CPREADER

Dokumentation

Version 1.0.1

MAGENTA aps

Forfatter:
Søren Kirkegård
Studerende på Professionsbacheloruddannelsen Webudvikling
KEA Medie/IT

14. januar 2014

MAGENTA aps

INDHOLDSFORTEGNELSE

1 INTRODUKTION

CPReader er en webbaseret grænseflade, som giver brugere adgang til at lave opslag på CPR-numre, samt søge på navne i CPR Brokeren. Brugergrænsefladen er baseret på Twitter Bootstrap 3, der er et såkaldt *mobile first* framework, som benytter CSS3 medie queries. Dette gør at applikationen vises fornuftigt på forskellige skærmstørrelser, men det sætter dog et krav til din installerede browser, da teknologien er relativt ny.

1.1 Understøttede browsere

- Chrome
- Firefox 3.5+
- Opera 9.5+
- Safari 3+
- Internet Explorer 9+

1.2 Brug af eksterne kilder

CPReader er lavet med henblik på at kunne fungere omkostningsfri, hvilket betyder at det er muligt at begrænse opslag i CPR Brokeren til kun at være lokale. Det er også værd at nævne at det kun er, hvis der laves et opslag på et CPR-nummer eller klikkes videre til en relation af en opslået person, at applikationen evt. vil benytte en ekstern kilde. Alle søgninger på navne foregå kun i lokalt data på din CPR Broker. Der henvises til afsnittet *Sådan benyttes accesslevel* for en grundig gennemgang af applikationens brug af CPR Broker i forhold til eksterne kilder.

1.3 JavaScript

Der benyttes JavaScript-kode på klientsiden i applikationen til at lave validering af indtastninger i søgefeltet, samt beslutte hvilken type forespørgsel, der skal laves baseret på indholdet. Dette betyder, at det er et krav for applikationen at JavaScript er slået til på din browser, da der ellers ikke kan sendes forespørgsler fra klientsiden til serversiden.

1.4 Sikkerhed

Det er anbefalet, at din CPR Broker er opsat med et SSL certifikat, da kommunikationen imellem de to applikationer så kan forgå krypteret. Det samme er gældende for den directory service[1], som du har valgt at bruge til autentificering og håndtering af tilladelser. Herudover anbefales det at kommunikationen imellem brugerne og CPReader forgår krypteret. Dette kan gøres ved at det opsættes på en webserver applikation foran med et SSL certifikat, der håndter klientside forespørgsler og videresender dem til CPReader. Der henvises til dokumentationen for din webserver applikation for information om, hvordan dette gøres rent praktisk. Du kan finde

1 directory service http://en.wikipedia.org/wiki/Directory_service

information om hvordan SSL ellers skal opsættes i CPReader under afsnittet *SSL*.

1.5 Demoversion

Der er stillet en demoversion af CPReader tilgængelig på addressen http://cpreader.magenta-aps.dk, hvor du kan logge ind med brugeren `test` og kodeordet `test`. Selvom applikationen er lavet med henblik på at fungere på open source software, så er demoen opsat på en Windows Server 2008 med en Internet Information Services til håndtering af krypteret SSL kommunikation mellem brugeren og CPReader. Dette er et bevidst valg for at fremvise at dette også er muligt. Der gøres opmærksomt på at det ikke er ægte cpr data på demoen, samt at det benyttede test data er stillet tilrådighed af CSC.

1.6 Licens

CPReader frigives under licenserne MPL 2.0[2], GPL 2.0[3] og LGPL 2.1[4], som er kompatible med licenserne for de benyttede moduler, biblioteker og frameworks, der er nærmere beskrevet på side 16. Samtidigt matcher dette CPR Brokerens nuværende licensering.

2 MPL 2.0 http://opensource.org/licenses/MPL-2.0

3 GPL 2.0 http://opensource.org/licenses/GPL-2.0

4 LGPL 2.1 http://opensource.org/licenses/LGPL-2.1

2 IMPLEMENTERING AF CPREADER

Det er to måder, hvorpå du kan implementer CPReader. Du kan benytte distributionsfilen eller kildekoden. Sidstnævnte giver lettest adgang til opsætningerne, men kræver at du selv installerer Play 2 frameworket.

2.1 Java

Begge metoder kræver at du har Java 7 JDK eller nyere installeret, samt at din path indeholder filstien til java og javac filerne. Dette kan kontrolleres ved at stå i et andet bibliotek end det, hvor java er installeret og skrive java -version og javac -version. Hvis disse to kommandoer ikke viser versionsnumrene, så har du enten ikke installeret Java eller fået det korrekt opsat i dit miljø.

2.2 Front-end webserver applikation

Det anbefales at opsætte en front-end webserver til at håndtere kommunikation mellem bruger og CPReader. Denne bør opsættes til at altid benytte SSL, så det hele forgår krypteret. Der henvises til dokumentationen for din valgte webserver applikation, samt Play! 2 dokumentationen om emnet her:

http://www.playframework.com/documentation/2.2.x/HTTPServer

Hvis din valgte webserver applikation er IIS 7+, så kan det anbefales at læse dokumentationen om URL Rewrites her:

http://www.iis.net/learn/extensions/url-rewrite-module/reverse-proxy-with-url-rewrite-v2-and-application-request-routing

2.3 Brug af kildekode

Hent kildekoden på softwarebørsen og udpak den på din server. Installer Play 2! frameworket fra http://www.playframework.com/.

