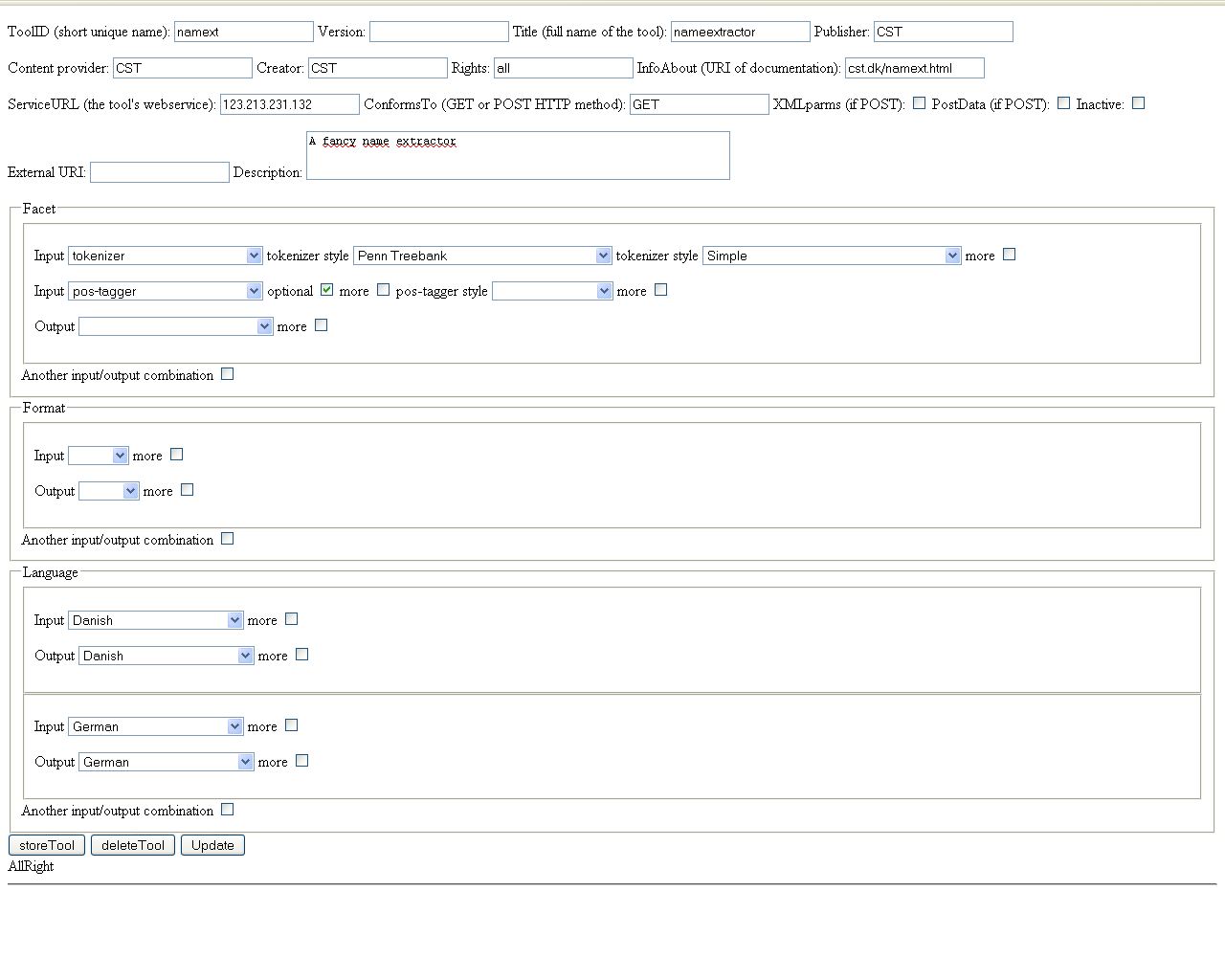


Med disse oplysninger registreres at værktøjet kan tage enten dansk eller tysk input, og at outputtet er dansk hvis inputtet er dansk og tysk hvis inputtet er tysk. Sprogparametrene ser sådan ud:

Language1.1Input=da&Language1.1Output=da&Language1More=on&Language2.1Input=de&Language2.1Output=de

Hvis et værktøj, som her, er registreret med alternative oplysninger, fx enten dansk eller tysk input/output, men ikke både dansk og tysk, bliver valget mellem mulighederne udsat til det tidspunkt hvor en bruger vil behandle en resurse med værktøjet. Hvis resursen er på dansk, bliver værktøjsindstillingerne svarende til dansk input valgt, og *mutatis mutandi* for tysk. Hvis resursen er på fransk, er værktøjet ikke anvendelig og afvises.

Lad os nu vende tilbage til den øverste del af formularen. Vi udfylder felterne og trykker derefter på storeTool-knappen:



Parametrene ser nu sådan ud:

name=namext&vrsion=&fllNm=nameextractor&pblshr=CST&cntPrvdr=CST&crtr=CST&rghts=all&docuri=cst.dk%2Fnamext.html&url=123.213.231.132&cnfrms=GET&xuri=&dscrptn=A+fancy+name+extractor&Facet1.1Input=tok&Facet1.1InputMore=on&Facet1.1.1InputSpc=PT&Facet1.1.1InputSpcMore=on&Facet1.1.2InputSpc=simple&Facet1.2Input=pos&Facet1.2InputOptional=on&Facet1.2.1InputSpc=&Facet1.1Output=&Format1.1Input=&Format1.1Output=&Language1.1Input=da&Language1.1Output=da&Language1More=on&Language2.1Input=de&Language2.1Output=de&bsubmit=storeTool

Værktøjet er nu registreret, hvilket vil sige at to tabeller i registreringsmodulets dataområde er udvidet med følgende oplysninger:

/usr/local/jboss/server/default/data/tools/tooladm.table:

+ ( (ToolID.namext)

(Version.)

(Title.nameextractor)

(ServiceURL."123.213.231.132")

(Publisher.CST)

(ContentProvider.CST)

(Creator.CST)

(Rights.all)

(InfoAbout."cst.dk/namext.html")

(Description."A fancy name extractor")

(ExternalURI.)

(ConformsTo.GET)

(XMLparms.)

(PostData.)

(Inactive.)

)

/usr/local/jboss/server/default/data/tools/toolprop.table:

+ ( namext

. (Facet,(tok^(PT+simple) pos.))

(Language,(da.da)+(de.de))

)

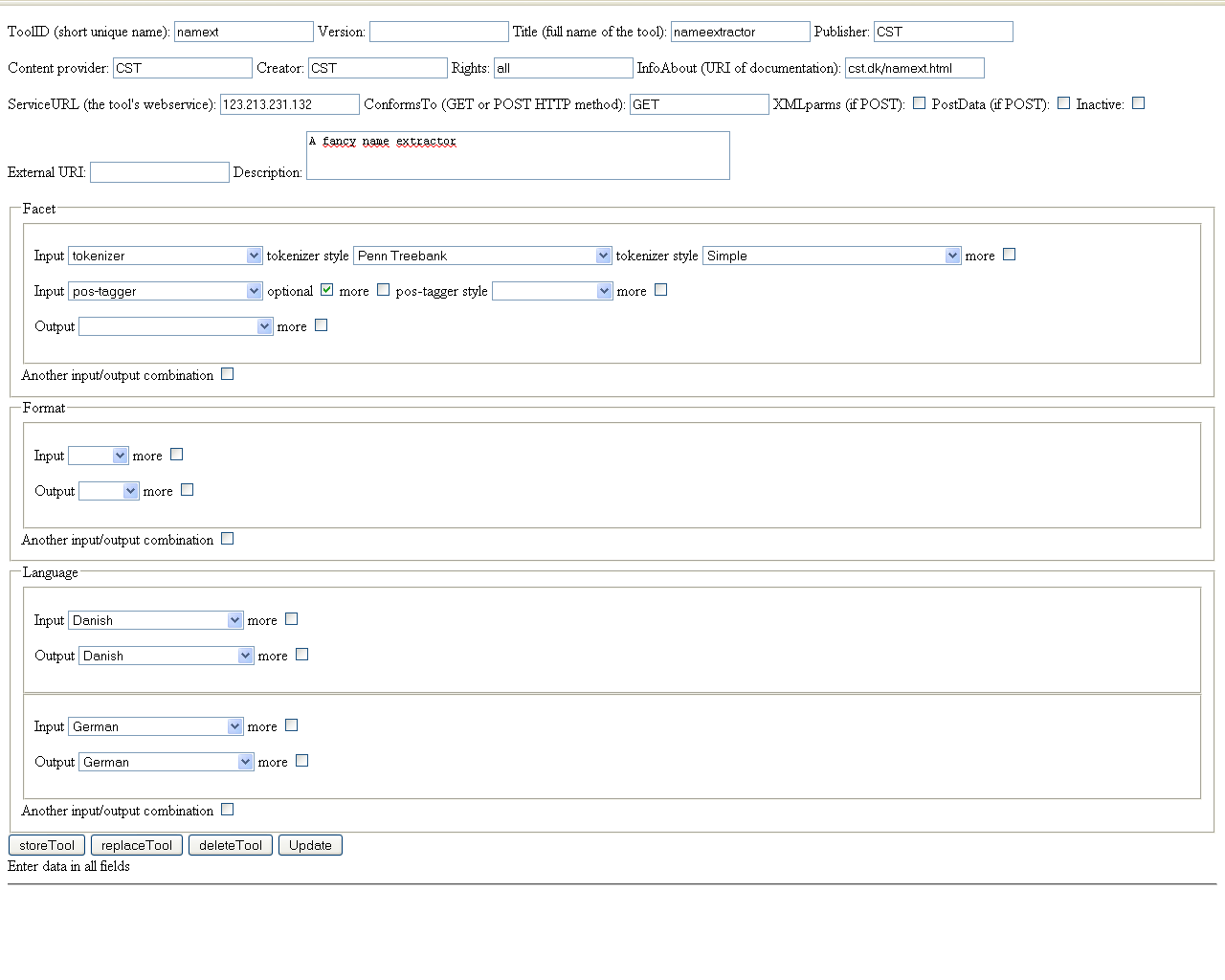
### Rettelse af oplysninger

Efter at have registreret et værktøj har værktøjsudbyderen ansvar for at ajourføre de registrerede oplysninger når der sker en ændring i værktøjet. Derfor er der i registreringsmodulet mulighed for at rette, slette og tilføje oplysninger.

