

Designdokument

eValg

I 2006 (etter sterkt påtrykk fra studentene) besluttet Universitetsstyret at man skulle implementere elektronisk valg ved UiO. Dette ble gjort ved en utviklingsoppgave, og Rune Frøysa ble gitt oppgaven å skrive dette.

Ettersom Rune Frøysa i skrivende stund ikke lenger er tilsatt ved UiO, finnes ingen begrunnelser for de designvalg som ble tatt.

Det er flere formål med eValg. Tidligere var valgene arrangert (for studenter) som urnevalg eller (for ansatte) som postvalg. For studentene var avkryssing i manntallet ekvivalent med å få et kryss på semesterkortet, noe som ikke akkurat sikrer at ingen får stemt flere ganger. Ansatte ville putte stemmeseddel i en konvolutt og sende den i internposten.

Løsningen med elektronisk valg øker dermed sikkerheten i valget (gitt at velgerne ikke påvirkes av frykt for elektroniske løsninger). Anonymt valg fungerte for studentene på samme måte som stortings-, fylkestings- og kommunevalg, men for de ansatte var det mulige forbedringer på dette området.

Opptelling av papirstemmer er noe som tar tid – spesielt når man har et komplisert opptellingsreglement, som preferansevalget vårt (single transferable vote/STV) er. Adskillige opptellingstimer er spart, og datamaskiner regner ikke feil (gitt at program og maskinvare er korrekt nok).

Kontaktpunkter

Drift av elektroniske valg:	evalg-drift@usit.uio.no
Drift av tjenerløsninger:	www-drift@usit.uio.no
Utvikling av systemet:	UAIT
Systemeier:	OPA?
Valgreglene:	valgsekreteriatet@admin.uio.no . https://www.uio.no/om/regelverk/orgadm/valgreglement.html

Maskinressurser

gt-mgmt1.uio.no:	Brukes til import av persondata og andre jobber mot basen
w3prod-ws01:	Prod-miljø i dag: valg.uio.no og valg.usit.no
w3test01:	Staging-miljøer.
w3utv-ws02:	Ny sandkasse med resin 4
w3utv-ws01:	Avskrudd sandkasse.

Introduksjon

eValg er et system for det meste basert på web: Både bruker- og administrasjonsgrensesnitt er webbasert. Det er også diverse andre jobber som går i bakgrunnen. Systemet understøtter disse ulike funksjonaliteter:

- Velgere kan stemme

- Tilganger basert på ulike roller: - superadmin - valgadministrator (på enkeltvalgbasis) - rettighet til å opprette nye valg
- Ulike valgtyper
- Webgrensesnitt for bruker og administratorer
- SAML-login
- Program for opptelling av stemmer

Stemmegivning

Stemmegivningen i eValg baseres på web-løsning. Utformingen av denne har en del føringer: krav til stemmesedler i valgreglementer, samt web accessibility guidelines (<http://www.w3.org/WAI/>). Det bør også kunne håndteres av alle mulige enheter og brukerinnstillinger, derfor er det f.eks. ikke brukt JavaScript i opptellingsmodulen.

En ny ting som er kommet, er krav til opplysning om bruk av cookies, som er nødvendig for sesjonsdata. Et alternativ kunne vært query-parameter, men dette medfører en betydelig større risiko for sesjonskapring.

Redegjørelse

Datamodellen ble modellert etter UiOs reglement i 2006. Når andre senere skulle bruke systemet, ble det gjort avsjekk mot dette, og kunden kunne i de fleste tilfeller akseptere føringer basert på dette.

Avhengigheter og føringer

Prosjektet avhenger av en del eksterne biblioteker, som listet opp i byggescriptet, build.gradle. Det kjøres som en Java servlet, og bruker Resin som servlet container (men dette bør kunne byttes ut).

Databasen er PostgreSQL, men det er lite som avhenger spesifikt av Postgres.

Til autentisering brukes SAML 2. Systemet var i utgangspunktet skrevet med antagelsen at fødselsnummer vil være bijektiv for personer. Som backup bør man kunne bruke feide-id, men denne er ikke injektiv.

Som føringer brukes ulike valgreglementer, samt at systemet er sikkert.

Mangler

Datamodellen har flere tabeller for kandidater, skjønt listevalg+ kan være fleksibelt nok til alle.

Alle kandidater ligger også i Person_info. Men det forekommer ofte nok at en kandidats navn i FS e.l. ikke er det han/hun bruker og ønsker å stille med. Ved enkelte institusjoner er dette løst ved å sette inn egne personobjekter (pid like 'candidate-%', f.eks.) for kandidaten.

Mangler funksjonalitet i brukergrensesnittet for:

- Opplasting av persondata
- Legge inn personer («personer») manuelt
- Håndtere tilganger
- I tillegg til at admingrensesnittet er dårlig designet.