Ved brug af kildekoden, så ligger konfigurationsfilerne i biblioteket `xxx/conf/`, hvor det er filen `application.conf` der skal redigeres og `xxx` er det bibliotek, som du har udpakket kildekoden til.

Når du har konfigureret applikationen til at passe i dit miljø, så startes den ved at skrive `play clean start` i applikationens rod bibliotek. Som default benytter Play! 2 port 9000, men dette kan ændres ved brug af parameteren `-Dhttp.port`.

Der kræves internet adgang ved denne løsning, da alle afhængigheder vil blive hentet, når CPReader opstartes ved denne metode.

Der henvises til Play! 2 dokumentationen for yderligere information:
http://www.playframework.com/documentation/2.2.x/Production

2.3.1 Filstruktur

```
app                      → Application sources
 └ assets                → Compiled asset sources
    └ stylesheets        → Typically LESS CSS sources
    └ javascripts        → Typically CoffeeScript sources
 └ controllers           → Application controllers
 └ models                → Application business layer
 └ views                 → Templates
build.sbt                → Application build script
conf                     → Configurations files and other non-
compiled resources (on classpath)
 └ application.conf      → Main configuration file
 └ application-logger.xml→ Main loggerconfiguration file
 └ routes                → Routes definition
public                   → Public assets
 └ stylesheets           → CSS files
 └ javascripts           → Javascript files
 └ images                → Image files
project                  → sbt configuration files
 └ build.properties      → Marker for sbt project
 └ plugins.sbt           → sbt plugins including the declaration
for Play itself
lib                      → Unmanaged libraries dependencies
logs                     → Standard logs folder
 └ application.log       → Default log file
 └ perf4j.log           → Default performance log file
target                   → Generated stuff
 └ scala-2.10.0
    └ cache
    └ classes            → Compiled class files
    └ classes_managed    → Managed class files (templates, ...)
    └ resource_managed   → Managed resources (less, ...)
    └ src_managed        → Generated sources (templates, ...)
test                     → source folder for unit or functional
tests
```

Her er et overblik af filstrukturen i kildekoden. Det understående er kopieret fra Play! 2 frameworkets dokumentation[5], som der henvises til for yderligere information.

2.4 Brug af distributionsfil

Hvis du vælger at bruge distributionsfilen, så har du ikke ikke adgang til at redigere

5 http://www.playframework.com/documentation/2.2.x/Anatomy

konfigurationsfilen: `application.conf`. Derfor skal du lave en selv og specificere at den skal benyttes. Du kan benytte eksemplet på en konfigurationsfil i fra denne dokumentation som udgangspunkt og gemme den i `/conf/` biblioteket. Bemærk at konfigurationsfilen ikke behøver at hedde `application.conf`, men at den skal være i UFT-8 formatet.

Når du har lavet en gyldig konfigurationsfil og gem den, så kan du starte CPReader applikationen ved at skrive:

```
play-cpreader -Dconfig.file=/path/to/conf/application.conf
```

Dette vil starte CPReader på port 9000. Hvis du ønsker at benytte en anden port kan du gøre det ved tilføje parameteren `-Dhttp.port=portnummer` til overstående. For yderligere information henvises til Play! 2 dokumentationen:

http://www.playframework.com/documentation/2.2.x/ProductionConfiguration

Du har ikke mulighed for at ændre `messages`-filen ved brug af distributionsfilen.

2.4.1 Filstruktur

```
bin                     → Application sources
 └ play-cpreader        → Executeable application file
 └ play-cpreader.bat    → Batch file for Windows
 └ logs                 → Compiled asset sources
    └ application.log   → Default log file
    └ perf4j.log        → Default performance log file
conf                    → Configurations files and other non-
compiled
lib                     → Unmanaged libraries dependencies
```

2.4.2 Windows

Du skal følge overstående vejledning, men hvis du bruger Windows, så skal du bruge `play-cpreader.bat` filen til at starte applikationen. For at kunne tilføje parameterne, så skal du oprette dem som en system variable med navnet `$PLAY_CPREADER_OPT`. Tilføj som minimum konfigurations parameteren til konfigurationsfilen i variables værdi.

3 OPSÆTNING

3.1 SSL

Alt kommunikation kan forgå over SSL. I det tilfælde er det nødvendigt at opsætte CPReader til at stole på de opsatte certifikater for de eksterne services, som benyttes: CPR Broker og din directory service. Det er mulig at lave en opsætning, som benytter forbindelser, der ikke er krypterede med SSL.

3.1.1 Keystore

Ændre værdien af `keystorefile` i opsætningen til at indeholde den absolutte filsti inklusiv filnavnet på keystore-filen i anførelsestegn.

