

ETORG - Elektronisk torg – versjon 1.02

Innholdsliste

1 BASIS funksjonalitet.....	1
1.1 Teknisk arkitektur, lagdeling og scopes.....	2
1.1.1 GUI lag.....	3
1.1.2 Domene (forretnings) lag.....	4
1.1.3 Persistens lag.....	5
1.2 Tekniske valg og hva som ble prioritert i eTorg.....	5
1.3 Kodetre og javadoc for eTorg.....	7
1.4 Funksjonalitet implementert.....	8
1.5 Nye ideer og forbedringer for eTorg.....	9
1.6 Kjøre miljø.....	10
1.7 Bruksanvisning.....	11
1.7.1 Velg produkt skjermbilde.....	11
1.7.2 Handlekurv skjermbildet:.....	12
1.7.3 Logg inn bildet.....	13
1.7.4 Kunde- og ordreinformasjon bildet.....	14
1.8 Hvordan se på, installere og kjøre applikasjonen.....	15

1 BASIS funksjonalitet

ETorg er slik den er implementert i dag en skisse til en nettbutikk.

Denne demo applikasjonen er skrevet i Java EE (JDK 1.6) /JSF 2.0 og inneholder funksjonalitet for:

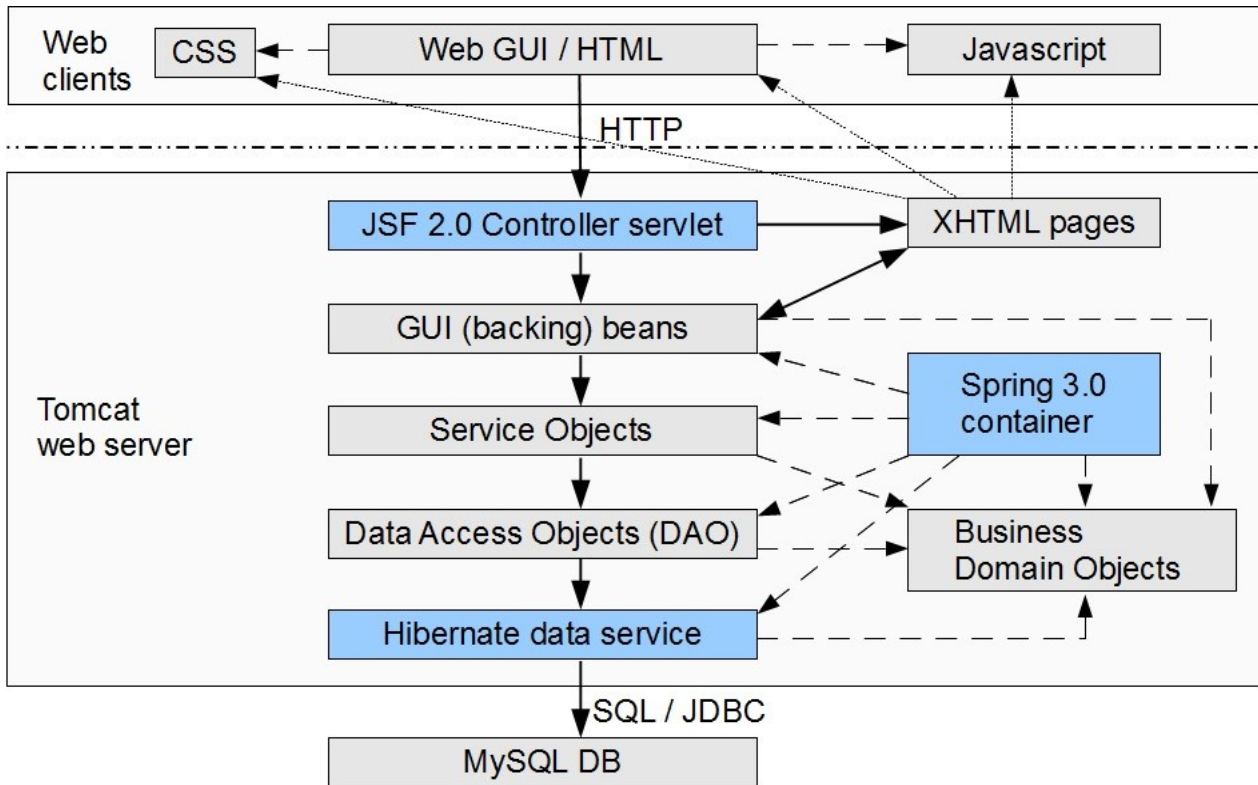
- Velg fra produktkatalog
- Handlekurv
- Logg inn med enkel passordhåndtering
- Definisjon av bruker (kunde)
- Ordrebestilling
- Enkel ordre administrasjon.

Det er lagt mest vekt på å teste ut nyeste versjoner av teknologi i denne demoen (JSF 2.0, Spring 3.0 og Hibernate). Det er brukt relativt enkle JSF 2.0 komponenter (ikke Ajax).

Grensesnittet for betaling er rimelig enkelt i mangel på kopling til web-tjeneste for betaling, kun trykk på en knapp og du har betalt. Produktene i denne versjonen er fiktive uansett, og ikke lagret i databasen i denne versjonen. De blir definert i produkt katalogen (ProductInventory) når denne opprettes i minnet.

1.1 Teknisk arkitektur, lagdeling og scopes

Web klientene (med eller uten Javascript) kaller Tomcat web server, som igjen kaller MySQL databasen. Se figur under. Web tjeneren og databasen ligger lagret på samme virtuelle (VPS) Linux server i det produksjonsmiljøet som er satt opp. Applikasjonen har på tjener siden et GUI lag, et begrenset domene lag (forretningslag) og et persistens lag. Persistens lagets databasetjenester blir også kalt opp direkte fra GUI laget. ETorg er derfor slik den er implementert i dag ikke en «sann» tre lags applikasjon, der laget over kun kjenner til laget under.