Deler av brukergrensesnittet (f.eks. stemmegivning) bærer preg av å være designet av komité; valgreglementer stiller krav som om det var papirvalg, og enkelte med ansvar synes wall of texts er god idé.

Kjønnskvolteringsstabellene er tilpasset UiOs preferansevalgreglement, og det har klare mangler i å spesifisere regler for varaer.

Vektete stemmer er i datamodellen angitt med et tall. Det finnes andre mer kompliserte vektingsregler som ikke er mulig:

- Institusjon X er spredd over flere steder (fusjon av to høyskoler, f.eks.)
- De ønsker vekting ikke bare basert på kategori, men også en overordnet vekting av stedene, slik at lillebror ikke blir overkjørt.

Det har forekommet med flikking at noen har klart å slette kandidater, og opprettet på nytt med annen kandidatid etter at noen har stemt på kandidaten. I så måte burde det ikke slettes rader fra kandidattabellene.

Versjonshistorikk

Versjon 1 ble laget i 2006 av Rune Frøysa. Robert Bauck Hamar tok over stort sett all utvikling av prosjektet, og gikk mot versjon 2, som hadde mer modularisert kode og en litt endret datamodell. (Versjon 1 støttet kun to valgtyper, implementert ved «if type = L then *listevalg* else *preferansevalg*.»

Bruk av systemet

Oppsett og installasjon

Systemets hovedfunksjon er en Java servlet som kommuniserer med en Postgresdatabase. GnuPG må være installert på maskinen som kjører valg.

Det kommer også med et Java-program (jar-fil), som brukes både til opptelling og til administrative oppgaver (disse bør imidlertid skilles ved anledning).

En fullstendig redegjørelse har vi ikke, men:

- master har en build.gradle-fil, og kommandoen `./gradlew build` skal i utgangspunktet bygge alt. Ev. `./gradlew war` som hopper over automatiske tester.
- Tidligere versjoner har en build.xml, hvor ant skal kunne bygge hele programmet. Dette forutsatte at katalogen `thirdparty` hadde nødvendige biblioteker, og at classpath inneholdt både disse og diverse J2EE-api-biblioteker.

Databasen initialiseres med skjema fra `src/sql/createdb.sql`

Resulterende war-ball prodsettes i ønsket miljø (testet med Resin og Tomcat), som er konfigurert slik at

- Ressurs ved navn «jdbc/postgres-valg» med type «javax.sql.DataSource» er konfigurert opp.
- valg-config.properties finnes et sted, og kan finnes av programmet som systemproperty `no.uio.valg.configFile` (se forøvrig `valg-config.properties.example` i reporoten)

Konfigurasjonsvariable

Alle disse er listet opp i `valg-config.properties.example`, og leses inn av klassen `no.uio.valg.ValgConfig`, hvis metode `init()` kalles av Struts under oppstart. De mest åpenbare er fjernet.

vote_dir

Krypterte (ev. ukrypterte) stemmer skrives til disk. Hit. Servleten trenger alle tilganger til mappen, men andre bør ikke ha det.

feide_username og feide_password

Ikke i bruk. (Ble brukt av den gamle FeideLogin-klassen som snakket SOAP med gamle feide.

receipt_from

Avsenderadresse på kvitteringer til stemmegiver.

receipt_body

Body på kvitteringer. {0} byttes ut med bruker, {1} byttes ut med valgnavn og {2} inneholder receipt_by_admin hvis stemmen blir avgitt av valgadmin.

maildomain

Versjon 1 hadde ikke mailadresse i person_info.

gpg_*

Hvordan kalle gpg for å kryptere stemmer og for å verifisere nøkkel.

super_admins

Versjon 1 hadde systemadministratorer her. Versjon 2 legger dette i basen. Men den fungerer fremdeles som en bootstrapping av dette.

login_class

Settes til klassen som håndterer autentisering. Se i pakken no.uio.valg.login: - DebugLogin: Passordet er «\$secret». Bør ikke brukes. - Md5Login: Passordet er linux-md5-cryptet i person_info-tabellen (password). - SAML2Login: Bruker SAML 2.

only_mail_self

Hvis «true», sendes alle mail til receipt_from-adressen

logout_url

Brukeren redirectes hit etter utlogging

saml2login.update_pi_from_login

Hvis true, oppdateres data fra SAML-innloggingen til databasen ved innlogging.

Innlegging av personer

Det er grunnleggende sett flere måter å legge personer inn i systemet. For UiOs del, eksporteres personer fra Cerebrum. Persondata eksporteres på ett av to formater: XML eller CSV.