I eksemplet har vi glemt at specificere output-facetten, og den vil vi nu gerne tilføje. For at se hvilke oplysninger der er registreret for et værktøj, skal *ToolID*-parameteren være udfylt. Så vi kunne starte med denne URI:

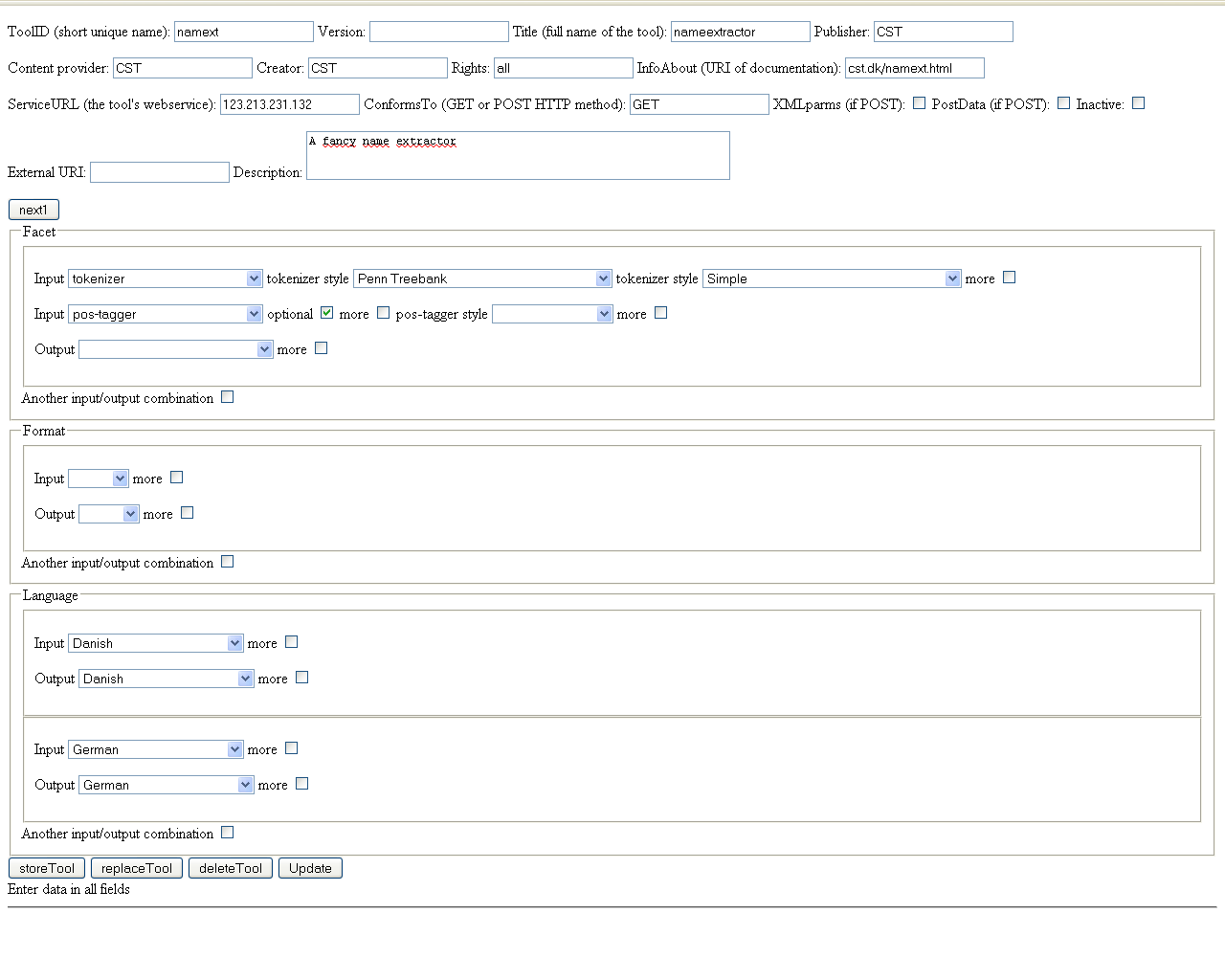
http://infra.clarin.dk/tools/register?name=namext

Registreringsmodulet svarer med registreringsformularen med alle oplysninger som kendes:

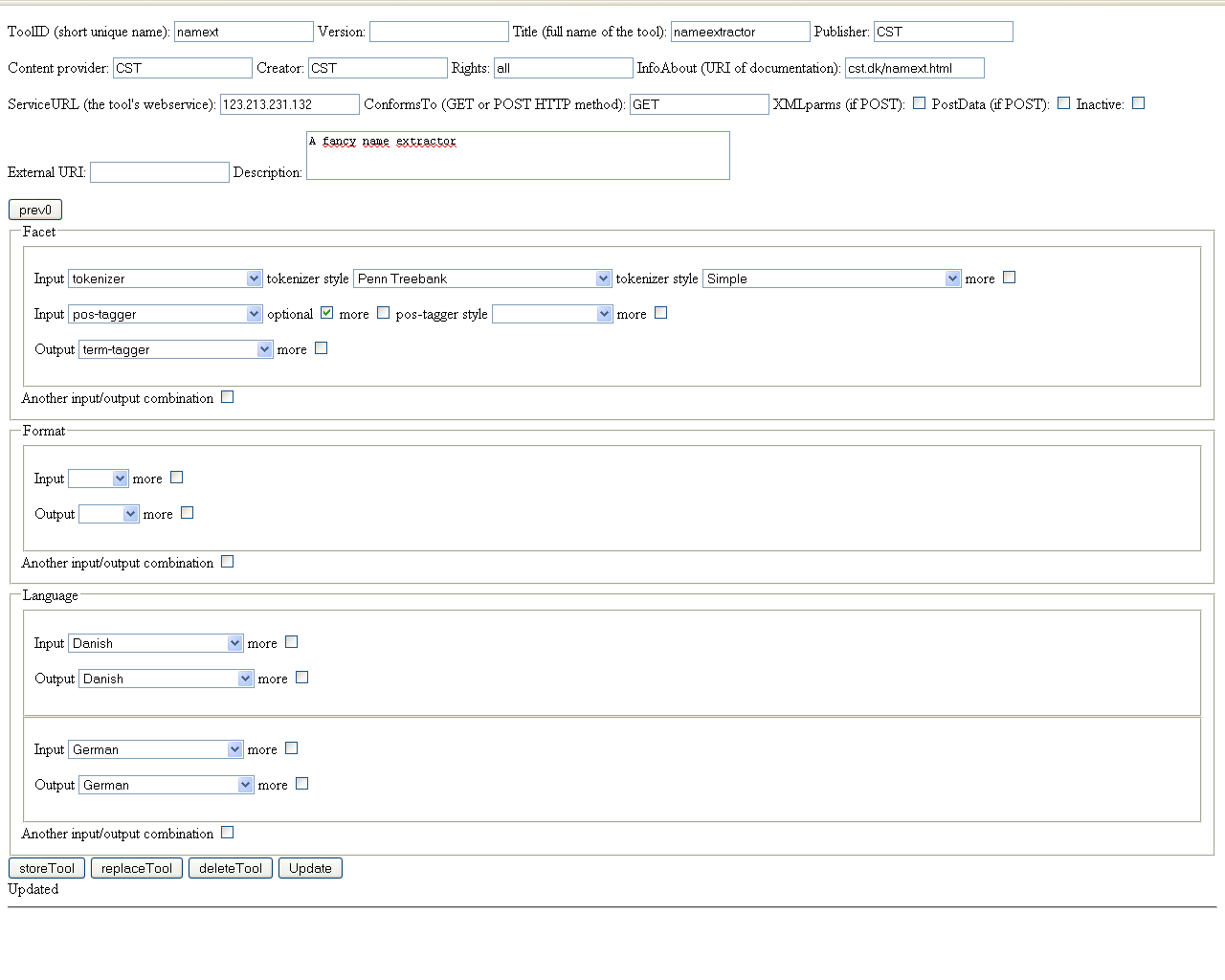


Nyt er knappen *replaceTool*. Hvis vi vælger værdien *term-tagger* for facet-outputtet og trykker på *replaceTool*, bliver feltet for facet-output ændret fra at være et tomt felt til at være et felt med værdien *term-tagger*. Trykker vi derimod på knappen *storeTool*, skaber vi en ny inkarnation af værktøjet som er identisk med den første, bortset fra a facet-outputtet har værdien *term-tagger*.