Hvis lokalisationen af ens keystore-fil er : `conf/jssecacerts`, så skal der stå:

```
keystorefile = ”conf/jssecacerts”
```

Ændre værdien af `keystorepassword` i opsætningerne til at indeholde kodeordet til din Keystore i anførelsestegn.

Hvis kodeordet til din keystore-fil er `changeit`, så skal der stå:

```
keystorepassword = ”changeit”
```

3.1.2 Certifikat i Keystore

Værktøjet `keytool` benyttes til at importere certifikater ind i Java's keystore. Det er nødvendigt at du kender filstien til din keystore og have en fil med certifikatet, som du vil importere. Skulle du ikke have dette tilgængeligt, så kan det hentes manuelt. Se punktet Download af certifikat.

Hvis lokalisationen af din keystore-fil er: `/cpreader/conf/jssecacerts`, og din certifikat-fil hedder `certifikat.pem`, så vil du kunne bruge følgende kommando for at importere dit certifikat:

```
keytool -import -alias somealias -file certificate.pem
-keystore /cpreader/conf/jssecacerts
```

Denne kommando vil gemme certifikatet under alias `somealias` i keystore-filen. I dette eksempel så udføres kommandoen fra samme bibliotek, som din certifikat-fil er.

3.1.3 Domænenavn for lokal IP

SSL certifikater fungere på domænenavnet, så det er nødvendigt at referere til domænenavnet på den givne service og ikke IP'en.

Hvis du har en directory service på IP'en 192.168.0.2, der er sat op med et certifikat på ds.example.com, så kan følgende linje indsættes i `/etc/hosts` på ens CPReader server:

```
192.168.0.2 ds.example.com
```

På Windows er filstien til denne typisk `C:\windows\system32\drivers\etc\hosts`

3.1.4 Download af certifikat

Det er muligt med værktøjerne openssl og sed at hente et certifikat fra en beskyttet server. Følgende kommando benyttes, hvor `[IP]` og `[PORT]` erstattes med værdierne, der er gældende for din server.

```
Openssl s_client -connect [IP]:[PORT] 2>&1| sed -ne '/B- BEGIN
CERTIFICATE -/,/-END CERTIFICATE-/p'
```

Dit certifikat vil nu blive vist, så du har muligheden for at at markere og kopier det ned i en fil f.eks. `certifikat.pem`.

3.2 CPR Broker

Dette forudsætter at du har opsat CPReader applikationen i CPR Broker og at denne er opsat med SSL, da det er et krav at kommunikationen er krypteret. De fire værdier, der skal opsættes din konfigurationsfil er:

- cprbroker.ssl – Dette er en boolean værdi (true/false), der afgører hvorvidt der skal benyttes SSL til at forbindelser mellem CPReader og CPR Broker.
    - Eks. cprbroker.ssl = true
- cprbroker.endpoint – Dette er en URL til din CPR Broker instans i anførelsestegn. Bemærk at ”http://” eller ”https://” vil automatisk blive sat foran adresssen udfra brugen af SSL eller ej.
    - Eks. cprbroker.endpoint = ”cprbroker.example.com/Services/Part.asmx”
- cprbroker.applicationtoken – Dette er et hexidecimal noteret GUID i anførelsestegn, der identificere din CPReader i din CPR Broker. Denne værdi fås fra din CPR Broker opsætning.
    - Eks. cprbroker.applicationtoken = ”`3ef0a846-4637-4079-bdf4-b3fa99d7dcc4`”
- cprbroker.usertoken – Dette er en tekst i anførelsestegn, der identificere din CPReader i din CPR Broker. Denne værdi fås fra din CPR Broker opsætning.
    - Eks. cprbroker.usertoken = ”CPReader”
- cprbroker.accesslevel – Dette er en talværdi, der angiver det niveau, der maksimalt er tilladt for din CPReader, at lave forespørgsler med. Det er vigtigt at skelne imellem denne angivelse og det brugte niveau. Se nedenstående afsnit for yderligere forklaring.
    - Eks. cprbroker.accesslevel = 0

3.2.1 Sådan benyttes accesslevel

De mulige værdier for accesslevel er 0,1 og 2, der betyder henholdsvis LocalOnly, LocalThenExternal og ExternalOnly. Dette er forskellige muligheder, som man kan lave forespørgsel til CPR Brokeren, der diktere hvordan din CPR Broker skal behandle forespørgselen i forhold til eksterne kilder. Det er dog vigtigt at notere sig at denne opsætning i din CPReader ikke angiver, hvordan alle forespørgsler vil foregå, men derimod det maksimalte niveau, som en forespørgsel kan foregå med. Hvis man f.eks. angiver 2, som værdien, så vil alle forespørgsler ikke være ExternalOnly, men blot tillade at CPReader benytter denne værdi. Nuværende version af CPReader benytter følgende funtionalitet i CPR Brokeren: Read, List, Search og GetUuid.

3.2.1.1 Read

Denne funktion kaldes når, der skal vises information om en given person. Hvis opsætningen tillader der, så vil denne funktion benytte LocalThenExternal. Der er tre forskellige måde, hvor dette sker.