XML

Filformatet er ganske enkelt:

```
<?xml version="1.0"?>
<data>
  <person first_name="Fornavn" last_name="Efternavn"
          fnr="pid"
          uname="brukernavn" email="email"/>
  <person ...
</data>
```

CSV

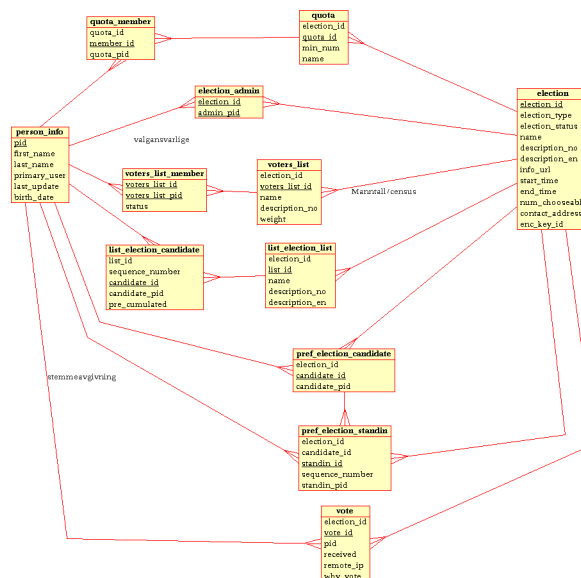
Formatet for CSV er:

```
pid;Fornavn;Efternavn;brukernavn;email
```

Tegnsettet brukt er UTF-8

Feltene

To felter er ikke helt selvforklarende: pid og brukernavn. Vi bruker fødselsnummer som personidentifikator. Dette er i essensen den eneste unike identifikatoren. For personer som ikke har dette, brukes primærbruker.



Datamodellene finnes i mappen `src/sql`. En del av tabellene er kodetabeller, og har gjerne feltene `code` (primærnøkkel), `code` (sekundærnøkkel) og `description` (beskrivelse). Disse brukes til å regulere hvilke muligheter som er aktivert.

Version

Holder versjonsdata. Tabellen støtter moduler, slik at man kan sette inn egne tabeller osv. En sjekk blir gjort ved oppstart på om tabellen inneholder korrekte verdier.

Language

Sprogkoder. Styrer sprogene sidene vises på. Brukeren kan legge inn beskrivelser på disse sprog. Dessverre er kodeverket til Javas lokaler dårlige når det gjelder norske målformer (må ha no-NO-B og no-NO-NY, og ikke nb og nn).

Domain

For hotelløsningen holder denne tabellen på institusjonsspesifikke ting. Ellers er tabellen strengt tatt redundant, men vil da holde én rad.

- name:** Navn på institusjon
- key:** matches mot brukerens feide-id ved innlogging
- dn:** Kan også brukes ved SAML2-autentisering, men ikke i bruk.
- logo_type:** MIME-type for logo-data
- logo_data:** Binær blokk med institusjonslogo
- stylesheet:** CSS for institusjonen

Localized domain names

Institusjonsnavn på forskjellige sprog.

Person info

Inneholder persondata. I tillegg til feltene fra personimporten har man

- last update:** Sist sett-felt. (For ev. rydding)
- password:** Crypt, brukes av spesielle login-klasser (f.eks. `no.uio.valg.login.Md5Login`)

Person external id

Tenkt brukt til å forenkle import og personsøk. Bruk så langt er å legge inn brukernavn uten *@institusjon* for raskere søk.

Election

Valgmetadata:

- election type:** Valgtype
- name:** Navn/tittel
- info url:** Lenke
- election status:** Valgstatus (init, klar, stengt)
- start time:** Åpningstid
- end time:** Lukketid

num Antall å velge
chooseable:
num Antall varaer
substitutes:
contact (Fritekst+HTML) Hvordan velger kommer i kontakt med valgstyret
address:
enc key id: Krypteringsnøkkel (Heksadesimal fingerprint)

Når status=init, vil valget ikke være synlig for velgeren, og admin kan forberede. Når status=klar, dukker valget opp på forsiden for brukeren, og det er åpent i angitt periode. Admin har anledning til å gjøre justeringer (f.eks. legge til personer i manntall, utvide stemmefristen) mens valget er klart. Valget må så stenges før opptelling kan gjøres. Valget forsvinner fra forsiden etter en gitt periode.

Election group

Et styre har ofte representanter fra studenter, teknisk og administrativt ansatte og vitenskapelige ansatte. Disse velges gjerne samtidig fra ulike delmengder av velgermassen. Er dette tilfelle, brukes valggruppe. Systemet vil da skjule de andre valgene fra forsiden enn de brukeren har stemmerett i.

Election description

Valgbeskrivelse på forskjellige sprog

Election type code

Systemet gjør dispatch på valgtipe flere steder (se `no.uio.valg.ValgFactory` og filen `resources/no/uio/valg/FactoryConfig.properties`).