Hver inkarnation af et værktøj kan bedst betragtes som et selvstændigt værktøj som tilfældigvis har de samme søgbare metadata som de andre inkarnationer. Hvis vi var kommet til at trykke på *storeTool*, og dermed havde skabt en ny inkarnation af værktøjet, ser formularen således ud når man får vist værktøjets oplysninger:



Formularen viser de oprindelige oplysninger, men der er også en ny knap: *next1*. Når den bliver aktiveret, bladrer man til oplysningerne for den næste inkarnation. Som forventet er den næste inkarnation magen til den første, bortset fra at outputfacetten er udfyldt med værdien *term-tagger*:



Parametrene hvormed denne side er produceret ser sådan ud:

name=namext&vrsion=&fllNm=nameextractor&pblshr=CST&cntPrvdr=CST&crtr=CST&rghts=all&docuri=cst.dk%2Fnamext.html&url=123.213.231.132&cnfrms=GET&xuri=&dscrptn=A+fancy+name+extractor&**bsubmit=next1**&Facet1.1Input=tok&Facet1.1InputMore=on&Facet1.1.1InputSpc=PT&Facet1.1.1InputSpcMore=on&Facet1.1.2InputSpc=simple&Facet1.2Input=pos&Facet1.2InputOptional=on&Facet1.2.1InputSpc=&Facet1.1Output=&Format1.1Input=&Format1.1Output=&Language1.1Input=da&Language1.1Output=da&Language1More=on&Language2.1Input=de&Language2.1Output=de&**nth=0**

Man bør lægge mærke til to parametre der fortæller noget om hvilken inkarnation af værktøjet vises: bsubmit=next1 og nth=0. Den første af de to fortæller at formularen er skabt fordi submit-knappen med navnet *next1* blev presset. 1-tallet blev brugt som indeks i rækken af inkarnationer. 0 er den laveste indeks, så bsubmit=next1 betyder at den anden inkarnation vises. Parameteren nth=0 fortæller at hvis det ikke var fordi *next1*-knappen var blevet presset, så ville den første inkarnation været blevet vist. Det vil sige at nth-parameteren bliver ignoreret hvis enten prev*N* eller next*N*-knappen bliver aktiveret (*N* et positivt tal eller nul). Det er da også bemærkelsesværdigt at notere at trækoplysningerne i parametrene svarer til den side vi lige er kommet fra, og ikke til den side der vises!

Man kan fra URI’en nemt styre hvilken inkarnation vises. Således viser også

http://infra.clarin.dk/tools/register?name=namext&nth=1

den anden inkarnation af værktøjet.

### Sletning af værktøjsoplysninger eller inaktivering af et værktøj

Et værktøj kan slettes ved at sende en parameter bsubmit=deleteTool til registreringsmodulet. For hver gang denne parameter bliver sendt, slettes én inkarnation, angivet af nth-parameteren. Værktøjets søgbare metadata kan ikke slettes. Man har altid mulighed for at gøre et værktøj inaktiv.

Hvis ikke hele inkarnationen, men kun et eller flere felter i en inkarnation skal slettes, kan man nøjes med at sende en bsubmit=replaceTool parameter sammen med alle ønskede parametre for den givne (nth-parameter) inkarnation. Hvis alle felter i en inkarnation er tomme, slettes inkarnationen.

### Værktøjsregistrering i detaljer: Backus–Naur Form

Den følgende BNF definition afslører ikke alle muligheder og begrænsninger. Derfor følgende tilføjelser:

1. Parametrenes rækkefølge er uden betydning
2. Et parameternavn optræder højst én gang
3. Visse udtryk er ikke eksplicit defineret: NCName, string, uri, url, non-negative integer, natural number.
4. I rækken af numre (natural number) må der ikke opstå huller: det første nummer er 1, det næste 2, osv.

<registerURI> = <endpoint> '?' <parms>

<endpoint> = 'http://infra.clarin.dk/tools/register' | 'http://infra.clarin.dk/tools/register'

<parms> = <staticparms> [ '&' <dynamicparms> ] [ '&' <nth> ] [ '&' <bsubmit> ]

<staticparms> = 'name=' NCName

[ '&vrsion=' string

'&fllNm=' string

'&pblshr=' string

'&cntPrvdr=' string

'&crtr=' string

'&rghts=' string

'&docuri=' uri

'&url=' url

'&cnfrms=' ( 'GET'

| 'POST' [ '&prmsxml=on' ] [ '&pstdt=on' ]

)

'&xuri=' uri

'&dscrptn=' string

['&nctv=on']

]

<dynamicparms> = <feature> { <feature }

<nth> = 'nth=' integernumber

<bsubmit> = 'bsubmit=' ( ( 'prev' | 'next' ) non-negative integer

| 'storeTool'

| 'deleteTool'

| 'Update'

| 'replaceTool'

)

<feature> = <featurealt> { <featurealt> }

<featurealt> = <featurename> natural number '.' <ioalt>

<featurename> = 'facet' | 'format' | 'lang' (This list can be extended!)

<ioalt> = natural number ( <io> | '.' <iospcalt> )

<io> = ( 'Input' | 'Output' ) ( '=' string | 'More=on' | 'Optional=on' )

<iospcalt> = natural number <iospc>

<iospc> = ( 'Input' | 'Output' ) 'Spc' ( '=' string | 'More=on' )

Eksempel (linjeskift er indsat for læsbarhedens skyld):

http://infra.clarin.dk/tools/register

?name=namext

&vrsion=

&fllNm=nameextractor

&pblshr=CST

&cntPrvdr=CST

&crtr=CST

&rghts=public

&docuri=cst.dk%2Fnamext.html

&url=123.456.789.12

&cnfrms=POST

&prmsxml=on

&pstdt=on

&nctv=on

&xuri=http%3A%2F%2Fida.hum.ku.dk%2Ftools%2F

&dscrptn=jhgjkl+hgjklgh

&facet1.1Input=txt

&facet1.1InputMore=on

&facet1.2Input=tok

&facet1.2InputOptional=on

&facet1.2.1InputSpc=DSLb

&facet1.2.1InputSpcMore=on

&facet1.2.2InputSpc=PT

&facet1.1Output=pos

&facet1.1OutputMore=on

&facet1.1.1OutputSpc=

&facet1.2Output=seg

&format1.1Input=flat

&format1.1.1InputSpc=

&format1.1Output=xm

&format1.1.1OutputSpc=

&language1.1Input=da

&language1.1Output=fr

&nth=0

&bsubmit=Update

## Aktivering

Aktivering af værktøjer sker efter en interaktiv proces med brugeren. Når brugeren kommer til værktøjsmodulet, har han eller hun allerede valgt en eller flere resurser som brugeren har autorisation til. Værktøjsmodulet modtager handles til disse resurser og kopierer resurserne til et såkaldt stagingområde. Samtidigt bliver resursernes metadata analyseret for at finde de oplysninger hvormed workflowplanneren kan udregne hvilke værktøjer der kan optræde som de(t) første værktøj(er) i det kommende workflow. Ved den senere afvikling af opgaverne i workflowet får værktøjerne udleveret data fra stagingområdet, og derfor skal værktøjerne ikke selv håndtere de følsomme brugerhandles. Værktøjsmodulet er først og fremmest tænkt til anvendelse på resurser der er lagret i infrastrukturens repositorie, men kan senere udvides til også at håndtere data udefra. I det sidste tilfælde skal brugeren selv give de fornødne oplysninger om dataene til værktøjsmodulet.

Når dataene er kopieret til stagingområdet, bliver brugeren inviteret til at karakterisere det ønskede output, hvorefter workflowplanneren beregner alle workflows der fører fra input til det ønskede output. Workflowplanneren håndterer de oplysninger om værktøjerne som værktøjsudbyderne har givet ved registrering af værktøjerne. Andre værktøjer end de registrerede kommer ikke i betragtning. Hvis der ikke findes et workflow der forbinder input med det ønskede output, får brugeren en meddelelse herom. Hvis der er mere end ét workflow, får brugeren mulighed for at vælge et workflow blandt de foreslåede workflows.

Når et workflow er valgt, kan brugeren starte afviklingen af alle delopgaver i workflowet. Når hele workflowet er afviklet, får brugeren en e-mail med et link til resultatet, som ligger gemt i stagingområdet, sammen med alle mellemresultater fra de forskellige trin i forløbet. Brugeren kan hente resultatet indenfor en bestemt tidsperiode, fx 24 timer eller to måneder. (Periodens længde er ikke besluttet endnu.)

Værktøjsmodulet tager sig ikke af deponering af resultater fra værktøjer, men forbereder deponeringen ved at udfylde metadata for resultaterne i så vidt muligt omfang. Disse metadata inkluderer en kortfattet specificering af de værktøjer og resurser der ligger til grund for den nye resurse, og i øvrigt oplysninger om de træk som værktøjsmodulet kan regne sig frem til på basis af de oplysninger værktøjsmodulet har om inputresurserne og de værktøjer der har været i spil. Normalt vil brugerens specificering af outputtet være en delmængde af disse træk.