1. Du har søgt på et navn og klikket på et af resultaterne fra listen. I denne situation vil CPR Brokeren altid have et lokalt resultat til forspørgelsen, da resultatlisten kun indeholder personer, som din CPR Broker har lokalt.

2. Du har søgt på et CPR nummer. Her kan der ske et automatisk opslag i en ekstern kilde, hvis din CPR Broker ikke har nogen information om personen, såfremt at din opsætning tillader det.

3. Du har klikket på en relation under en person. Dette vil vise informationerne om den pågældende relation, som ved et opslag på dennes CPR nummer, såfremt at din opsætning tillader det.

3.2.1.2 GetUuid

Denne funktion kan ignorer din opsætning, da den aldrig vil lave kald til eksterne kilder. Den benytter din PersonMaster til at oprette et nyt Uuid, hvis det ikke findes lokalt i din CPR Broker. Enhver søgning på et CPR nummer vil kalde denne funktion. Resultatet fra dette kald bliver brugt til at kalde Read, så der kan vises informationer om personen.

3.2.1.3 Search

Denne funktion benyttes til at lave søgninger på navn og vil altid kun søge i resultater, der findes lokalt i din CPR Broker.

3.2.1.4 List

Denne funktion er en Read for flere Uuid. Dette benyttes i applikationen til at hente de tilgængelige informationer om en opslået persons relationer og til at hente resultaterne fra en navnesøgning. I din CPReader vil der kun laves kald til denne funktionalitet, der returnere resultater, som allerede findes lokalt i din CPR Broker.

3.3 LDAP

CPReader antager at din directory service er opsat, som en variant af Illustration 1: Directory Service konfiguration, derfor tager følgende vejledning udgangspunkt i denne.

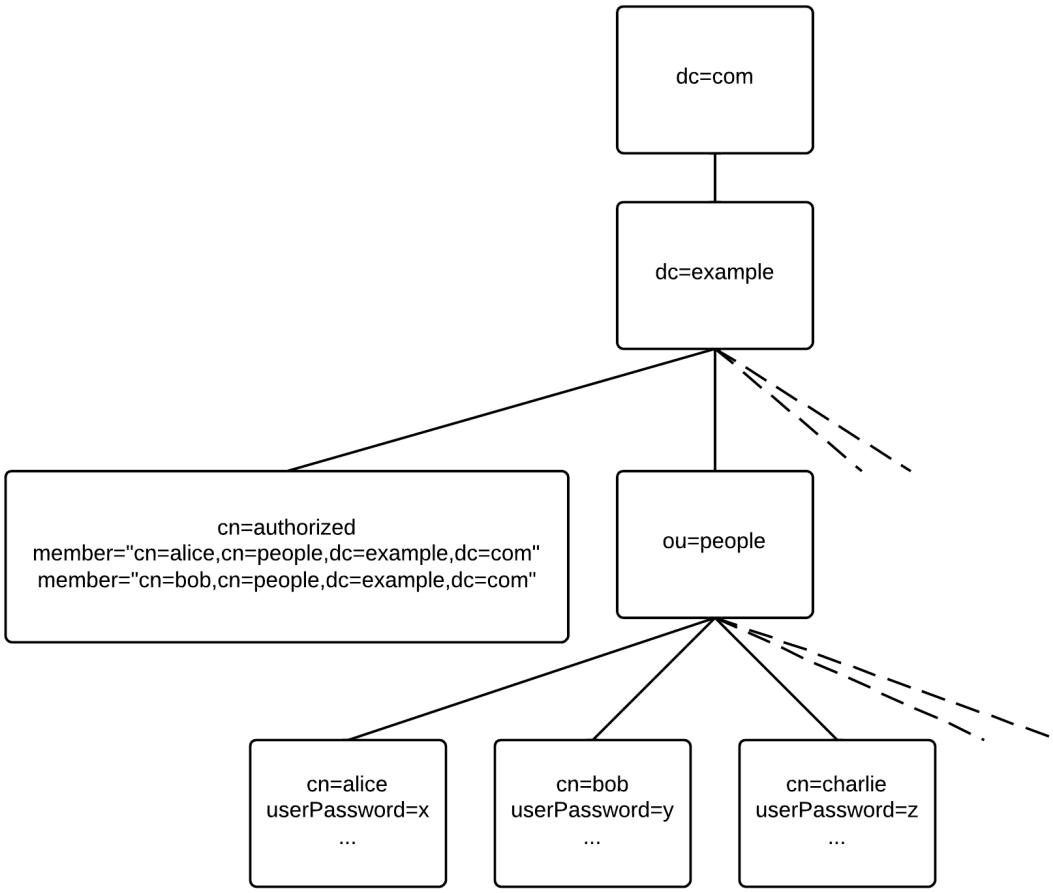

Illustration 1: Directory Service konfiguration

Autentificering forgår ved et LDAP bind kald med eller uden SSL fra CPReader til din directory service. I denne forbindelse vil bind DN være sammensat på følgende måde:

```
[userattribute]=[username],[usergroupdn],[basedn]
```

`[username]` vil her være brugerens input ved log ind. Resten af felterne er defineret i konfigurationsfilen.