Permissions

ACL for systemet. Av historiske årsaker er det også en tabell for valgadministratorer.

Permission type code

Vi har to koder:

perm create: Bruker kan opprette valg. Domain kan begrense hvilke domener som er lovlig. Typisk en rettighet man kan gi til LITA'er.
perm admin: Adminrettigheter. Domain = null er av historiske årsaker superadmin (alle rettigheter)

Election admin

Valgadministrator. Av historiske årsaker egen tabell, men kunne fint vært i permissions-tabellen. TBD: Trenger vi også en observator-funksjon? (Altså lesetilgang)

Census

Manntall. Valgreglene våre lar enkelte valg være vektet mellom ulike velgergrupper, derfor kan et valg ha flere manntallslistes. Manntallene gis prioritet for å sortere ut muligheten for at personer havner i flere manntall (f.eks. ansatt som også er student).

Census member

Velger.

census pid: Velgerens brukerid. Kan legges inn i systemet uten å være i person info.

status: For rapportens del skilles det mellom *imported*, *added*, og *accepted*. Det viktige er å sortere ut *deleted*, som angir at velgeren ikke har stemmerett i manntallet.

tag: I dag ubrukt, men tenkt brukt til statistikk.

List election list

Liste for listevalg. Har navn, beskrivelse (på ulike sprog) og kandidater.

List election candidate

Kandidat tilknyttet listevalg. Listen har et ordnet utvalg kandidater, som hver kan være kumulert (ekstrastemme etter reglene). Det er også mulig med beskrivelse (fritekst) for f.eks. å angi fakultetstilhørighet.

Person election candidate

Knytter en person til personvalg. (Merk: Det viser seg at UiB har en hybridløsning: personvalg hvor forslagsstillerne kan foreslå enten lister eller enkeltpersoner. Velgeren kan enten velge en liste eller gjennomføre sine endringer. I så måte kunne det vært lurt å slå sammen kandidattabellene. Ellers er tabellen ganske lik `list_election_candidate`.

Person election substitute

Brukes i valg for team: Rektor stiller f.eks. med egne prorektorkandidater. Rent datamodellteknisk kunne også `list_election_list` vært brukt til formålet.

Poll option

Brukes i uravstemninger. Et alternativ har et navn (tittel) som er lik for alle, og en lokalisert beskrivelse.

Quota

Brukes for kjønnskvoltering. En rad i tabellen angir en regel om at tilhørende kandidater (`quota_member`) skal ha et gitt antall representanter.

Vote

Stemmemetadata. Selve stemmeseddelen lagres kryptert på disk med filnavn basert på `vote_id`. Dette skal forhindre at de som har tilgang til basen også kan se hva som er stemt.

pid: Hvem som har stemt. Når man laster ned stemmer, hentes det ut hvilket manntall stemmegiver kommer fra, og pid brytes.

received: Når det ble stemt. Redundant, men brukes til statistikk og ev kontroll mot loggen.

remote_ip: Hvor det er stemt fra. Brukes ev. ved mistanke om feil.

why_vote: Er stemmegiver ikke i manntallet, fyller han ut dette. Fritekst, og vises for valgstyret.

paper_ballot: Ikke i bruk. Tiltent en bruk hvor man kombinerer elektronisk valg med papirstemmer.

Filstruktur i repoet

Rune Frøysa utviklet versjon 1 i CVS. Det var da to CVS-trær: ett for «`valg_script`» (sic) og ett for «valg». Disse to repoene ble slått sammen i 2007 (ved å flytte historikkfilene) og så importert til Mercurial. Bygging var tidligere gjort med Rune Frøysas Eclipse. Dette ble gjort om til å bruke et ant-script.

I 2015 ble det gjort noe arbeid for å flytte filene i standard Mavenstruktur (og Gradle), og biblioteker ble flyttet fra `thirdparty`-mappen med manuelle instruksjoner til `build.gradle`, som bruker mavens avhengighetsstyring.

Det som gjenstår av dette er:

- `src/scripts`
- `src/sql`
- `thirdparty/README` bør få leve inntil alle er på versjoner som bygges av gradle, men skal på sikt bort.
- mappen `test/` skal vel flyttes til `src/test/...` etter hvert.

For å finne frem:

docs:	Dokumentasjon
libs:	Biblioteker som ikke finnes i maven
src/main/java/:	Javaklasser
src/main/resources/:	Tekster og <code>FactoryConfig.properties</code>
src/main/webapp/:	JSP-filer og web-configurasjon

Versjonsinnsetting

Under bygging kjøres scriptet `src/scripts/make_version.py`. Dette finner versjonsdata i repoet og setter inn i klassen `no.uio.valg.VersionInfo`. Hovedpoenget er å sette inn versjon (basert på tag som starter med «v» i navnet), årstall og revisjonsid (SHA1). Denne ble i utgangspunktet laget for mercurial, dog.