JSF og Spring benytter begrepet «scope», som i all enkelthet betyr hvor lenge en Java bønne lever i beholderen («containeren») sin. F.eks. "Request scope" betyr at brukeren i nettleseren sender inn en forespørsel til web-tjeneren. Dataene leses så inn via databasekall til en Java bønne (objekt), som igjen oversettes via XHTML (Facelets) til HTML og innholdet presenteres for bruker. Java bønna fjernes så fra minnet på web tjeneren. Dvs. denne lever bare fra brukeren trykker på knappen til brukeren har fått svaret sitt i web leseren. Ved neste forespørsel er bønna borte og en ny blir lest inn.

Til info fra JSF (og Spring):

- Application scope (felles for alle bruker av applikasjonen)
- Session scope (data lagres så lenge brukeren har samme web sesjon).
- View scope (data lagres så lenge man er i samme GUI bilde)
- Request scope (data lagres ikke over en HTTP request / response)
- Custom scope (egendefinert), f.eks. «Conversational» scope for en brukerdialog.

Alle disse kan benyttes i eTorg, men Application scope for deling av data mellom brukere er ikke brukt i dagens versjon. I den nåværende versjonen benyttes Request scope og Session scope.

1.1.1 GUI lag

GUI laget består av Java «backing» bønner som tilhører JSF skjermbilder (*.xhtml), CSS, noe Javascript og JSF Controller servlet. JSF Controller servlet håndterer alle forespørsler fra web-klienter. Se figur i avsnitt 1.1.

I denne versjonen er det valgt å legge Java «Backing» bønnene i Spring (3.0) container og ikke bruke JSF sin. Dette er først og fremst fordi man da kan bruke all funksjonaliteten til Spring. Spring har imidlertid ikke «View Scope» som standard, men dette kan legges til som et Spring «Custom Scope». Jeg fant en slik på nettet, men denne fungerer ikke 100% etter testing. «Request Scope» er derfor brukt i stor grad i eTorg. Et alternativ hadde vært å bruke JSF sin container for GUI laget og Spring's container for de andre lagene. Dette gir noe økt kompleksitet, men gir bedre separasjon av de tre lagene (GUI, domene og persistens-lag).

Jeg har testet ut JSF 2.0 «Flash minne» for å kunne beholde data gjennom en Post Redirect Get PRG syklus. Min erfaring er at dette er vanskelig å bruke og gir ofte uforutsigbar respons fordi man må være nøye med i hvilke JSF faser som Flash minnet aksesseres. Derfor er data heller lagt persistent i en Session bønne når de skal beholdes ut over en PRG syklus. Bruk av «View Scope» kan være et alternativ for slik lagring. Min mening er at manglende muligheter for enkel programmatisk styring av hvor lenge en bønne skal leve er en klar svakhet ved utvikling av web-applikasjoner. Det finnes web rammeverk som støtter dette direkte (Ikke JSF). Det finnes mye diskusjoner om dette temaet på nettet.

Handlekurven er laget for å kunne eksistere uten database og innlogging. Derfor er denne denne laget i «Session scope». Alle data som ikke skal lagres i databasen gjemmes dermed i denne. Jeg kunne selvsagt også laget en egen bønne for dette. Egentlig burde det vært brukt et «Custom scope» her også: Applikasjonen støtter multiple handlekurver, der referansen til handlekurv dataene må endres for å kunne bytte mellom handlekurver. Kun en handlekurv kan være transient per kunde.

Handlekurver som lagres i databasen blir definert som ordre. Statusverdier for ordren angir hvilken tilstand ordren er i (CART, DEFINED, PAID, SENT, RECEIVED). I denne versjonen følger vi ordren fram til kunden har betalt (PAID), etterfølgende tilstander brukes ikke.

Det er benyttet en facelet template (mal) for applikasjonen for alle sidene. Dette gjør at man kan endre på generell layout uten å endre på den enkelte side. Det er også gjort diverse forsøk på når en bør bruke html tabeller (panelGrid) eller html div (panelGroup og layout=block) taggen i kombinasjon med CSS. Jeg har nok kommet til at bruk av div for overordnet oppdeling av web siden gir det beste resultatet, gjerne i kombinasjon med float (left/right) og overflow (til scrolling). Det er imidlertid veldig mange feller å gå i ved avansert bruk av CSS og alle nettlesere fungerer ikke likt heller med hensyn til dette. Mye testing på ulike nettlesere er helt vesentlig for et godt resultat.

1.1.2 Domene (forretnings) lag

Domene laget definerer selve den objektorienterte eTorg modellen (databasetabell i parentes) og består av:

- Order (CART): Ordren/handlekurven.
- Product (PRODUCT): Produktene i ordren/handlekurven.
- ProductType: Produkttypen, egentlig ment mest for framtidige utvidelser og definerings av en persistent produktkatalog. Per i dag så arves disse egenskapene bare ned til Product.
- ProductInventory: Produktkatalogen.
- User (USER): Bruker inkludert kunde, som kan inneholde ordre med produkter.

Det finnes i eTorg ikke mye logikk i domene-laget, bort sett i fra enkelte beregnede attributter til bruk i JSF GUIet. Domene laget finner man igjen i «Business Domain Objects» i figuren i avsnitt 1.1.

For at eTorg skal bli en sann 3 lags applikasjon, burde det vært lagt inn enda en komponent mellom «GUI (Backing)» bønner og «Service Objects». For eksempel kan denne betegnes «Business Service Objects». «Service Objects» burde da ha endret navn til «Database Service Objects». En slik komponent blir da uavhengig av om det er brukt en database tjeneste til å implementere denne eller en web-tjeneste (web services). All transaksjons-håndtering vil da ligge på nivået under. Hvis det trengs mer avansert forretnings-logikk (som det ikke er i eTorg per i dag), vil denne komponenten benytte et sett med domene objekter («Business Domain Objects») til dette. Disse domene objekter kan da leve slik som i dag innenfor «Request scope», men andre «scopes» vil også være aktuelle avhengig av behov.

Relasjonene mellom objekter må være helt presist og riktig definert ved database transaksjoner, ellers får Hibernate som regel problemer ved generering av SQL-kode.