Tilladelse til at benytte CPReader bliver afgjort ved at der efterfulgt en succesfuld autentificering laves en søgning på om der findes følgende attribut under RDN'et `[authorizedgroupdn]`:

```
[authorizedattribute]=[binddn]
```

Når der tages udgangspunkt i Illustration 1: Directory Service konfiguration, hvor directory servicen bliver hostet på ds.example.com port 636, så er værdierne i konfigurationsfilen følgende:

```
ldap.hostname = ”ds.example.com”
ldap.ssl = true
#ldap.port = 636
ldap.basedn = ”dc=example,dc=com”
ldap.usergroupdn = ”ou=people”
ldap.userattribute = ”cn”
ldap.authorizedgroupdn = ”cn=authorized”
ldap.autherizedattribute = ”member”
```

Bemærk at her er ldap.port kommenteret ud. Hvis dette er tilfældet, så vil CPReader vælge mellem port 636/389 alt efter om ldap.ssl er true eller false. Det er muligt at ændre standard porten ved at benytte ldap.port.

For at gøre eksemplet komplet er der her en gennemgang af forløbet, hvor brugeren alice forsøger at logge ind. `[binddn]` vil hermed blive:

```
cn=alice,ou=people,dc=example,dc=com
```

Og ved et succesfuldt bind, så vil søgningen i RDN `”cn=authorized”` være efter attributten:

```
member=”cn=alice,ou=people,dc=example,dc=com”
```

Brugeren charlie vil i dette eksempel, derfor kunne autentificere, men har ikke tilladelse til at benytte CPReader, hans DN ikke er at finde som en member-attribut i RDN `”cn=authorized”`.

3.4 Logning

CPReader benytter logback[6] til logging. Dette framework kan blive opsat med en xml-notifikation. Der bliver lavet log på to niveauer: Bruger og ydelse. Standard implantationen skriver loggen til tekst filer i /logs biblioteket. Dette kan ændres i `application-logger.xml`, der er placeret i `/conf` biblioteket. Der henvises til dokumentationen[7] af logback for information om, hvordan du laver opsætningen af denne fil.

3.4.1 Brugerniveau

Denne log er navngivet `application.log` for den logning, og indeholder information for det nuværende døgn. Herefter bliver den omdøbt til `application.[åååå-mm-dd].log`, hvor åååå, mm, dd er henholdsvis år, måned og dag for den dato, som logen omhandler.

6 Logback http://logback.qos.ch/

7 Logback dokumentation http://logback.qos.ch/documentation.html

3.4.2 Ydelsesniveau

Det samme er gældende for loggen for ydelse, som det gælder for brugerniveauet. Der er dog følgende forskelle:

- Filen hedder `perf4j.log` og `perf4j.[åååå-mm-dd].log` for de historiske data
- Det en rullende log, der maksimalt gemmer 30 dages histotik.

Det er Perf4j[8], der benyttes som et lag overpå logback. For information om hvordan, hvordan man kan benytte disse log filer til statistik henvises til dokumentation for Perf4j.

3.5 Applikationshemmelighed

Det er vigtig at du laver en ny applikationshemmelighed i din konfigurationsfil, da denne blandt andet benyttes til at sikre applikationens cookies. Hvis denne kendes kan det kompromiser sikkerheden, fordi en ondsindet bruger vil kunne ændre værdier de cookies, som applikationen benytter. Det kan opsættes i feltet `application.secret` i konfigurationsfilen. Nøglen skal være en tilfældig streng af 64 tegn eksklusiv tegnene ”, `[`, `]` og `$` og omrammet af anførelsestegn.

3.6 Tekst i applikationen

CPReader er lavet med muligheden for at ændre tekst i applikationen uden at skulle ændre i selve koden. Der benyttes en fil ved navn `messages` til at konfigurere, hvad der vises i applikationen. Hvis du har brug for det, kan dette system understøtte flere sprog.

3.6.1 Understøttede sprog

CPReader er lavet til at være en dansk sproget applikation, men dette kan let udvides. I konfigurationsfilen angives, hvilke sprog der understøttes. Dette skal angives i feltet `application.langs` med kommasepareret ISO landekoder indkapslet af anførelsestegn.

```
application.langs="da"
```

Hvis du vil understøtte flere sprog, så skal der laves en `messages.xxx` fil for hvert sprog, som der skal understøttes, hvor `xxx` er den ISO landekode, som du understøtter og har angivet i `application.langs`. Filen `messages` er en default, der benyttes til alle sprog.'

3.6.2 Relationer imellem konfiguration og grænseflade

For at give et overblik over mulighederne i `messages`-filen, så vises herunder udsnit af forskellige skærm billeder, hvor det er markeret med rødt, hvad relationerne er til konfiguration.