### Aktivering i pseudokode

*/\* Before the tools module comes into action, the user already*

*has selected one or more resources. \*/*

Copy the resources to the staging area of the Tools-module.

Analyse the selected resources and extract information pertaining to all features that the Tools module can handle. */\* e.g. type of resource (facet), language and format.\*/*

Ask the user to specify one or more features that characterize the output.

Compute which workflows consisting of zero or more tools produce the sought for goal from the given input.

Let the user choose one of the workflows. Give the user a reason if the set goal cannot be reached with the registered tools.

If the chosen path is not completely fixed already, ask for supplemental input from the user.

Ask the user whether to submit the workflow

**If** user answers confirmative

**While** goal not fulfilled

From the workflow, select all tools for which input is available in the staging

area.

Remove the selected tools from the workflow.

Activate the selected tools.

*/\*The tool gains access to the input, either by receiving the data in a POST*

*or by retrieving the data using a GET parameter as address.\*/*

**When** a tool replies with its result

Copy the result to staging area of the Tools-module

**If** all went well

Add metadata to the result based on tools data and input

Send user an email with a link (URL) to the output.

*/\*User fetches output from staging area, User supplies metadata and*

*User deposits the new resource using the deposit service.\*/*

**Else**

Send user an email telling that something went wrong

### Aktivering: URL

Hvis tools/create kaldes med HTTP GET-metoden, gives inputdata som item-id’er i requestparametrene:

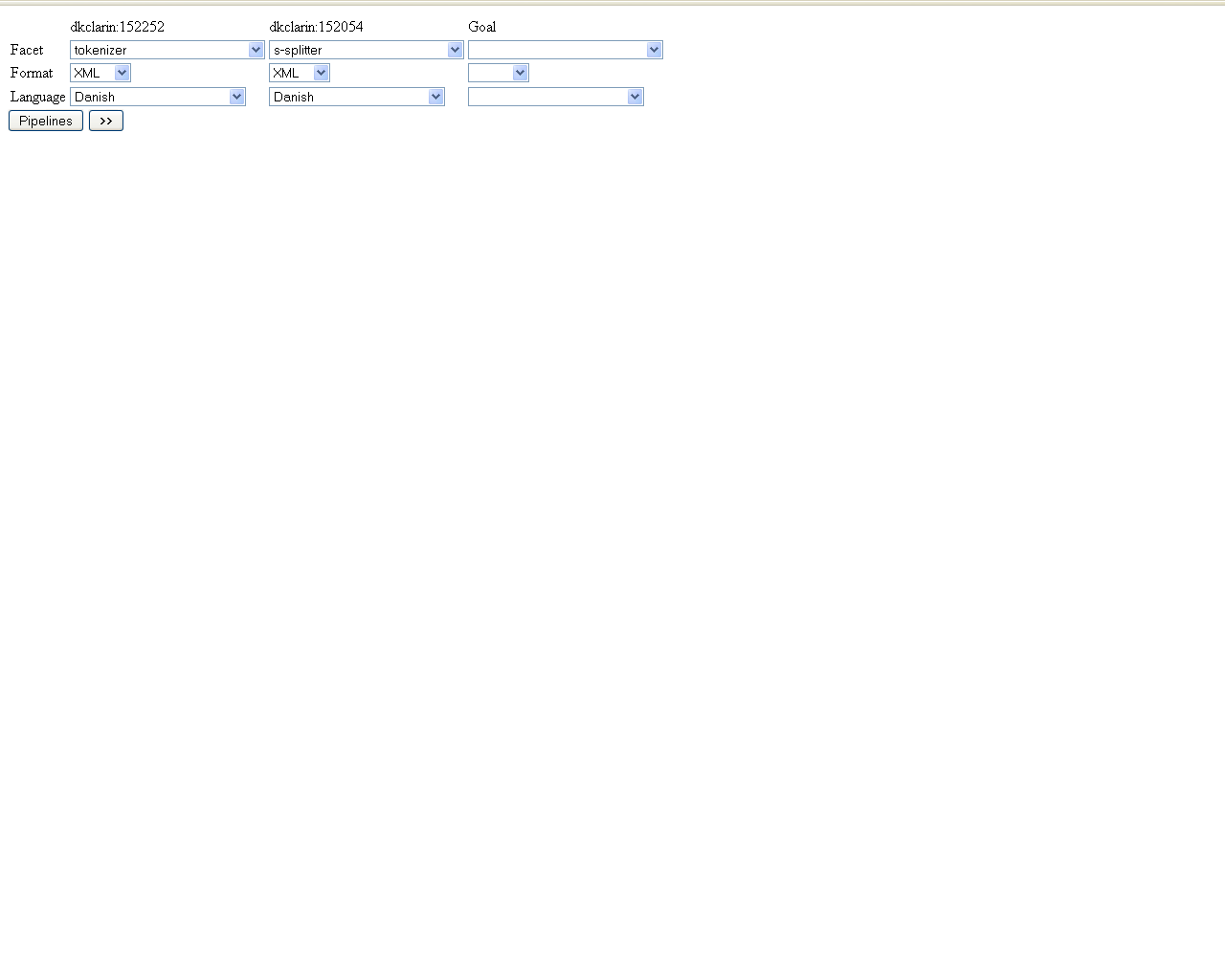
http://infra.clarin.dk/tools/create?item=dkclarin%3A152252&item=dkclarin%3A152172

Disse parametre indikerer at Tools skal arbejde med disse to resurser

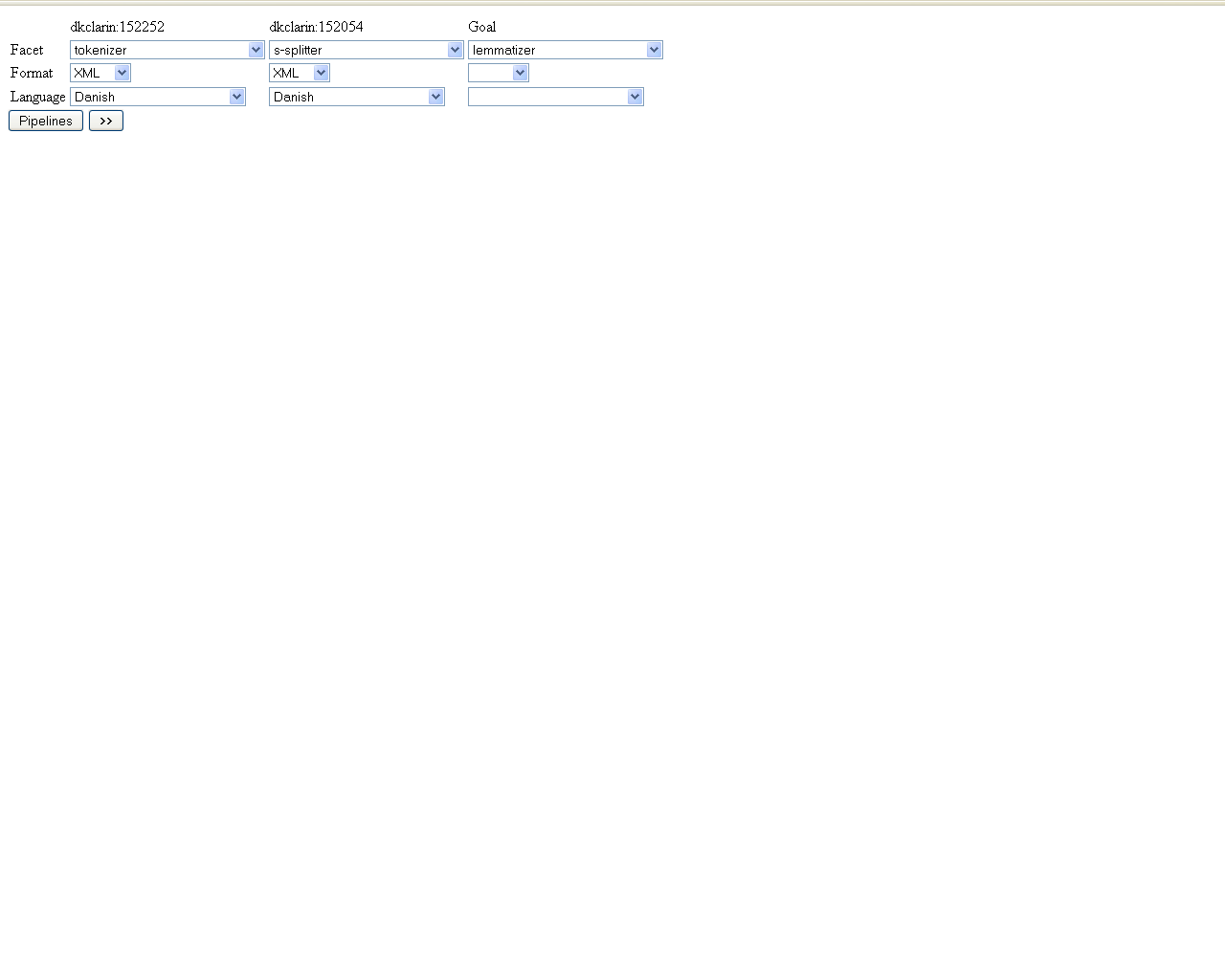
dkclarin:152252

dkclarin:152054

Hvis ikke flere end disse oplysninger er givet, svarer Tools med en XHTML formular som beder brugeren om at specificere målet (eng. *goal*) med brugen af Tools-modulet.