Datamodellen er slik: $(User_{(1)} \langle \text{---} \rangle_{(0..n)} Order_{(1)} \langle \text{---} \rangle_{(0..n)} Product)$, som tilsvare domene objektene med noe mindre frihetsgrad. Order kan f.eks. eksistere uten User i domene-modellen men ikke i databasen. Hibernate kaskade kall er definert mellom disse entitetene (delete_orphan og create_save) i domene-laget. Se også persistens laget under der selve kallene foregår.

1.1.3 Persistens lag

Persistens laget er todelt og består av:

- Tjeneste (service) lag.
- DAO (data access objekt) lag

I tillegg benyttes alt det som Hibernate har å by på som objekt-relasjons mapping mekanismer. Se figur i avsnitt 1.1.

Alle tjenester kjøres innenfor en transaksjon og er definert med Spring annotasjonen `@Transactional`. Dette gjør at Spring tar hånd om håndtering av unntak i tjenestelaget (feilede database kall) og unntak som skjer i transaksjonen må fanges opp på nivået over. Dette er veldig enkelt, men man mister litt muligheter til detaljert kontroll på denne måten.

Man må være forsiktig når man koder pga. Hibernate lat datalasting, slik at man ikke leser utenfor en transaksjon. Hvis dette skulle skje gir Hibernate unntak umiddelbart (forsøk på Hibernate kall utenom transaksjon), og dette ødelegger som regel også JSF GUIet. Det beste er å definere mest mulig til å skje innenfor en transaksjon, da unngår man dette problemet og gir Hibernate bedre muligheter til optimalisering av database kall. En forespørsel fra en web klient bør normalt tilsvare en transaksjon.

Tjenestene kaller så DAO laget der databasekallene utføres direkte. Tjeneste laget kan inneholde sammensatt databasekall innenfor en transaksjon. Hver tjeneste klasse tilsvare som regel domene-klassen på laget over.

Typisk så er CRUD (Create, read, update, delete) operasjoner implementert i DAO laget for hver sin DAO klasse som tilsvare et domene objekt. Disse kallene består enten av direkte Hibernate kall, eller bruk av Queries. Jeg har foretrukket å bruke Hibernate Criteria kall og ikke HQL (SQL) kall. Jeg er ingen fan av hardkodet SQL. Hibernate SessionFactory er inkludert i DAO laget (autowired).

Domene-objekter (Business Domain Objects) eller identifikatorer er brukt som parametre og retur verdier i disse kallene, eventuelt inkludert i lister der dette er naturlig.

1.2 Tekniske valg og hva som ble prioritert i eTorg.

Jeg har ikke lagt veldig mye vekt på fancy GUI. Men jeg har i siste versjon av eTorg innført generisk layout. Dette er gjort ved bruk av en facelet template og mer avansert bruk av CSS. Dette gjør at demoen også lettere kan brukes som mal for andre web-applikasjoner.

Det er lagt mye vekt på at selve arbeidsprosessen med å bestille varer skal være enkel å utføre for en kunde. Jeg har lagt mest vekt på å sy sammen applikasjonen gjennom alle lag og å få rammeverkene til å spille sammen på en god måte. Stylesheets (.css) er benyttet. ICEfaces komponenter brukt i stedet for JSF komponenter kan anbefales. Basis JSF komponenter er rett og slett ikke godt nok egentlig for en «seriøs» applikasjon.

Jeg mener at mest mulig av funksjonaliteten bør legges i Java og minst mulig (annet enn ren presentasjon) i JSF. Kanskje kan til og med bruk av WYSIWIG verktøy eller liknende basert på JSF gjøre web-design prosessen enklere. Jeg har ikke noe i mot å bruke verktøy i stedet for kode.

Tilsvarende har jeg fulgt samme filosofi for grensesnittet ned mot databasen.

Javascript gir muligheter for klient side validering. I den siste versjonen er dette innført i hele applikasjonen. Dette virker fint, selv om spesielt det å overføre data fra Javascript til JSF/Java er triksete (må bruke JSF skjulte inputfelter). Jeg er ingen fan av Javascript, gir muligheter for sikkerhetshull og fører også til dobbeltimplementering av den samme koden for validering av felter. Men jeg har sett at Javascript er så utbredt i web-applikasjoner i dag at jeg fant ut at jeg skulle prøve, og det kan i noen tilfeller gi bedre ytelse. *Jeg har valgt å følge den filosofien at hvis Javascript skal brukes i en web-side, så skal også web-siden fungere uten Javascript¹.*

Jeg har laget en generisk funksjon for validering av felter basert på regex («regular expressions»). Denne er implementert både i Javascript og i JSF (Java). Dette gir muligheten for validering basert på regex i ressursfiler (f.eks. en norsk og en engelsk variant av validering av postnummer).

Noe som jeg har observert i forhold til tidligere implementasjoner basert på JSF og Spring er at nyere versjoner av rammeverk og spesielt bruk av Java annotasjoner gjør at koden blir enkel og rimelig oversiktlig. Dette er en klar fordel ved utvikling og vedlikehold av programvare. Det er faktisk ganske mye funksjonalitet som er laget ved hjelp av relativt lite kode. Men noe av det som jeg har brukt mest tid på er oppsett og konfigurasjon av hele opplegget, så her får man mye gratis neste gang det samme skal gjøres i et liknende prosjekt.

Jeg har lagt vekt på å utnytte disse nye egenskaper ved de siste versjoner av Spring (3.0) og JSF (2.0), ved å f.eks. bruke @annotation i koden i stedet for alt for mye XML-konfigurasjon. I de fleste tilfeller vil dette gi en mer oversiktlig og lettere implementerbar applikasjon. Syntaksen for annotasjoner er også i de fleste tilfeller noe enklere enn tilsvarende XML-konfigurasjon, men man har noe mindre muligheter til detaljkonfigurasjon.