Kort om hva byggingen gjør

Byggingen laster først ned dependencies fra maven central repoet.

1. `versionInfo` setter inn versjon
2. `compileJava` kompilerer java-filer til `build/classes/main/`
3. `clientJar` pakker et utvalg java-klasser sammen med `FactoryConfig.properties`
4. `war` setter sammen bygde klasser med filer fra `resources` og `webapp`, samt legger til `evalg.jar`.

Build vil også kunne bygge det lille som er av tester.

I tillegg bruker vi `gradle-versions-plugin` som gir en `dependencyUpdates`-task, som ser etter nyere versjoner av biblioteker på nettet.

Ferdigbygde filer havner i `build/libs/`

Litt om Java servlets og Struts

Det fineste med Java er alle bibliotekene og standardiseringen av store deler av hva man trenger. Java har en standardmåte å lage websider på:

- Man pakker sammen en war-ball (zip-fil med spesifisert struktur og filenavn «.war», for *web archive*)
- War-filen gis til en *servlet container*, som f.eks. Resin eller Tomcat. (Det finnes flere også)
- `WEB-INF/classes/` legges inn i classpath av servlet container
- `WEB-INF/libs/` inneholder jar-filer som også legges inn i classpath

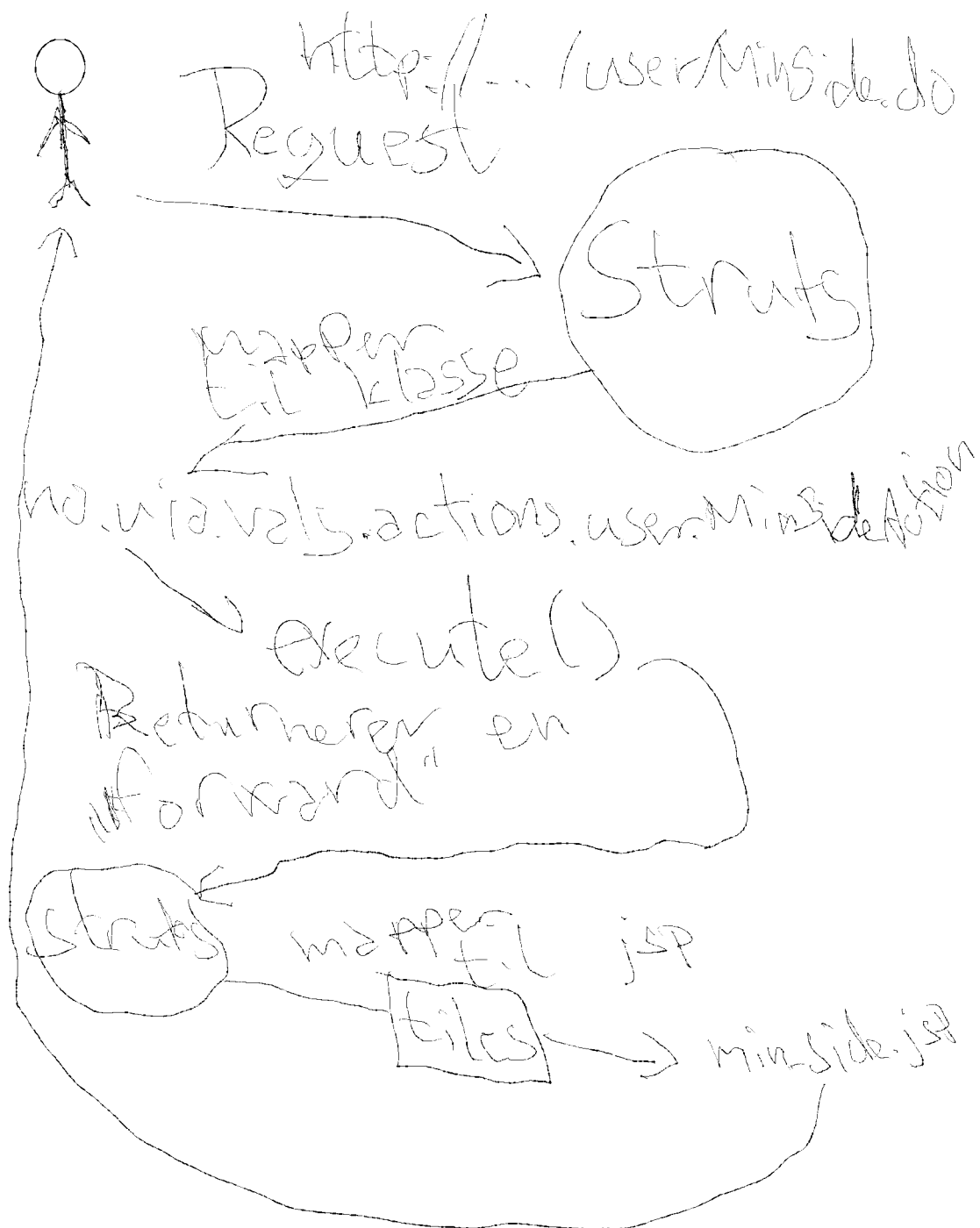
- WEB-INF/web.xml leses inn spesifisert format av Sun/Oracle/Java (kall det hva du vil).
- JSP-filer i warballen kompileres først til Java-kode, som så kompileres til java byte code.