Brugeren kan udfylde et eller flere *goal*-felter, fx værdien *lemmatizer* i facet-feltet (det øverste af felterne).



Hvis man således har specificeret at målet skal være en annotation med lemmaer (fork. lem) og trykker på den højre af de to knapper (>>), bliver følgende parametre sendt til http://infra.clarin.dk/tools/create:

Item="dkclarin%3A152252"+(facet.tok)+(format.xm)+(lang.da)&Item="dkclarin%3A152054"+(facet.seg)+(format.xm)+(lang.da)&Ifacet=tok&Ifacet=seg&Ofacet=lem&Iformat=xm&Iformat=xm&Oformat=&Ilang=da&Ilang=da&Olang=&bsubmit=>>

I mere læsbar form:

Item="dkclarin%3A152252"+(facet.tok)+(format.xm)+(lang.da)

Item="dkclarin%3A152054"+(facet.seg)+(format.xm)+(lang.da)

Ifacet=tok

Ifacet=seg

Ofacet=lem

Iformat=xm

Iformat=xm

Oformat=

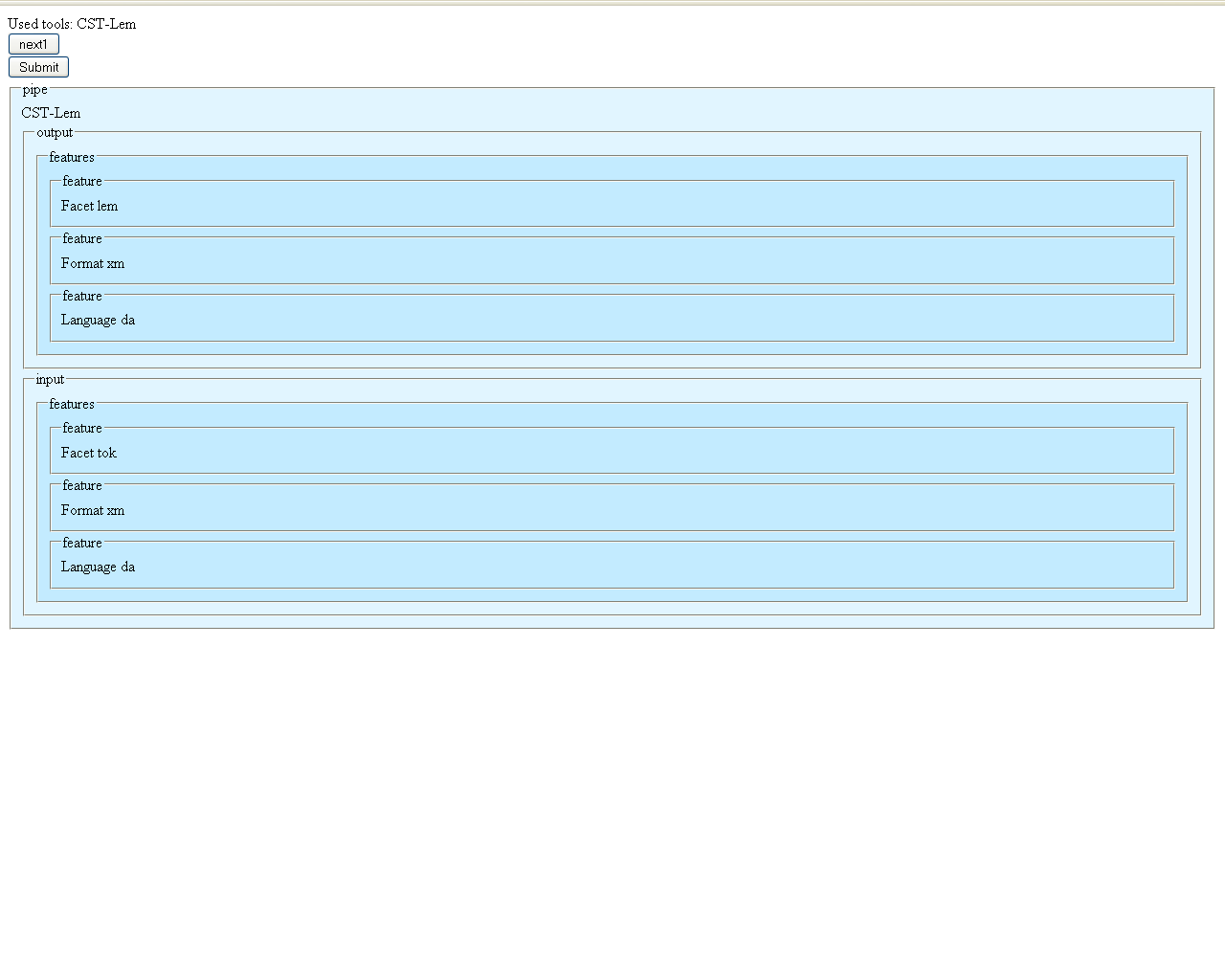
Ilang=da

Ilang=da

Olang=

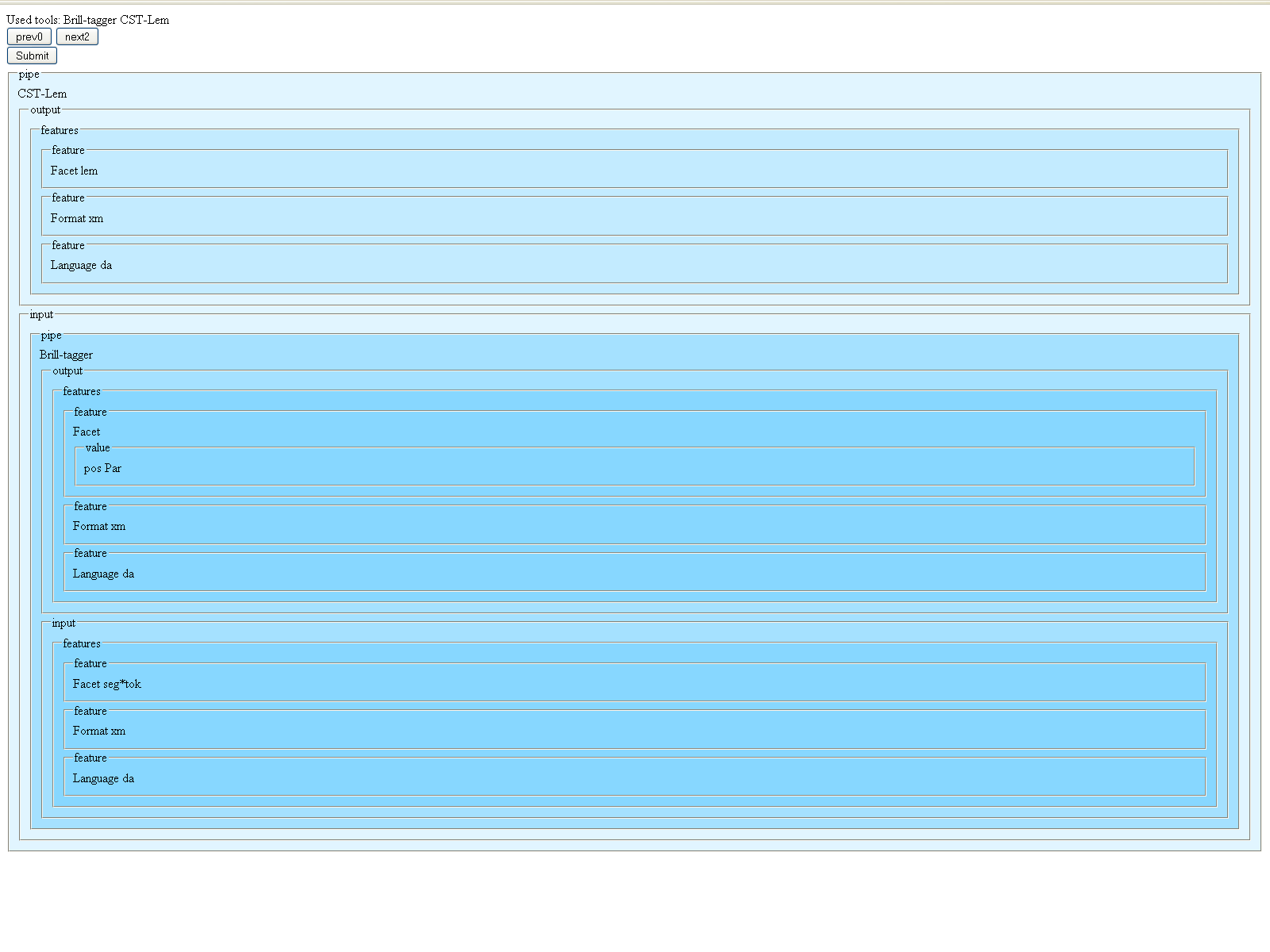
bsubmit=>>

Tools regner på forespørgslen og finder ud af at der findes flere muligheder for at opnå målet. Svaret bliver sendt tilbage som følgende XHTML-dokument indeholdende en formular:



Den foreslåede workflow anvender et enkelt værktøj, CST-lem. Hvis der er flere mulige workflows, som i dette eksempel, kan brugeren trykke på *next*- og *prev*-knapper for at få præsenteret den næste hhv. forrige workflow. Hvis der er uoverskuelig mange mulige workflows, kan brugeren vælge at gå tilbage til den forrige formular og specificere flere træk som karakteriserer målet. En anden måde at formindske antallet af workflows på er at specificere et værktøj som skal være en del af workflowet. Udover navigationsknapperne og *submit*-knappen er der ikke noget brugeren har mulighed for at reagere på, hvilket skyldes at Tools har kunnet regne ud hvad værdierne for alle ikke udfyldte felter må være: formatet og sproget af outputtet.