MySQL er valgt brukt som database. Jeg kunne sikkert ha brukt hvilken som helst relasjons-database. Det er valgt å bruke ORM teknologi og Hibernate er benyttet til dette. All SQL blir da generert. Jeg er ikke noe fan egentlig av å skrive mye SQL selv. Vedlikehold av denne koden har det med å bli dyr, selv om det sikkert er lettere å skrive mer optimal SQL på denne måten. Oppsettet er slik at hvis et tomt MySQL database skjema eTorg er opprettet, så vil Hibernate opprette nødvendige tabeller ved oppstart av web tjeneren. Dette er veldig fint for testing og demo formål, men hadde ikke gått i et produksjonsmiljø selvsagt.

Hibernate er magisk, men noen ganger får man også «magiske uforståelige» feilmeldinger. Et problem med Hibernate som jeg også tidligere har sett litt på, er at det krever en god del å få alle mulige Hibernate feilmeldinger oversatt til feilmeldinger som en bruker kan forstå. Denne utfordringen er nå løst.

Applikasjonen er språk-uavhengig og man kan lett bytte mellom norsk og engelsk. Dette gjøres ved å endre standard oppsett i valgt nettleser. Demoen støtter per i dag engelsk UK (en) og norsk bokmål (no). Se «messages» og «resources» filer. Det finnes i dag automatiske oversettere i de fleste nettlesere, men disse gir i mange tilfeller svært dårlige oversettelser («save» blir f.eks.

1) Etter nøyere testing har det vist seg at enkelte JSF tagger er helt avhengig av at Javascript er skrudd på i nettleseren. Dette gjelder h:CommandLink. Jeg har derfor brukt h:CommandButton i stedet med en layout på knappen som likner en link. Det er heller ikke mulig å sende inn deler av en web side (h:form) uten å bruke Javascript (enten ren Javascript er brukt eller Ajax inkludert i JSF 2.0). F.eks. hvis man endrer antallet av en vare direkte i handlekurven, må man enten sende en forespørsel til web-tjener eller bruke Javascript for å rekalkulere prisen. Dette har gjort at jeg har lagt inn en ekstra oppfrisk knapp som vises når Javascript er deaktivert i nettleseren.

oversatt til «redde» i Google oversetteren).

NB! Ett unntak angående språk-uavhengighet er produktene i produktkatalogen, fordi disse i denne versjonen kun er hardkodet.

PS: Disse ressursfilene styrer også om man bruker \$ eller £ eller euro for betaling f.eks., hvordan dato skal skrives ut, og man kan også justere for tidssone. Det eneste jeg har lagt merke til er at den måten JSF gjør dette på, gjør at angivelse av tidssone må endres med en time ved overgang fra sommer til vintertid. Rent generelt er dette med tidsangivelse internasjonalt faktisk ikke en helt enkel problemstilling.

1.3 Kodetre og javadoc for eTorg

Dette er et maven prosjekt, og oppbygning av kataloger er slik som standard satt opp for dette byggeverktøyet for web utvikling.

main/java/eTorg inneholder Java pakkene (se Javadoc).

main/java/resources inneholder melding og ressursfiler (norske og engelske og standard):

- messages*.properties filer definerer språkavhengige meldinger.
- resources*.properties filer definerer språkavhengige GUI ressurser.
- hibernate_mysql.properties fil definerer databasegrensesnittet til MySQL.
Hvis ikke tilsvarende fil finnes utenfor .war filen, brukes denne (se kapittel 1.8).
- dbconfig.xml definerer annet database og Hibernate oppsett som ikke endres så ofte.
- applicationContext.xml definerer Spring konfigurasjonen.

webapp katalogen inneholder:

- JSF 2.0 filer (.xhtml), inkludert eTorgTemplate.xhtml (mal fil)
- web konfigurasjons filer som faces.config.xml og web.xml).
- css filer i css katalogen.
- Javascript filer i js katalogen
- Grafikk i images katalogen

pom (maven byggefilen ligger på rot-katalogen: eTorg).

Javadoc er også generert og finnes i .zip fil i katalog doc (klikk på index.html), det står også litt mer om tekniske ting angående applikasjonen der.

1.4 Funksjonalitet implementert

Funksjonalitet implementert:

- Vis produkter (hardkodet produktliste).
- Legg i handlekurv.
- Endringer i handlekurv (slett valgt produktinstans, flere/færre produktinstanser, ...).
- Beregning av totalpris.
- Verifiser ordre data:
 - Eventuelt blir man bedt om å logge på.
 - Ordredata verifiseres ved at man åpner kunde bildet.
 - Lagrer også ordren/handlekurven.
- Pålogging og avlogging:
 - Logg på med eksisterende eller ny bruker.
 - Innlegging av nytt passord og muligheter for å endre passord.
 - Logg av (manuelt eller tidsstyrt fra Tomcat)
 - Visning av brukerens navn nederst.
- Lagre handlekurv (kun hvis logget på).
- Betal (forenklet, selve betalings GUI / API er ikke med).
- Ny handlekurv:
 - Opprettes enten ved oppstart av web-leser
 - eller ved å velge nye produkter når varene er betalt,
 - eller ved utlogging og etterfølgende innlogging.
- Se eksisterende ordre/handlekurver:
 - Hvis man ikke har valgt produkter før man logger inn, kan en bruker se sine eksisterende ordre ved å velge «verifiser ordredata». Da vil en ny tom handlekurv heller ikke opprettes.
- Kunde bildet (din kunde informasjon og ordre):
 - Bekreft kundeinformasjon.
 - Vis status for flere ordre/handlekurv.
 - Gå til handlekurv (ordre), eventuelt bytt handlekurv (ordre).
 - Slett handlekurver (låste handlekurver kan ikke slettes, gjøres direkte i DB)
 - Detaljert validering av felter (klient- og tjener-side, regex basert)
 - Bekreft boks (i Javascript) hvis man logger av og kundedata er endret
- Muligheter for engelsk/norsk/internasjonal variant ved konfigurering (ikke programmering):
 - Gjelder nå alle type ressurser, f.eks. ledetekster og regex uttrykk.
 - Gjelder nå alle type meldinger og feilmeldinger (egendefinerte, JSF, Hibernate).
 - Styres av standard oppsett i nettleser, eller av bruker.
- God feilhåndtering ved forventede feil og unntakssituasjoner:
 - Tilpassede feilmeldinger på riktig språk for kjente feilsituasjoner/unntak.
 - Fatale meldinger som ikke skal skje (spesielt Hibernate) på engelsk med originalinnhold. Dette gjør det lettere for utviklere å rette opp feil ved direkte brukerkontakt.
- Logging ved bruk av log4j: I denne versjonen logges det kun til konsollet, men det er bare snakk om konfigurasjon av log4j.properties filen for å endre dette.
- Automatisk lagring av data når man går ut av skjermbilder (kunde bildet og handlekurven), hvis noe er endret. Applikasjonen er designet slik av bruker normalt ikke trenger å bruke lagre knappene.
- Integrasjon med hjemmeside og bruk av template for å lage denne og eTorg.