8 Perf4j http://perf4j.codehaus.org/

Illustration 2: Log ind skærmbillede

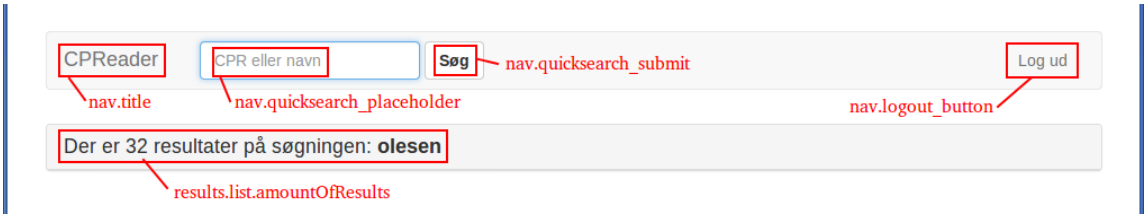

Illustration 3: Navigation og resultat skærmbillede

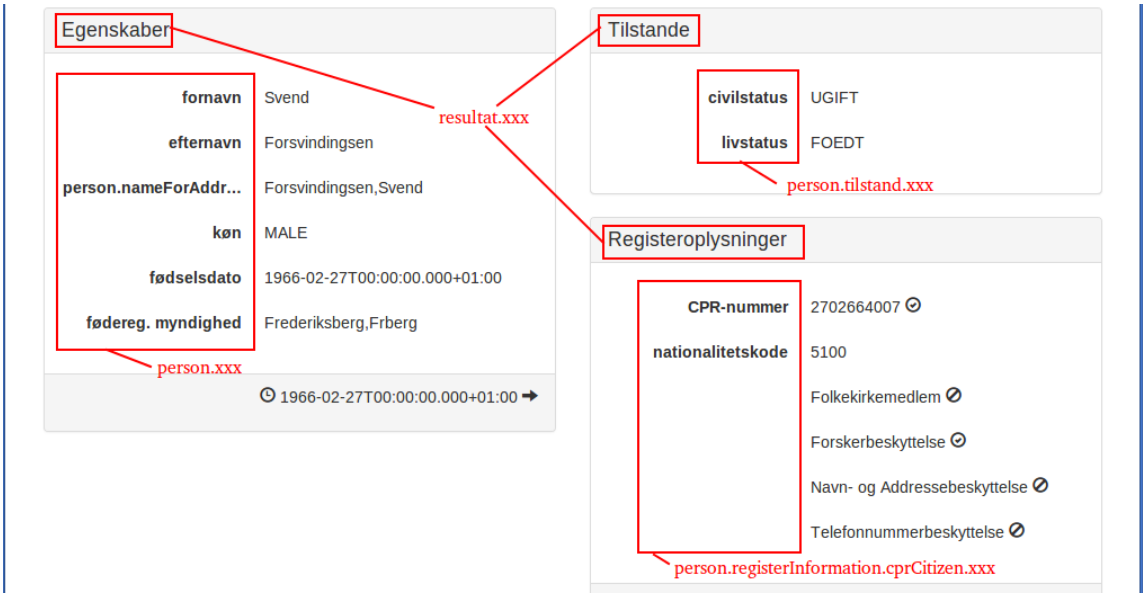

Illustration 4: Generelt information skærmbillede

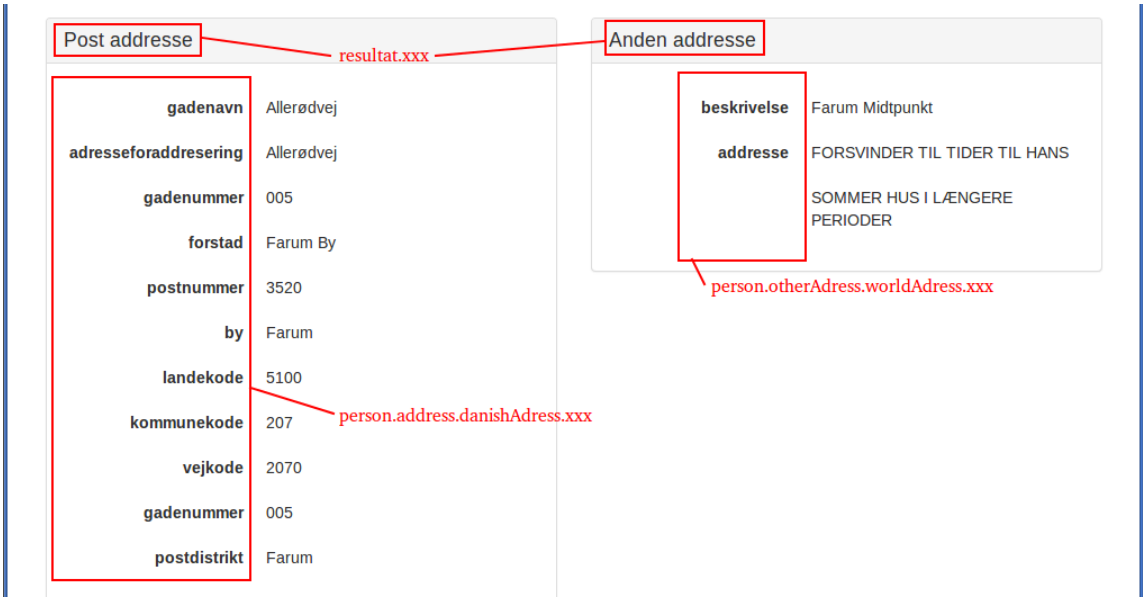

Illustration 5: Addresse skærmbillede

Illustration 6: Relationer skærmbillede

4 EKSEMPEL PÅ KONFIGURATIONSFIL

Nedenstående er et eksempel på indholdet af en konfigurationsfil:

```
# This is the main configuration file for the application.

# Secret key
# The secret key is used to secure cryptographics functions.
# If you deploy your application to several instances be sure to
use the same key!
application.secret="1=7L6&QyyXSbHk8l8-!
>z:qH{[l~{37ruhi{+yhmI3lRDd^eDgsivUzHX(.N,gZ1"

# The application languages
application.langs="da"

# Logger
# Logging is setup in the applictaion-logger.xml
# SSL settings
keystorefile = "conf/jssecacerts"
keystorepassword = "changeit"

# LDAP configuration
ldap.hostname = "ad.example.com"
ldap.ssl = true