Det viste workflow udnytter slet ikke at der er et input der fortæller hvor sætningsgrænserne går i teksten. CST-lem nøjes med at tage tekstens tokens som input. Det næste workflow er:



Dette workflow beskriver aktiveringen af to værktøjer: først Brill-tagger og derefter CST-lem.

Når brugeren trykker på *Submit*, sendes følgende parametre til http://infra.clarin.dk/tools/create

bsubmit=Submit&nth=0&Item="dkclarin%3A152252"+(facet.tok)+(format.xm)+(lang.da)&Item="dkclarin%3A152054"+(facet.seg)+(format.xm)+(lang.da)&Ifacet=tok&Ifacet=seg&Ofacet=pos&Iformat=xm&Iformat=xm&Ilang=da&Ilang=da

Toolsmodulet kreerer to jobs som skal afvikles efter hinanden. Internt ser stakken af jobs sådan ud:

( 281

. jobby1

. RePos15 RePos16

. "https://ida.hum.ku.dk/CST-tagger/"

. ( I

. (facet.(seg.F.RePos16)+(tok.F.RePos15))

+ (format.xm.0)

+ (lang.da.0)

)

+ (O.(facet.pos.Par)+(format.xm.0)+(lang.da.0))

+ (base."https://infra.clarin.dk/toolsdata/")

+ (job.281-jobby1)

+ (post2."https://infra.clarin.dk/tools/upload")

. "FilnavnetXYZ-281-jobby1.xml"

)

( 281

. jobby2

. (jobby1,pending) RePos15

. "https://ida.hum.ku.dk/CST-lemma/"

. ( I

. ( facet

. (pos.(F."FilnavnetXYZ-281-jobby1.xml")+(Par.0))

+ (tok.F.RePos15)

)

+ (lang.da.0)

)

+ (O.(facet.lem.0)+(format.xm.0)+(lang.da.0))

+ (base."https://infra.clarin.dk/toolsdata/")

+ (job.281-jobby2)

+ (post2."https://infra.clarin.dk/tools/upload")

. "FilnavnetXYZ-281-jobby2.xml"

)

De to jobs holdes sammen ved løbenummeret, *281* i dette tilfælde. De to jobs er kaldt *jobby1* og *jobby2*. Jobby1 tager to inputs som er kopieret fra repositoriet til staginområdet, *RePos15* og *RePos16*. Brill-taggerens service-URL er *http://ida.hum.ku.dk/CST-tagger/*. Input- og outputparametrene som skal sendes til Brill-taggeren er noteret i et neutral format, og konverteres enten til request-parametre eller til en XML-struktur, afhængig af værktøjets ønske. *I*-parametrene beskriver input, *O*-parametrene output. Den sidste linje i det første job fortæller under hvilket navn outputtet fra Brill-taggeren vil blive gemt i stagingområdet. Det andet job er opbygget på samme måde som det første job. Som input tager CST-lemma samme *RePos15*, men også output fra det første job, og derfor er den markeret som *pending*.

Infrastrukturen sender først en forespørgsel til værktøjet Brill-tagger. Når der kommer resultat fra Brill-taggeren, undersøger Tools-modulet hvilke jobs ikke behøver vente mere på deres *pending* inputs. I dette tilfælde er det CST-lemma. Dette job bliver nu også afviklet. Hver gang et job er blevet afviklet, bliver det fjernet fra jobstakken. Hvis et job fejler, bliver ikke kun det fejlende job, men også de jobs som er afhængige af det fejlede job, slettet.

De to resultater, *FilnavnetXYZ-281-jobby1.xml* (POS-tags) og *FilnavnetXYZ-281-jobby2.xml* (lemmaer) iklædes metadata og brugeren gøres opmærksom på at disse to resultater nu ligger klar.

For det videre forløb henvises til kap. 3.

## Datasikkerhed

Der sørges for at de for et værktøj registrerede oplysninger kun kan redigeres af værktøjsudbyderen selv eller af en person med administratorrolle. Sletning af et værktøj vil ikke være muligt når der allerede er blevet skabt nye resurser ved hjælp af værktøjet. Hvis man ikke ønsker at et værktøj bruges, kan det gøres inaktiv.

Når et værktøj bliver aktiveret, sker det på basis af brugerens rettigheder i forhold til inputtet. Værktøjer har ikke selv autorisation til at hente resurser fra repositoriet og arbejder derfor med kopier af resurser som er lavet på vegne af brugeren. Datasikkerheden er derfor betinget af:

1. Brugerens rettigheder,
2. Kvaliteten af datakryptering ved transport mellem clarin.dk-serveren og værktøjsserveren og vice versa og
3. Adgangskontrollen til værktøjsserveren

Ad 1) Afgøres vha. WAYF

Ad 2) Der kræves sikre SSL (HTTPS) forbindelser, så chancen for succesfuld ”eavesdropping” er lille.

Ad 3) Dette er det svageste led i kæden: en værktøjsudbyder er normalt ikke den person der holder opsyn med den server hvor værktøjet kører. Hvis serveren deles med funktioner der ligger udenfor clarin.dk, er garantien for fuld datasikkerhed ikke bedre end hvad den ansvarlige for serveren er villig til at give. Denne garanti kan være fuldstændig ikke-eksisterende, fx hvis værktøjet helt eller delvist benytter sig af en offentlig tilgængelig webservice udenfor clarin.dk’s regi, eksempelvis en af de mange WebLicht webservices.

# Værktøjsintegrationsmanual

Denne manual giver en beskrivelse af integrationsprocesser for værktøjer i clarin.dk’s infrastruktur. Se afsnit 2.5.4 i DK-CLARIN: specifikation af teknisk infrastruktur (http://intern.dkclarin.dk/files/Clarin%20kravspec%20051009.doc).

Integrationsprocessen vil næppe foregå uden manuelt arbejde både på infrastruktursiden og værktøjsudbydersiden. Noget af arbejdet vil foregå vha. udfyldning af formularer på infrastrukturens hjemmeside, men der kan være aspekter ved et nyt værktøj som infrastrukturen ikke er forberedt til. For at mindske behovet for at tilpasse infrastrukturen til de enkelte værktøjers behov stiller infrastrukturen nogle overordnede krav som værktøjet som udgangspunkt skal opfylde.

## On-line registrering af værktøjer

Både til integrerede og ikke integrerede værktøjer er der en formular på clarin.dk-serveren hvor værktøjsudbyderen kan indtaste værktøjets metadata. Formularen består af en samling felter der beskriver værktøjets placering i forhold til verden omkring, og, for integrerede værktøjers vedkommende, en eller flere samlinger af felter der bruges af Tools-modulet i infrastrukturen til at beregne hvordan en bruger kan nå sit mål med startpunkt i en eller flere givne resurser.

De nævnte formularer er placeret på denne adresse: https://infra.clarin.dk/tools/register.

## Afgrænsning

Tools-modulet i infrastrukturen kan håndtere et værktøj hvis det opfylder følgende krav:

1. Værktøjet aktiveres ved en HTTPS-request.
2. Værktøjet er ”public”: alle der er logget ind på infrastrukturen kan anvende værktøjet.
3. Outputtet fra et værktøj er i et format som tillader at outputtet deponeres i infrastrukturens repositorie som indholdet af en ny resurse. Alternativt er outputtet i et ikke deponerbart (men veldokumenteret) format. Dette udelukker derfor værktøjer der har til hensigt at ændre eller supplere en resurses metadata, fx et værktøj der normaliserer forfatternavn eller gætter det anvendte sprog i resursen.
4. Værktøjets input er en eller flere clarin.dk resursers indhold (ikke metadata). Alternativt er inputtet i et ikke deponerbart (men veldokumenteret) format. Hvis det sidste er tilfældet, kan værktøjet kun modtage input som er output fra et andet værktøj. For eksempel: værktøj A udtrækker teksten fra lydsporet i en clarin.dk videoresurse i et fladt, ikke deponerbar format, og værktøj B konverter efterfølgende A’s resultat til en tekstresurse som kan deponeres eller videre bearbejdes.
5. Hvis en for værktøjet nødvendig oplysning ikke er indeholdt i resursens indhold (fx sprog, format eller anden karakterisering af resursen), modtager værktøjet oplysningen i form af en HTTPS request-parameter. (Eller som XML-data.)
6. Alle parametre som et værktøj ønsker at modtage, skal oplyses ved registrering af værktøjet. Det gælder både input- og outputparametre.
7. I tilfælde at et værktøj arbejder med en deponerbar resurse som input, skal Tools-modulet kunne finde de ønskede inputparametre og deres aktuelle værdier i metadata tilknyttet inputresursen, hvilket betyder at inputresursens metadata skal være beskrevet i et format som understøttes i clarin.dk infrastrukturen.
8. Værktøjet henter ikke selv resurser fra repositoriet og deponerer ikke selv resurser i repositoriet.
9. Værktøjet har ikke selv direkte interaktion med brugeren. Outputtet fra værktøjet sendes til Tools-modulet i infrastrukturen. (Adressen er indeholdt i request-parametrene til værktøjet.)