1.5 Nye ideer og forbedringer for eTorg

Områder med forbedringspotensiale for å gjøre dette til mer enn en enkel demo:

- Bedre layout generelt (Icefaces) og andre ting.
 - Bytt spesielt ut tabellkomponenten med en Icefaces/Ajax komponent.
 - Se på profesjonelle handlekurver og de er mer visuelle (bilder av varene, ..).
- GUI Backing-bønnene skal ikke kalle database-tjenester direkte:
 - Dette betyr innføring av ekte 3 lags arkitektur, og at man går via domene laget.
 - Håndtering av unntak: Skjer i GUI laget (ref @Transactional), liker ikke dette.
 - Bruk av JSF View scope i stedet for Session scope
(har testet et Spring Custom View scope, fungerer ikke 100%)
 - En fordel med dagens versjon er at 3 lag trolig vil forbruke noe mer minne.
- Bedre sikkerhetsløsning (passord sendes i klartekst, diverse):
 - Innføring av Digest sikkerhet (kryptering av innlogging/passord).
 - Innføring av SSL (kryptering av innlogging/passord/innhold).
Digest (HTTP Digest autentisering med MD5) er det letteste under forutsetning av at det støttes av aktuelle nettlesere, men f.eks. et eventuelt kontonummer vil ikke krypteres med den løsningen. Egne tabeller for brukernavn, passord og roller må sannsynligvis innføres for begge disse løsningene.
- Grensesnitt muligheter mot betalingstjenester.
- Oppfølging av sendte ordre (GUI og grensesnitt)
- Innføring av administrator brukere med vide rettigheter.
 - Blant annet for sletting av brukere og resetting av passord etc.
- Produkt type definisjons GUI og DB tabeller (for administrator brukere).
- Søkebilde for produkt typer:
 - Søk på en eller flere kolonner, som Name, Country and Manufacturer.
 - Søk innenfor en kolonne kan være eksakt eller omtrent (wild card *, ..).
 - Når man angir søk på flere kolonner er det best å bruke unionen av kriteriene, dvs.
Country = .. AND Manufacturer = .., ikke Country = .. OR Manufacturer = ..
- Automatisk lagring når man går ut av skjermbilder baserer seg på et flagg som blir satt ved JSF «change events» eller når bruker trykker på visse knapper. Det hadde vært bedre å gjort dette til funksjonalitet implementert i persistens laget. Jeg mener at Hibernate muligens har slik funksjonalitet, men har foreløpig ikke klart å finne ut av dette.

1.6 Kjøre miljø

Programmet er testet på følgende lettvekts web-servere: Jetty 6 (Eclipse innebygd) og Jetty 7, og også Tomcat 6 / 7; og skal virke på alle disse versjoner av web-servere. Det rare er at med den nye versjonen av Eclipse (Helios) og Eclipse WTP (Web tools plattform), så klarer jeg fortsatt ikke kjøre med annet enn den gamle Jetty 6 innebygget i Eclipse under utvikling. Produksjonsmiljøet bruker Tomcat 6. Jeg har lagt merke til at ved bruk av Tomcat 6 i produksjon, så klarer programmet ikke alltid å finne korrekt språk-oppsett (en svakhet med Tomcat 6 i forhold til Tomcat 7).

Jeg har også laget en variant av Jetty 7 web server som er en kjørbare jar fil, som gjør at det er lett å demonstrere eTorg uten installasjon av web server. Den leter da etter .war filer som inkluderes i Jetty web serveren. Men husk at du likevel må ha en relasjonsdatabase installert for å kunne kjøre all eTorg funksjonalitet. Tidligere var det ikke noen database i eTorg og da var hele greia noe enklere.

1.7 Bruksanvisning

De vedlagte skjermbildene viser den norske varianten. Man skifter mellom norsk og engelsk variant ved å endre standard oppsett i nettleser (fra no til en).

1.7.1 Velg produkt skjermbilde

Velg blant våre produkter.

- «Antall» setter antall produktinstanser av et gitt produkt som velges
- Produktene må velges med «Velg» for å kunne legge til handle kurv med «Legg valgte i handlekurv».
- «Fjern valgte» fjerner valg
- «Legg til valgte» legger varer i handlekurven.
- Gå til handlekurv med «Til handlekurv». Opprinnelig gikk man direkte til handlekurv etter «Legg til valgte», men brukertesting og erfaring med virkelige handlekurver gjorde at dette ble droppet.

The screenshot shows a web browser window with the URL `www.sigmondsmart.com/eTorg/productInventory.xhtml;jsessionid=A04CD9D35E39E4ECB75696D220A4C8FE`. The page title is "Velg blant våre produkter" and it includes a copyright notice for Hans Reier Sigmond. On the left, there is a sidebar with "Reier's linker" containing links to SigmondsMart, eTorg, eTorg dokumentasjon, Linked-in, Facebook (eTorg), Lillehammer o-klubb, Ringsaker o-klubb, Lucky Næroset, Vær Moelv, and reier-s@online.no. The main content area features a table with columns: Navn, Beskrivelse, Produsent, Land, Pris, Velg, and Antall. The table lists products such as First Price bananer, Dole bananer, Fairtrade bananer, Gravensten, Pink Lady, Lett Gulost, and Ekte Geitost. Below the table are buttons for "Legg valgte i handlekurv", "Fjern valgte", and "Til handlekurv". The total price is shown as "TOTAL PRIS: 0.00 €". A disclaimer box at the bottom states: "Demo programvare skrevet av Hans Reier Sigmond. - Java EE/JSP2.0/Spring 3.0/Hibernate/oss/JavaScript. - Alle produkter, betaling og forsendelse er virtuelle. - Ingenting blir lagret i MySQL databasen med mindre du definerer deg som bruker. - Sikkerhetslåsningen er foreløpig relativt enkel, så ikke gjenbruk passord fra andre steder. - Kontakt forfatteren med epost hvis du er interessert: reier-s@online.no - SigmondsMart er lagret på en VPS tjener hos Oxoos og jeg er web master." At the bottom of the browser window, it says "Javascript validering aktivert" and the language is set to "Norsk".