# Only use ldap.port only to set a custom port.
# Default is 389/636 for ldap.ssl false/true
#ldap.port = 636
ldap.basedn = "dc=example,dc=com"
ldap.usergrouprdn = "ou=people"
ldap.userattribute = "cn"
ldap.authorizedgrouprdn = "cn=authorized"
ldap.authorizedattribute = "member"

# CPR Broker configuration
# accesslevel can be 0/1/2 which represents:
# localOnly/localThenExternal/externalOnly
cprbroker.endpoint = "cprbroker.example.com/Services/Part.asmx"
cprbroker.ssl = true
cprbroker.applicationtoken = "3ef0a846-4637-4079-bdf4-b3fa99d7dcc4"
cprbroker.usertoken = "CPReader"
cprbroker.accesslevel = 0
```

5 ARKITEKTUR

CPReader er baseret på Play! 2 frameworket[9], der er lavet i programmeringssproget Scala[10]. Da dette sprog er lavet til at kunne blive afviklet på JVM[11], så tillader det brug af Java[12], hvilket er sproget, der er benyttet til konstruktion af selve web applikationen. Herudover giver det mulighed for at benytte biblioteker, der er lavet til Java. Du kan få et overblik over den brugte teknologi-stak på Illustration 7: CPReader arkitektur.