## Registreringsformularen

De metadata der beskriver værktøjets placering i forhold til verden omkring er følgende:

### Generelle tools-metadata

1. *ToolID*: Værktøjets korte navn. Dette navn vil være værktøjets kaldenavn og det skal være unikt. Hvis et navn vælges som allerede er i brug, bliver man bedt om at angive et andet navn. Navnet skal overholde kravene til et xml:id. (Se http://www.w3.org/TR/xml-id/)
2. *Version*. (Kan stå tomt)
3. *Title*: Værktøjets fulde navn.
4. *Publisher*
5. *Content Provider*
6. *Creator* of the tool
7. *Rights*
8. *InfoAbout*: URL som peger på værktøjets dokumentation
9. *Description*: Kort beskrivelse af værktøjet.
10. *ServiceURL*: URL som er værktøjets entry-point (for integrerede værktøjer)
11. *External URI* (for eksterne værktøjer)
12. *ConformsTo*: anvend HTTPS POST eller GET. Med GET skal webservicen selv hente resourcen og bliver parametrene sendt som request-parametre. Med POST er der to yderligere felter:
13. *PostData*: ønskes inputtet uploaded (*on*) eller vil webservicen selv hente den.
14. *XMLparms*: ønskes parametrene tilsendt som xml data (*on*) eller som request-parametre.
15. *Inactive*: ønskes værktøjet taget ud af brug (*on*).

*ServiceURL* kan udelades hvis værktøjet ikke er integreret. Der vil være en beskyttelse mod uautoriseret brug af registreringsformularen. Det er oplagt at kun et begrænset antal brugere skal have lov til at redigere værktøjsmetadata, og at redigeringsrettigheder for et bestemt værktøj kun skal gives til værktøjets udbyder, medmindre andet er aftalt. Vi stræber efter en løsning der bruger de samme autoriseringsmekanismer som brugt i sammenhæng med dataresurser.

### Registrering af integrerede værktøjer

I den anden del af registreringsformularen, som kun skal udfyldes for integrerede værktøjer, specificerer værktøjsudbyderen de forskellige træk som kendetegner værktøjet og som gør det muligt på kontrolleret vis at fodre output fra et værktøj som input til et andet værktøj. For hvert træk skal værktøjsudbyderen specificere hvad der kræves på inputsiden og hvad der kommer ud på outputsiden, givet hvad der blev specificeret på inputsiden.

Mindst ét træk skal specificeres. Træk som ikke specificeres, betragtes som ukritiske for værktøjets funktion og defineres til at være ens på input- og outputsiden.

For tiden skønnes at følgende træk er nødvendige, og måske også tilstrækkelige:

1. Facet. Med dette træk udtrykkes hvilken side af resursen der belyses. Normalt udtrykkes brugerens formål i dette træk. Eksempler (for tekstresurser) er:
   1. Selve teksten
   2. tekstens mindste enheder (tokens)
   3. tekstens sætninger
   4. ordklassen for tekstens tokens
   5. leksikonoplsagsformen for hvert token i teksten (lemma)
   6. hyppigheden for hvert token i teksten
2. Format. Med dette træk udtrykkes hvordan resursen er udtrykt i bits og bytes. Normalt vil dette træk være fastlagt på baggrund af de formateringskrav clarin.dk stiller til data der skal gemmes i repositoriet.
3. Sprog (language). Værktøjer der arbejder med tekstresurser er normalt begrænset til at arbejde med ét sprog.

Til samme værktøj kan man definere flere træksamlinger, og det er også muligt at definere mere end en kombination af input/output-træk. Således er det muligt at definere værktøjet ABC som et værktøj der finder navne i enten bulgarsk, serbisk, eller russisk, men ikke i andre sprog. Og det er muligt at definere at samme værktøj tager flad tekst som input hvis det arbejder på bulgarsk, men udelukkende clarin.dk’s standard-format hvis den skal arbejde på serbisk eller russisk.

## Forberedelse til integration af et nyt værktøj

### Webservice

Værktøjet skal kunne aktiveres over en sikker internetforbindelse (https). Hvis værktøjet kræver input fra clarin.dk’s repositorium (fx en tekst, en annotation eller en film), vil værktøjet modtage URL’er til hvert af disse data i en GET- eller POST-request fra Tools-modulet i infrastrukturen. Hvis værktøjet ikke får inputtet tilsendt i en POST-request, skal værktøjet hente (med GET metoden) de nævnte resurser. Efter processering skal værktøjet returnere resultatet til infrastrukturen.

### Resursetyper

For så vidt værktøjet skal arbejde med resurser som administreres af clarin.dk-infrastrukturen, skal værktøjet overholde de aftalte dataformater for clarin.dk resurser.

### Parametre

Værktøjsudbyderen skal bruge formularen til at oplyse hvilke parametre værktøjet har og hvilke værdier disse parametre kan antage. Formularen giver et antal valgmuligheder for både parametre og parameterværdier. Hvis en parameter ikke kan udtrykkes i formularen, er der overordnet to muligheder for at komme videre:

1. Den ansvarlige for Tools-modulet[[1]](#footnote-1) kontaktes og problemet løses ved at udvide listen af mulige værdier for et givent træk eller ved at introducere et nyt træk, inklusiv en liste med værdier det nye træk kan antage.
2. Værktøjsudvikleren træffer et hardkodet parametervalg.

Parameternavne og –værdier skal uden problemer kunne bruges i forskellige sammenhænge, både i direkte kontakt med brugere i infrastrukturens brugergrænseflade og i tekniske sammenhænge såsom kommunikation mellem infrastrukturen og integrerede værktøjer. Tools-modulet håndterer for hvert parameternavn og for hver parameterværdi både en lang form til brug i kommunikation til brugeren og en kort form til tekniske formål. De lange former stiller ikke andre krav til syntax end at de ikke skal indeholde linjeskift og at længden skal begrænses til 30 karakterer. De korte former skal kunne bruges i både http query strenge og som elementnavne i XML. For at undgå unødige tekniske problemer, skal de korte former overholde disse krav:

1. Maksimal 6 karakterer, gerne kortere.
2. Kun ASCII-tegn (ingen danske bogstaver!).
3. Første tegn skal være et bogstav.
4. De efterfølgende tegn må være bogstaver, cifre og bindestreg (minus-tegn).
5. Formen må ikke begynde med bogstaverne (('X'|'x') ('M'|'m') ('L'|'l')).

Udtryk der overholder disse krav er gyldige som NCName og som http request-parametre.

### Parametre som request-streng

Infrastrukturen kommunikerer værktøjets argumenter til værktøjet vha. GET- eller POST-parametre. Vi har bestræbt os på at holde URL’ens længde til et minimum. Samtidig har vi forsøgt at lave URL’ens opbygning generel. Derfor kan det hænde at et værktøj modtager parametre som er overflødige. Eksempelvis vil et værktøj som er registreret som et værktøj der kan operere på dansksproget input, altid få en parameter der udtrykker at inputtet er dansksproget, selvom værktøjet ikke kan operere på andet end dansksproget input og sprogparameteren derfor indeholder overflødig information. Det er op til værktøjet at bruge de relevante parametre og at ignorere resten.

Parametrene kan deles i to grupper. Den ene gruppe indeholder udelukkende parametrene som identificerer værktøjets aktuelle opgave (job), baseadressen for værktøjets input (base) og adressen hvor resultatet skal sendes til (post2). Disse parametre skal bruges hvis værktøjet arbejder asynkront (job og post2) hhv. selv forventes at downloade inputtet fra infrastrukturen (base). Den anden gruppe er ikke en lukket klasse, men vil vokse med tiden for at følge med i det voksende antal træk, trækværdier og trækværdipræciseringer som bliver nødvendigt for at karakterisere stedse nye typer af resurser. For nu ikke at gøre det nødvendigt at tillære eksisterende værktøjer nye fortolkningsmetoder, er alle parametre opbygget således:

<PARM> = <IO> ’=’ <TRÆK>

| <IO> <TRÆK> ’=’ <TRÆKVÆRDI>

| <IO> <TRÆK> <TRÆKVÆRDI> ’=’ <SPEC>

| <IO> <TRÆK> <TRÆKVÆRDI> <SPEC> ’=’ <STRENG>

| <IO> <TRÆK> <TRÆKVÆRDI> ’F’ ’=’ <STRENG>

<IO> := ’I’ | ’O’

<TRÆK> := STRENG(et vedtaget navn for et træk) = ’facet’ | ’format’ | ’lang’

<TRÆKVÆRDI> := STRENG(et vedtaget navn for en trækværdi passende til trækket)

<SPEC> := STRENG(et vedtaget navn for en trækværdispecificering passende til trækværdien)

Hvis et parameternavn med N (N > 2) dele forekommer i parameterlisten, forekommer også parameternavnet med de N-1 første dele i listen, med det N’te led som værdi. En parameterfortolker kan derfor lave en systematisk parsning af alle parametre ved at starte med at finde parametrene med de korteste navne, I og O. Den næste generation af parameternavnkandidater opbygges og findes, og processen gentages indtil alle parametre er parset. Alternativt kan værktøjet søge efter bestemte parametre som er relevante. Bemærk at parametrenes rækkefølge ikke har nogen betydning og at den ikke er lagt fast.