Navn	Beskrivelse	Produsent	Land	Pris	Velg	Antall
First Price bananer	banan	Kiwi	Costa Rica	3.00 €	<input checked="" type="checkbox"/>	3
Dole bananer	banan	Dole	Kongo	3.80 €	<input checked="" type="checkbox"/>	1
Fairtrade bananer	banan	Fairtrade	Sør Afrika	4.20 €	<input checked="" type="checkbox"/>	1
Gravensten	eple	Bama	Norge	3.60 €	<input type="checkbox"/>	1
Pink Lady	eple		New Zealand	6.00 €	<input type="checkbox"/>	1
Pink Lady	eple		Spania	6.00 €	<input type="checkbox"/>	1
Lett Gulost	gulost	Tine	Norge	3.70 €	<input type="checkbox"/>	1
Ekte Geitost	brunost	Tine	Norge	3.70 €	<input type="checkbox"/>	1

1.7.2 Handlekurv skjermbildet:

- Alle knapper er i dette tilfellet aktive, styres av status verdi.
 - Du kan ikke bestille før kunde og ordre informasjon er verifisert.
 - Du kan endre på handlekurven / ordren og legge til og fjerne produkter når som helst.
 - Men du kan ikke endre på hvilke/antall bestilte produkter hvis ordren er bestilt / betalt. (I praksis finnes det handlekurver på nettet som delvis tillater dette).
- Du kan legge inn en beskrivelse eller kommentar til ordren.
- «Lagre» lagrer handlekurven/ordren.
- Antall kan settes inn i «Antall» kolonnen, eller ved bruk av «Øk/minsk valgte».
- «Valgte av/på» styrer hvilke produkter som kan slettes/øke eller minske i antall.
- «Velg produkter» går til produktvalg skjermbildet.
- «Verifiser ordredata» lagrer ordren og går til kunde- og ordreinformasjons-bildet.
 - NB! Hvis man ikke er logget på gjøres dette først i logg inn bildet.
- «Bestill produkter» bestiller og «betaler» produktene (forenklet betaling).
- Total pris rekalkuleres hvis det er gjort endringer.
- Status kan endre seg fra «Handlekurv» til «Definert» (ordren eksisterer og er lagret i databasen) til «Betalt» (og bestilt).

DIN HANDLEKURV eTorg™ ©Hans Reier Sigmond

Navn	Beskrivelse	Produsent	Land	Pris	Antall	Total	Velg
First Price bananer	banan	Kiwi	Costa Rica	3.00 Nkr	<input type="text" value="3"/>	9.00	<input checked="" type="checkbox"/>
Dole bananer	banan	Dole	Kongo	3.80 Nkr	<input type="text" value="1"/>	3.80	<input checked="" type="checkbox"/>
Fairtrade bananer	banan	Fairtrade	Sør Afrika	4.20 Nkr	<input type="text" value="1"/>	4.20	<input checked="" type="checkbox"/>

Slett valgte Øk valgte Minsk valgte Valgte av/på Rekalkuler

TOTAL PRIS: 17.00 Nkr

Beskrivelse

Endret dato: 09:03:22 18.12.2010
Status: Handlekurv

Velg produkter Verifiser ordredata Lagre Bestill produkter

Javascript validering aktivert Norsk

1.7.3 Logg inn bildet

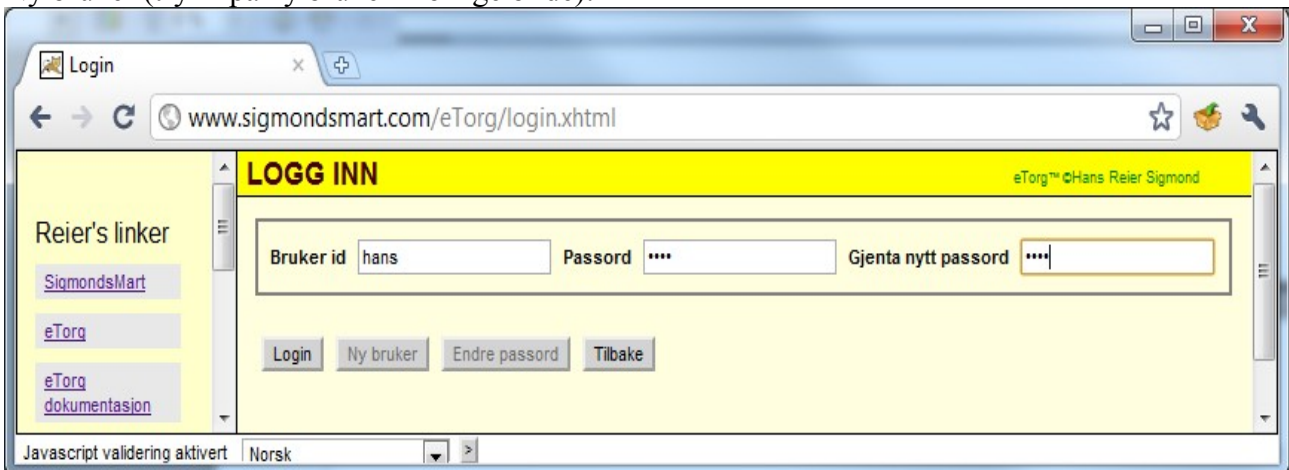
- Logg inn: Som eksisterende bruker.
- Ny bruker: Hvis du ikke har dette.
- Tilbake: Ikke logg inn, returnerer til handlekurven.
- Endre passord mulighet.