I web.xml gjør vi følgende:

- Definerer en servlet for biblioteket struts. Alle requests til urler som matcher *.do i stien, vil sendes til org.apache.struts.action.ActionServlet, denne leser WEB-INF/struts-config.xml for å mappe stier til klasser.
- I tillegg har vi også en servlet for OIOSAML, som er en wrapper rundt OpenSaml. alt som havner under \$SERVLET_ROOT/saml/ sendes til OIOsaml's dispatcher.
- Struts definerer også egne tags i JSP-filene, gjennom taglibs.

Det som ellers ligger i roten av warballen vil serves som normalt av en http daemon.

Struts er ett av flere biblioteker som implementerer MVC-modellen i Java. Struts 1 har nådd end of life, så det er på tide å gå over til noe annet, f.eks. Struts 2



Vi bruker også Struts tiles. Dette brukes da til å sette inn jsp-dokumenter i andre. Vi har da `mainUserLayout.jsp` og `mainLayout.jsp` som inneholder felles-HTML for alle sider. Via `src/main/webapp/WEB-INF/tiles-defs.xml` konfigureres hver enkelt side med alle tile-elementer.

no.uio.valg.*

Pakken her inneholder overordnede elementer:

- CharsetFilter:** Et filter som opprinnelig setter UTF-8. Utvidet med logikk for å sende uautorisert bruker til access denied-side. Kalles av servlet container.
- ValgConfig:** Håndterer configfilen (properties).
- ValgFactory:** Diverse logikk for å dispatche basert på valgtype.
- VersionInfo:** Versjonsinfo. Opprettes under bygging.
- ValgLogger + VerySimpleFormatter:** Logging

no.uio.valg.utils

Pakken har diverse mikrobiblioteker:

- ConversionUtils:** Datoformattering og XML-flikking.
- GPGUtil:** Håndterer kryptering, nøkler osv.
- MailUtil:** Metoder for å sende standardmailene.
- PersonSearchUtil:** Generer SQL for å søke opp personer. (Ny feature, som foreløpig bare er brukt i adminsetting).
- SiteMessageResources:** Gjør instansspesifikke meldinger.
- SiteMessageResourceFactory:** For injection av SiteMessageResources

no.uio.valg.actions

Klassene her tilsvarer de ulike sidene i systemet: [https://.../\\$x.do](https://.../$x.do) mapper stort sett til klassen no.uio.valg.actions.\$xAction. Her kan \$x inneholde undermapper. Det skilles mellom underpakkene user og admin (se også no.uio.valg.CharsetFilter).

AbstractDBAction

Baseklasse som arver fra Struts' Action. Har diverse praktiske funksjoner.

LocaleAction

Håndterer sprogskifte i klienten.

LoginAction

Leser inn configvariabel for loginklasse og videresender request til angitt klasse i login-pakken. Ved innlogging finnes også beste sprog basert på innstillingene i browser.

LogoutAction

Håndterer utlogging. Note: SAML/singlesignonutlogging håndteres ved at SAML-biblioteket videresender til /Logout.do når alt annet er fullført.

LogoAction

Returnerer rett logo basert på brukerens Domain satt i innlogging.

user.AbstractUserAction

Praktiske funksjoner som ofte trengs i stemmegivningsprosess. Inneholder også konseptet Bucket, som mellomlagrer data i minnet under stemmegivningen. Denne hentes i sesjonen med en unik, tilfeldig id (for å vanskeliggjøre scriptangrep).

user.HandleXAction

Håndterer valgprosess av angitt type.

ApprovalElection:	Flertallsvalg (personvalg med avkryssing).
ListElection:	Listevalg.
Poll:	Uravstemning.
PreferenceElection:	Preferansevalg (STV)

Det finnes variasjoner i reglementer, og det fører til klasser navngitt etter institusjonen som først kom med kravet.