Her er et tænkt eksempel på https-parametre sendt til et værktøj:

base=https://infra.clarin.dk/toolsdata/&post2=https://infra.clarin.dk/tools/upload&job=8-jobby1&O=facet&Ofacet=tok&Ofacettok=DSLb&O=format&Oformat=xm&Oformatxm=teip5wp2&O=lang&Olang=da&I=facet&Ifacet=txt&I=format&Iformat=xm&I=lang&Ilang=da& Ifacettxt=F&IfacettxtF=FileXYZ18

Lad os se på hver parameter:

base=https://infra.clarin.dk/toolsdata/

base parameteren fortæller hvor input til værktøjet skal hentes fra (https GET)

post2=https://infra.clarin.dk/tools/upload

post2 parameteren fortæller hvor output fra værktøjet skal sendes til (https POST). Denne parameter kan ignoreres hvis værktøjet returnerer resultatet med det samme

job=8-jobby1

job parameteren skal sendes uændret til adressen nævnt i post2. job parameteren gør det muligt at associere det POSTede resultat med en workflow, som Tools-modulet i infrastrukturen har ansvar for at afvikle.

O=facet

Denne parameter fortæller at outputtet specificeres med et træk og at trækket er facet. Med den viden kan fortolkeren lede efter en eller flere andre parametre som har et navn bestående af denne parameters navn (O), efterfulgt af denne parameters værdi (facet). I dette tilfælde findes en sådan parameter:

Ofacet=tok

Alle parametre der starter med O specificerer et aspekt ved værktøjets output. Med rette kan vi kalde O for et keyword som fortolkeren skal kende. Efter O´-et kommer tre bogstaver der refererer til et træk. I øjeblikket er følgende træk defineret:

facet facet

format format

lang language

Ovenstående parameter fortæller at outputfacetten er tok, hvilket er en forkortelse for *tokenizer*. Igen sættes nu navn og værdi sammen og fortolkeren leder efter parametre med navn Ofacettok. En sådan parameter findes:

Ofacettok=DSLb

I parameteren Ofacettok gentages altså førnævnte parameternavn, men med førnævnte parameter*værdi* som ekstra krølle. Parameteren udtrykker en ekstra oplysning om outputfacetten, nemlig at den er af type DSLb, hvilket kunne betyde *DSL-basis*. Næste parameter er

O=format

Bemærk at vi tidligere havde en parameter ved samme navn, men med en anden værdi. Det er tilladt, og det kan ske for alle parametre undtagen job, base, og post2-parametrene. Igen sætter vi navn og værdi sammen til et nyt parameternavn og leder efter alle forekomster:

Oformat=xm

Denne parameter siger at outputformatet er xml. Næste parameter er

Oformatxm=teip5wp2

Igen, parameternavnet Oformatxm er en sammentrækning af navnet og værdien i den forrige parameter. Parameterens værdi er en præcisering af outputformatet.

O=lang

Olang=da

Outputsproget er dansk

I=facet

Ifacet=txt

Alle parametre med navne startende med I refererer til inputtet. I er ligesom O et keyword som fortolkeren skal kende og reagere på. Denne parameter fortæller at inputtet er *tekst*.

I=format

Iformat=xm

Formatet af inputtet er XML

I=lang

Ilang=da

Sproget af inputtet er dansk.

Ifacettxt=F

Denne parameter fortæller at tekstinputtet er associeret med en fil. Vi bruger her et nyt reserveret keyword ’F’. En nærmere specificering er i den næste parameter:

IfacettxtF=FileXYZ18

Denne parameter fortæller at tekstinputtet hedder FileXYZ18. Når værktøjet skal hente filen skal base-parameteren og filnavnet sættes sammen til en URL:

https://infra.clarin.dk/toolsdata/FileXYZ18

Begrundelsen for den omstændelige made at specificere inputfilen på er at et værktøj som tager flere end én inputfil skal være i stand til at kende forskel. Hvis fx en dansk og en fransk tekst skal aligneres, kan disse inputs specificeres med

Ilangda=F

IlangdaF=FileABCpolit.xml

Ilangfr=F

IlangfrF=FilXYZmond.xml

Ofte vil der være mange måder at specificere en inputfil på. Det ene træk er ikke mere fremtrædende end det andet, og derfor kunne inputtet til vores eksempelværktøj relatere til format eller sprog, i stedet for til facet:

Iformatxm=F

IformatxmF= FileXYZ18

hhv

Ilangda=F

IlangdaF= FileXYZ18

### Aktivering af webservice med POST- i stedet for GET-request

Værktøjsudbyderen kan vælge at infrastrukturen sender et POST-request i stedet for et GET request til værktøjet. Det giver et par ekstra valgmuligheder:

* Parametrene kan pakkes ind i XML-data.
* Infrastrukturen kan sende inputtet til webservicen som file upload, i hvilket tilfælde webservicen ikke selv behøver hente resursen. Det kan betyde en større datasikkerhed, da inputtet ikke på noget tidspunkt behøver være tilgængeligt via en URL.

Parametrene fra eksemplet

base=https://infra.clarin.dk/toolsdata/&post2=https://infra.clarin.dk/tools/upload&job=8-jobby1&O=facet&Ofacet=tok&Ofacettok=DSLb&O=format&Oformat=xm&Oformatxm=teip5wp2&O=lang&Olang=da&I=facet&Ifacet=txt&I=format&Iformat=xm&I=lang&Ilang=da& Ifacettxt=F&IfacettxtF=FileXYZ18

udtrykkes som XML som følger:

<?xml version="1.0"?>

<parms>

<input>

<facet>

<txt>

<F>FileXYZ18</F>

</txt>

</facet>

<format>

<xm />

</format>

<lang>

<da />

</lang>

</input>

<output>

<facet>

<tok>

<DSLb />

</tok>

</facet>

<format>

<xm>

<teip5wp2 />

</xm>

</format>

<lang>

<da />

</lang>

</output>

<base>https://infra.clarin.dk/toolsdata/</base>

<post2>https://infra.clarin.dk/tools/upload</post2>

<job>8-jobby1</job>

</parms>

Her er et eksempel på hvordan XML-input til en søgemaskinegenerator kan se ud:

<?xml version="1.0"?>

<parms>

    <input>

        <facet>

            <boerge />

        </facet>

        <format>

            <praat>

                <F>FileMan18</F>

                <F>FileTir19</F>

                <F>FileOns26</F>

                <F>FileTor46</F>

                <F>FileFre47</F>

</praat>

        </format>

    </input>

    <output>

        <facet>

            <searchengine />

        </facet>

        <format>

            <url />

        </format>

    </output>

    <base>https://infra.clarin.dk/toolsdata/</base>

    <post2>https://infra.clarin.dk/tools/upload</post2>

    <job>8-jobby1</job>

</parms>

Forklaring:

Inputtet har ikke en p.t. veldefineret facet, men da vi gerne vil kunne omtale outputfacetten, har vi introduceret en facetværdi der kan bruges som inputfacet, boerge[[2]](#footnote-2). Formatet er defineret som Praat, og der er fem resurser med det format. Outputtet har facet searchengine, og det er formateret som url. Søgemaskinegeneratoren forventes at generere en søgeindeks til de fem Praat-filer og at returnere en URI der giver en bruger adgang til en søgemaskine som bruger den genererede søgeindeks. URI’en skal være pakket ind i en fil og returneres til infrastrukturen som resultat af søgemaskinegenereringen. Søgeindekserne selv kan fx ligge på den server som URI’en henviser til. Efter noget tid (fx en dag) bliver søgeindekser slettet, hvorefter søgemaskinen ikke længere kan søge i de fem Praat-resurser.

Parametrene i eksemplet kan alternativt sendes til søgemaskinegeneratoren som request parametre i søgemaskinegeneratorens URI.

### HTTP status codes

Hvis et værktøj returnerer resultatet som svar på GET-forespørgselen fra Tools, skal statuskoden være 200: OK. Hvis værktøjet arbejder asynkront og senere sender et resultat med en POST, skal værktøjet svare med 202: Accepted. Hvis værktøjet returnerer en anden statuskode og dermed indikerer at forespørgslen mislykkedes, skal det returnerede indhold være en forklarende tekst som er egnet til visning for brugeren.

### Afvikling

Et værktøj skal reagere på en request inden der sker en time-out.

Hvis værktøjets processering af dataene tager for lang tid til at kunne nås inden time-out, skal processeringen ske asynkront med værktøjets svar, som i så fald vil bestå af en HTTP status code 202: Accepted. Brugeren får i det tilfælde besked fra infrastrukturen at outputtet forventes på et senere tidspunkt. Ved asynkron processering sendes resultatet med POST til en adresse som er nævnt i post2 parameteren og job-parameteren returneres som query string sammen med resultatet, som om resultatet fra værktøjet stammede fra et formular med et file-upload element og et input af typen *text* med navn job og med samme værdi som job-parameteren i request strengen til værktøjet. Værktøjet får statuskode 200: OK tilbage fra infrastrukturen.

1. P.t. Bart Jongejan, bartj@hum.ku.dk [↑](#footnote-ref-1)
2. Dette er en nødløsning. Det er meningen at introducere en facet der er opkaldt efter Praats output. [↑](#footnote-ref-2)