Etter vellykket innlogging går brukeren automatisk til kunde- og ordreinformasjons- bildet.

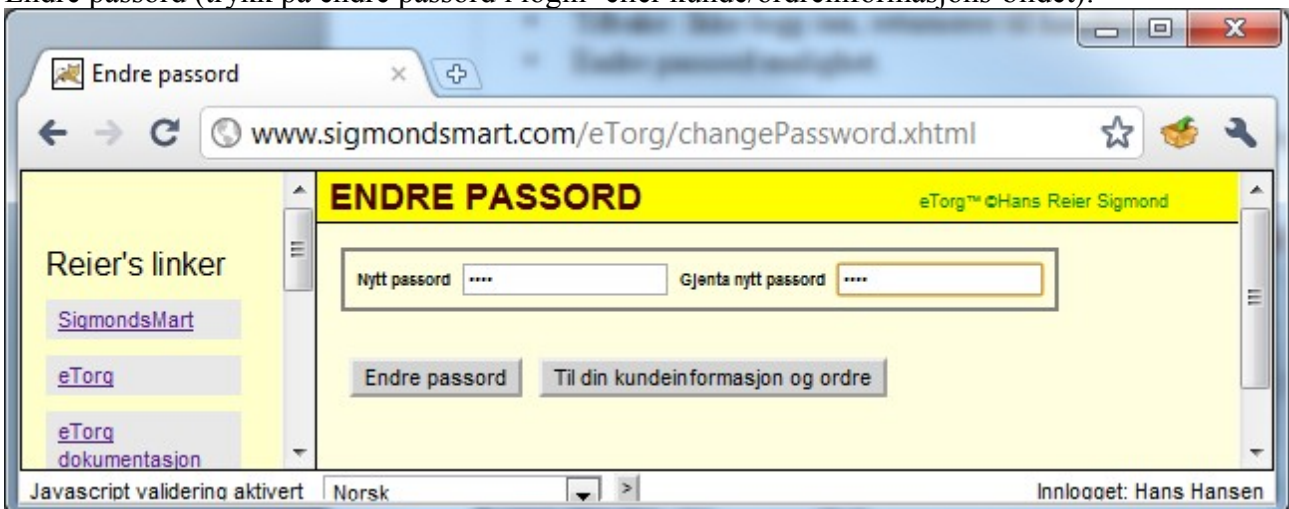
Eksisterende bruker:



Ny bruker (trykk på ny bruker i forrige bilde):



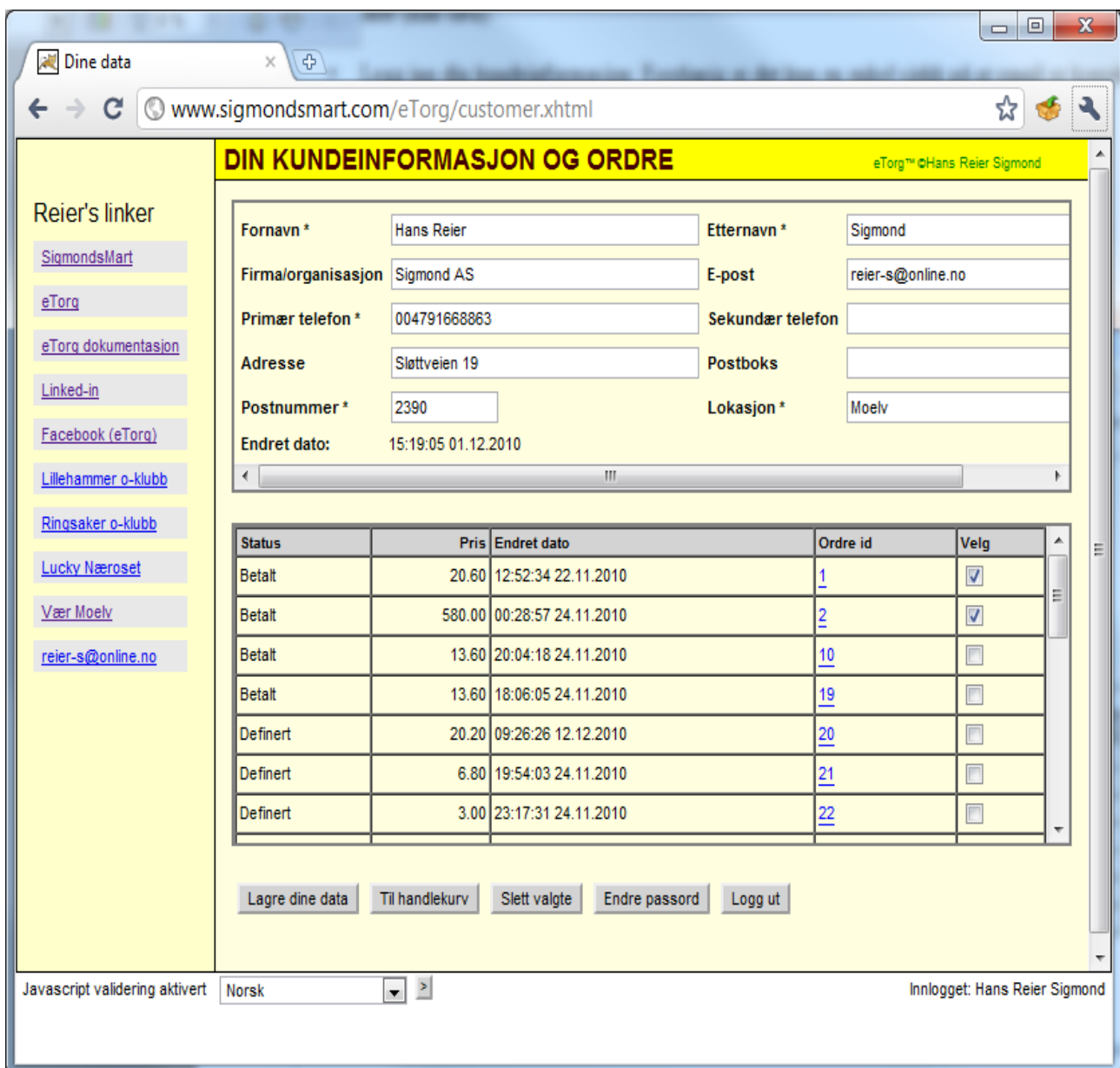
Endre passord (trykk på endre passord i login- eller kunde/ordreinformasjons-bildet):



1.7.4 Kunde- og ordreinformasjon bildet.

Viser status på ordrene, totalpris, endret dato og ordre identifikator. Ordre kan også velges for bulkoperasjoner (kun slett).