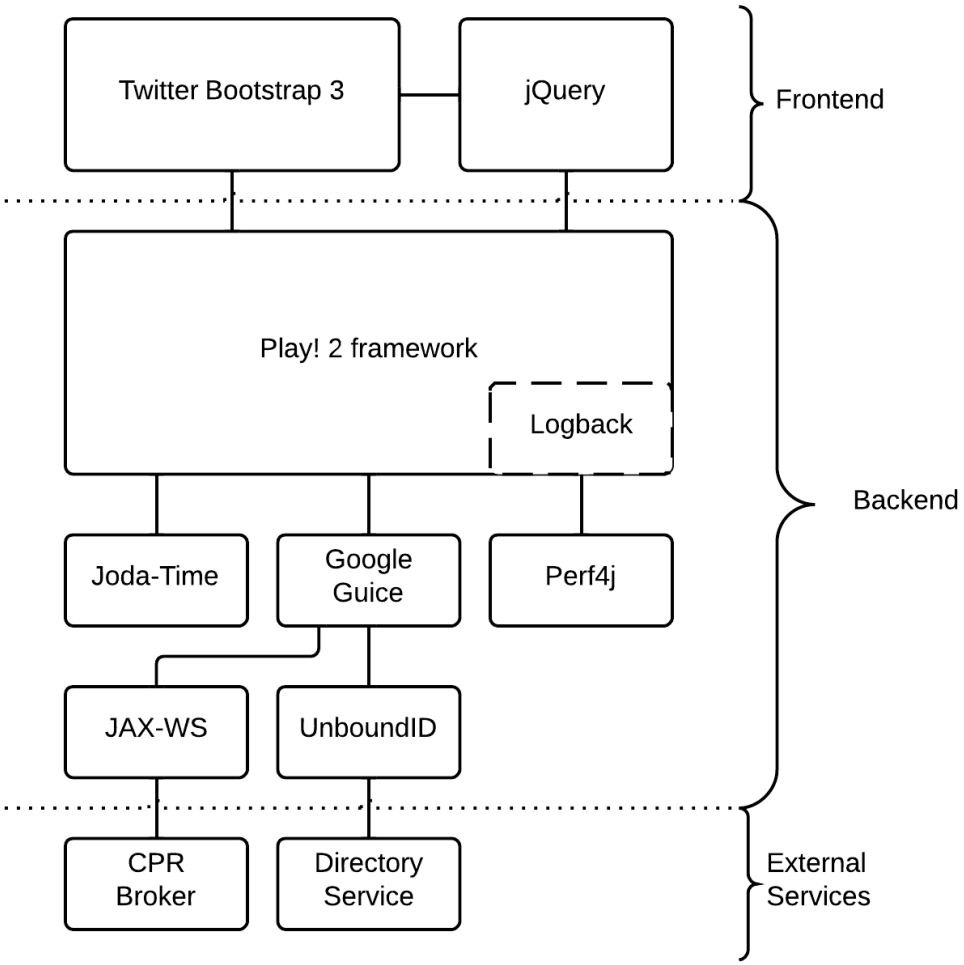

Illustration 7: CPReader arkitektur

9 Play! 2 framework http://www.playframework.com/

10 Scala http://www.scala-lang.org/

11 JVM http://en.wikipedia.org/wiki/Java_virtual_machine

12 Java http://www.oracle.com/us/technologies/java/overview/index.html

I front-end er Twitter Bootstrap 3[13] (TB3), som har en afhængihed af jQuery[14]. TB3 er udviklet til at være et mobile first[15] framework, hvilket gør at CPReader umiddelbart bør virke fornuftigt uden yderligere tilpasning på mobile enheder. Der bruges JavaScript[16] og jQuery til validering af søgning og til at beslutte hvilken URI, der skal laves en forespørgsel til. Derfor er det et krav for at CPReader skal være funktionel, at den benyttede browser kan afvikle JavaScript og at det er slået til.

Logning på brugerniveau sker ved hjælp af Logback[17], hvilket er en implementeret del af selve Play! 2 frameworket. Ydelsesniveau log laves med Perf4J[18], der er kompatibelt med Logback, hvilket gør at det kan opsættes med samme konfigurationsfil. Herudover giver Perf4J en del muligheder for at parse log-filer til en sammenfatning af den målte ydelse. Dette kan bruges direkte, som data til en graf eller blot til at give et overblik over , hvordan applikationen yder i de forskellige led.

Datoer i Java har et ikke ubegrundet rygte for at have forskellige problemer, hvilket har udledt i JSR-310[19], men da dette forslag ikke er implementeret i Java 7, så benyttes Joda-Time i stedet. Denne beslutning kan tages op til reevaluering, når Java 8 udkommer, da JSR-310 officielt er en del af funktionalitetslisten.

Google Guice[20] benyttes til dependency injection[21] af JAX-WS[22] og UnboundID[23] implementeringerne. Dette gøres for at det er lettere at teste, samt udskifte med en anden løsning, hvis det skulle blive en nødvendigt på et senere tidpunkt.

Til SOAP kommunikation med CPR Brokeren bruges JAX-WS, der siden Java 6 har været en del af Java SE. UnboundID er valgt til LDAP kommunikation, da det har et simplificeret API, i forhold til JNDI[24].

5.1 Licenser

Dette er en sammenfatning over licenserne af de brugte enheder.

13 Twitter Boostrap 3 http://getbootstrap.com/

14 jQuery http://jquery.com/

15 Mobile first http://en.wikipedia.org/wiki/Responsive_web_design#Mobile_first.2C_unobtrusive_JavaScript.2C_and_progressive_enhancement

16 JavaScript http://en.wikipedia.org/wiki/JavaScript

17 Logback http://logback.qos.ch/

18 Perf4J http://perf4j.codehaus.org/

19 JSR-310 http://jcp.org/en/jsr/detail?id=310

20 Google Guice https://code.google.com/p/google-guice/

21 Dependency Injection http://en.wikipedia.org/wiki/Dependency_injection

22 JAX-WS http://en.wikipedia.org/wiki/Java_API_for_XML_Web_Services

23 Unboundid https://www.unboundid.com/products/ldap-sdk/

24 JNDI http://en.wikipedia.org/wiki/Java_Naming_and_Directory_Interface

- Twitter Bootstrap 3 – Apache License, Version 2.0[25]
- jQuery 1.9 – MIT License[26]
- Play! 2 – Apache License, Version 2.0
- Joda-Time – Apache License, Version 2.0
- Perf4J – Apache License, Version 2.0
- JAX-WS (Java 7 SE[27]) - Er ikke distribueret med CPReader, men benyttes. GPLv2
- UnboundID – GPLv2[28], LGPLv2.1[29] og UnboundID Free Use License[30]

- Google Guice 3 – Apache License, Version 2.0

25 Apache License, Version 2.0 http://www.apache.org/licenses/LICENSE-2.0.html

26 MIT License http://opensource.org/licenses/MIT

27 Java Platform, Standard Edition 7, Reference Implementations https://jdk7.java.net/java-se-7-ri/

28 GPLv2 http://www.gnu.org/licenses/gpl-2.0.html

29 LGPLv2.1 http://www.gnu.org/licenses/lgpl-2.1.html

30 UnboundID Free Use License https://www.unboundid.com/products/ldapsdk/docs/LICENSE-UnboundID-LDAPSDK.txt

MAGENTA aps

adresse
Studiestræde 14, 1.
1455 København K
email
info@magenta-aps.dk
telefon
(+45) 33 36 96 96