Videre kan X også inneholde:

Modify:	Til listevalg for å rette listen. I.e. kumulere, slenge, endre rekkefølge.
SubmitVote:	Viser frem stemmeseddel til kontroll før levering Pakker stemmeseddel i bucket og sender bruker til WhyVote eller SubmitVote
WoBlank:	Fordi Harald Baldersheim mente velgerne ikke burde kunne stemme blankt. Ham om det. Det går fremdeles an å stemme blankt, men det er ikke like enkelt.
WhyVote:	Lar brukeren begrunne hvorfor det er feil at hun ikke er i manntallet.

Disse bruker matchende jsp-filer:

user/*_show_note.jsp:	Vis stemmeseddel for kontroll
user/*_vote.jsp:	Stemmegivning
user/*_vote_cumulate.jsp:	Rette liste
user/why_vote.jsp:	WhyVote

user.HandleSubmitVoteAction

Leverer inn seddel, sender mail og viser kvittering. (user/vote_receipt.jsp)

user.ListProgrammeAction

Forsiden. Lister opp aktive valg, passive valg, kommende valg og avsluttede valg begrenset innenfor en viss tidsramme. (user/list_programme.jsp)

user.MenuAction

Informerer bruker om hvor langt han er i stemmegivningsprosessen. (user/submenu.jsp) Settes inn på siden via tiles og tiles-defs.xml.

user.ListElectionsAction

Den gamle forsiden, laget av Rune Frøysa. Såpass lite brukervennlig at professorer på medisin ikke klarer bruke den. :-) (user/list_all_elections.jsp)

admin.AdminController

Forsiden for administrator. Den eneste (?) klassen som ikke følger url til klassenavn-mappingen. Denne dispatcher på queryparametret do:

frontpage:	Ikke i bruk ennå. Tenkt som forsiden etter innlogging. (admin/frontpage.jsp)
go_list_valg:	Lister opp alle valg brukeren har tilgang til. (admin/list_all_elections.jsp)
go_detaljer:	Dispatcher på valgtypen for å legge inn - kandidater, (admin/change_person*_election.jsp) - lister (med kandidater) eller (admin/change_list_*.jsp) - uravstemningsalternativer. (admin/change_poll_option.jsp)
go_new_election:	For å sette opp et nytt valg. (admin/new-election*.jsp)
download_program:	Laster ned evalg.jar (opptellingsprogrammet)
download_zipvotes:	Laster ned stemmesedler

admin.ChangeDomainAction

For øyeblikket er ikke /admin/ChangeDomain.do linket til noe sted. Kan gjøre endringer på Domains:

- Logo
- CSS
- Navn
- Nøkler.

admin.NewElectionWizard

Setter opp nytt valg i bitvise steg.

Sider for redigering av valgdata

ChangeElectionInfoAction:	Metadata
ChangeElectionQuotaAction:	Kvoter
ChangeListElectionAction:	Listevalglistar
ChangeListCandidatesAction:	Listemedlemmer
ChangePersonElectionAction:	Personvalgkandidater
ChangePersonSubElectionAction:	Personvalgkandidater (Team)

ChangePollAction:	Uravstemningsalternativer
ChangeVotersListAction:	Redigere manntallsdefinisjoner
ChangeVotersListMembersAction:	Redigere manntallsmedlemmer

HandleNonmemberVoterAction

Håndterer personer utenfor manntallet.

ChangeElectionStatusAction

Lar admin endre fra nytt til klar, og fra klar til stengt. For stengte valg, også mulighet for å laste ned stemmesedler, evalg.jar og valgrapport del 1.

no.uio.valg.data

Klasser som representerer (i stor grad) tabellene. Fungerer også som beans til jsp-ene. Election-klassen har logikk for dispatch basert på valgstatus. Unntatt er vote, som ligger i vote-pakken, og Permissions, som ligger i auth-pakken.

no.uio.valg.vote

Stemmesedler og håndtering av disse. VoteContents-klassene inneholder stemmeseddeldata, samt logikk for å lese inn dette fra fil og skrive til fil.

VoteTransmitter håndterer filsystem og legger på kryptering.

VoteReceiver kontrollerer stemmeseddel og tilganger før stemmen håndteres av VoteTransmitter.

no.uio.valg.script

Funksjonalitet i evalg.jar. Av en eller annen grunn er dette både opptellingsprogram og «bakdør» inn i databasen. Disse to funksjonalitetene burde selvsagt vært adskilt.