- Legg inn din kundeinformasjon: Foreløpig er det kun en enkel sjekk på at email er korrekt.
- «Lagre dine data» Lagre dine kundedata.
- «Til handlekurv» går til handlekurv.
- «Klikk på ordre linken» tar opp (en annen) ordre og går til handlekurv.
- «Slett valgte» sletter valgte ordre (ikke de som er betalte).
- Ordre id er genererte Java Ider for unik-het, derfor så lang tekst, forbedringspotensiale!
- Logg ut



The screenshot shows a web browser window with the URL www.sigmondsmart.com/eTorg/customer.xhtml. The page title is "DINE KUNDEINFORMASJON OG ORDRE" and the user is identified as "eTorg™ ©Hans Reier Sigmund".

Reier's linker

- [SigmondsMart](#)
- [eTorg](#)
- [eTorg dokumentasjon](#)
- [Linked-in](#)
- [Facebook \(eTorg\)](#)
- [Lillehammer o-klubb](#)
- [Ringsaker o-klubb](#)
- [Lucky Næroset](#)
- [Vær Moelv](#)
- reier-s@online.no

DINE KUNDEINFORMASJON OG ORDRE

Endret dato: 15:19:05 01.12.2010

Fornavn *	Hans Reier	Etternavn *	Sigmund
Firma/organisasjon	Sigmund AS	E-post	reier-s@online.no
Primær telefon *	004791668863	Sekundær telefon	
Adresse	Sløttveien 19	Postboks	
Postnummer *	2390	Lokasjon *	Moelv

Status	Pris	Endret dato	Ordre id	Velg
Betalt	20.60	12:52:34 22.11.2010	1	<input checked="" type="checkbox"/>
Betalt	580.00	00:28:57 24.11.2010	2	<input checked="" type="checkbox"/>
Betalt	13.60	20:04:18 24.11.2010	10	<input type="checkbox"/>
Betalt	13.60	18:06:05 24.11.2010	19	<input type="checkbox"/>
Definert	20.20	09:26:26 12.12.2010	20	<input type="checkbox"/>
Definert	6.80	19:54:03 24.11.2010	21	<input type="checkbox"/>
Definert	3.00	23:17:31 24.11.2010	22	<input type="checkbox"/>

Lagre dine data Til handlekurv Slett valgte Endre passord Logg ut

JavaScript validering aktivert Norsk Innlogget: Hans Reier Sigmund

1.8 Hvordan se på, installere og kjøre applikasjonen

Nytt fra den 22.11.2010 er at applikasjonen er lagt på nettet og kan kjøres fra:

<http://www.sigmondsmart.com/eTorg> eller indirekte fra <http://www.sigmondsmart.com>.

Kodetreet inkludert Javadoc er pakket i en zippet fil eTorg.zip. Se kapittel 1.3.

Jetty eller Tomcat kan installeres og startes opp med war fil eller eller utpakket war fil, eller eventuelt innebygd i Eclipse (Tomcat 7 eller Jetty 6).

Det er nå laget en mulighet for å kjøre applikasjonen hvis Java 6 (JRE 1.6) er installert på PCen uten at man trenger å installere en web server.

MySQL eller en annen relasjonsdatabase må være installert for å kjøre eTorg. Jeg har brukt versjon 5.2 av MySQL for å kjøre applikasjonen. Eldre versjoner kan sikkert også brukes. MySQL kan med fordel kjøres som en Windows-tjeneste. Database skjemaet etorg må opprettes, og pass på at informasjonen i filen hibernate_mysql.properties stemmer overens med definisjonen i databasen.

```
jdbc.driver.class=com.mysql.jdbc.Driver
jdbc.hibernate.dialect=org.hibernate.dialect.MySQLDialect
jdbc.url=jdbc:mysql://localhost:3306/etorg
jdbc.username=....
jdbc.password=....
```

Kjør filen startJetty.bat fra et DOS vindu, som starter opp eTorg. Denne filen finner hibernate_mysql.properties (må ligge på samme katalogen som startJetty.bat), og starter opp web serveren. Du kan også studere innholdet i .bat filen:

```
@echo off
set ETORG_DIR=%CD%
java -jar jettyServer-1.one-jar.jar
```

ETORG_DIR er en miljøvariabel som angir absolutt sti til hibernate_mysql.properties. %CD% bare finner stien til der du er nå. Du kan lett endre dette i bat filen til hva du måtte ønske. Web serveren startes opp med jettyServer-1.one-jar.jar..

Ta opp en web-leser og skriv <http://localhost:8080/eTorg> eventuelt <http://localhost:8080/JSFReg> (et annet enkelt testprogram som ikke aksesserer noen database). Hvis dette skal virke så må eTorg.war være på samme katalog eller i en underkatalog til jar filen. Tilsvarende gjelder for JSFReg.war (det andre testprogrammet). hibernate_mysql.properties må ligge i samme katalog som jar filen også.

Avslutt web serveren med CTRL C i operativsystem vinduet. Når web-serveren starter opp vil du kunne se at den leter og finner war filene. Deretter kommer en rekke meldinger. Ignorer glatt disse meldingene og se på web-leseren i stedet. Hvis noe har gått feil, kan du jo studere disse meldingene seinere.

Når web tjeneren starter opp skal nødvendige databasetabeller bli opprettet automatisk.