Calculate*:	Teller opp valg av angitt type.
MainClass:	Inneholder main()
ProgressInfo:	Progressbar-interface laget for både cli og gui
ConsoleProgressBar:	Tegner [=====>] under opptelling.
VoteDecoder:	Leser stemmeseddelfiler (XML). Dytter deler av jobben til valgklassen.
QRCodeReader:	Leser inn strekkode fra bilde (passord til dekrypteringsnøkkel)
Counter:	Inneholder metoder som brukes av opptellingen: utskrift og loddtrekning.
gui.DialogProgressBar:	Lager vindu med progressbar.
gui.*Action:	Swing actions.
ImportPersonsTool:	Importerer personer i XML eller CSV-format. Mulighet for å legge inn callback, f.eks. for å legge inn i manntall.

Se også seksjon om opptelling av stemmer.

Mindre pakker

no.uio.valg.excptions:	Unntaksklasser.
no.uio.valg.auth:	Autorisasjonsdata.
no.uio.valg.html5:	Taglib-utvidelser for HTML5
no.uio.valg.login:	Login-klasser

Opptelling av stemmer

Opptelling av stemmer foregår ved å hente ned stemmer i en zip-fil:

`/admin/adm_ctrl.do?do=download_zipvotes&electionId=xxx` går til
«`no.uio.valg.actions.AdminController.handleDownload()`», og den bruker igjen klassen
«`no.uio.valg.script.ElectionResultExporter`», som henter ned alle stemme-id'er som skal med, stikker om på rekkefølgen, og lager en zip-fil:

electionInfo.xml:	Inneholder metainformasjon om valg og kandidater.
voteId2VotersListId:	Én linje per stemmeseddel: «stemmeseddelid/manntallsid», hvor stemmeseddelid er nummerert fra null og oppover.
<stemmeseddelid>.pgp eller .xml:	Stemmeseddel.

Opptellingsprogrammet (`evalg.jar`) kjøres på lokal maskin. `no.uio.valg.script.VoteDecoder.parseVoteZip()` leser inn `electionInfo.xml` og lager et `Election`-objekt av riktig type (basert på valgt type).

Stemmesedlene leses inn ved å sende XML-data til et `no.uio.valg.vote.VoteContents`-objekt av rett type.

Så telles stemmesedlene med parser.

Scripts

Mappen `src/scripts` (mappen har hett dette hele tiden, og ikke vært flyttet siden vi gikk over til standard maven-layout) inneholder en del praktiske scripts.

archive.py

Ble brukt for å lage tarballer av enkeltversjoner i repoet. Må endres for å fungere med git.

checkresources.py

Brukes til å kontrollere at meldinger er oversatt til alle sprog. Bør kanskje innkorporeres i automatiske tester.

CreateQRCode.java

Dette er vel mer et eksempel enn noe script. Brukes som backup om ikke QR-kode-biten i `gen_ark.py` fungerer.

election_stats.py

Dette skulle blitt et fantastisk script til å lage masse statistikk. Akkurat nå er det begrenset til å lage et gnuplot-diagram som viser når folk har stemt.

gen_ark.py

Lager LaTeX-dokument av passordene. Se avsnitt om nøkler.

Scriptet bruker Java-biblioteker for å lage QR-kode, og bør således kjøres med Jython og zxing-biblioteker i python-path.

generate_election_stats.py

Etter å ha dumpet manntallsdata med evalg.jar --dump-census, kan dette scriptet kjøres mot Cerebrum for å generere en rapport over oppmøte per OU. Det kan gis argumenter hvor mange nivåer ned i OU-strukturen man skal ta med, og/eller en grense for hvor få som kan være i en OU før den slås sammen med forelder-OU. Nyere løsning for samme problem vil være å bruke tags i manntallsdata.

import_user_manntall.py

Bør slettes

keygen.py

Genererer nøkkelpar med GnuPG. Outputter «new_key\d+.(pub|sec)», og en fil som inneholder passord.

mail_current_election_data.py

Kjører som cronjobb og sender mail med status for pågående valg til ønskede mottagere.

make_version.py

Nødvendig for bygging: Setter inn versjonsdata i Version.java basert på hvilken versjon som er utsjekket. Den kan også lese filen .hg_archival.txt, som genereres av kommandoen «hg archive».

migrate.py

Migrerer fra gammelt til nytt databaseskjema. Er vel bare av historisk interesse nå.

parse_log_for_logins.pl

Skriver ut hvem som logget inn mellom 12. april 2010 kl 15.23 og neste dag kl 8.35. Det kom av at en oppgradering av Struts gjorde at en av sidene ikke fungerte, og valgansvarlig ville ha en liste over berørte brukere.

test_election_counter.py

Dette er utkast/pilotversjon av forbedret opptelling av listevalg. Forbedringene er portet til Java og ligger inne i evalg.jar

test_nhh_pref_election.py

Teller opp preferansevalg etter NHHs reglement. (Utkast/pilotversjon). NHH har forøvrig et ganske interessant reglement.

test_pers_election_counter.py

Utkast/pilot til opptelling av flertallsvalg.