Design av IDE for presentasjonsmønstre i DPG 2.0

Hvordan utvikle verktøy for å utvikle presentasjonsmønstre

Jostein Bjørge

Institutt for informatikk
Universitetet i Bergen
Norge

Lang Masteroppgave
2010

0.1 Forord

Hei og hoppp [4]...

Innhold

Figurer

Tabeller

0.2 Definisjoner og Akronymer

- CMS - Content Management System
- DPG - Dynamisk PresentasjonsGenerator
- DSL - Domain Specific Language
- IDE - Integrated Development Environment
- JAFU - JAva for FjernUndervisning
- PDE - Plugin Development Environment

Kapittel 1

Innledning

1.1 Bakgrunn

Denne oppgaven er skrevet i tilknytning til JAFU-prosjektet (JAva for Fjern-Undervisning) ved Institutt for informatikk, Universitetet i Bergen. Dette prosjektet ble startet opp i 1998, og var da kjent under navnet Bergen Webucator. Prosjektets mål er å kunne tilby en rekke programmeringskurs som fjernundervisningskurs. Senter for videre- og etterutdanning ved UiB (SEVU) er også involvert i dette prosjektet. Sentralt i JAFU-prosjektet er Dynamic Presentation Generator (DPG). DPG er et generisk innholdshåndteringssystem (CMS) som benytter seg av et konsept kalt presentasjonsmønstre.

Et presentasjonsmønster følger en presentasjonsmønsterspesifikasjon [3], og definerer hvordan en web-presentasjon ser ut og oppfører seg, samt strukturen til informasjonen på nettsiden. Et av hovedpoengene er å fremme gjenbruk av denne informasjonen ved at en informasjonsstruktur kan presenteres på flere forskjellige måter. Jeg vil i denne oppgaven ta for meg utviklingen av presentasjonsmønstre, og hvordan utviklingsprosessen kan forenkles.

1.2 Motivasjon

Utviklingen av presentasjonsmønstre har vært en manuell prosess som har vært svært tidkrevende og vanskelig. Ved hjelp av et IDE for utvikling av presentasjonsmønstre, og å «lære» IDE'et hvordan presentasjonsmønstrene skal være bygget opp, er målet å forenkle hele utviklingsprosessen for presentasjonsmønstrene. Det vil kunne gi økt produktivitet, enklere feilsøking, og økt kvalitet på sluttproduktet. Presentasjonsmønstrene er modulært bygget opp, og disse modulene har et gitt hierarki og en gitt sammenheng som bør kunne automatiseres.

Jeg vil utforske andre muligheter for å definere presentasjonsmønsterspesifikasjonen på en måte som gir kodefullføring og syntaksfargelegging av koden. Ved å utforske hvordan jeg kan lage forskjellige veivisere (wizards) som hjelper til med å sette opp strukturen til et presentasjonsmønster ut i fra input gitt av brukeren av pluginet, er målet at det skal bli lettere å komme i gang med utviklingen av presentasjonsmønstere.

Jeg har valgt å lage et plugin til Eclipse, siden dette er et verktøy som jeg

er godt kjent med, som er open source, og som er det verktøyet som oftest blir brukt på dette prosjektet.

Nytt i Eclipse 3.5 som ble lansert 24. juni 2009 er et verktøy som heter Xtext [1] som er ment for å designe domenespesifikke språk. Xtext har et eget syntaks for å definere grammatikken til et språk, og vil deretter gi tilbakemeldinger på om koden som blir skrevet oppfyller kravene til strukturen som er gitt. Ut i fra dette blir det generert en egen editor for språket som gir kodefullføring og kryssreferanser, og det er støtte for autogenerering av kode gjennom for eksempel veivisere. Jeg vil se på muligheten for å bruke Xtext til å utvikle den ønskede funksjonaliteten i et plugin til Eclipse 3.5 som et domenespesifikt språk. Det er nå uvisst om Xtext er det riktige verktøyet å bruke til denne typen XML-baserte spesifikasjoner.

1.3 Målsetninger

Jeg vil her gi en oversikt over målene jeg ønsker å oppnå med denne oppgaven.

1.3.1 Hovedmål

Oppgaven min faller inn under Design Science. Jeg skal lage et design som forenkler arbeidet ved utvikling av presentasjonsmønstre. Resultatet av dette designet skal være en softwarepakke bestående av et verktøy basert på Eclipse og en plugin. Hovedmålet er at dette verktøyet skal gi brukerne en klar fordel i forhold til hvordan utviklingen foregikk tidligere og tilby blant annet:

- Økt effektivitet
- Økt produktivitet
- Bedre kvalitetssikring
- Bedre brukervennlighet

1.3.2 Delmål

For å oppnå hovedmålet med oppgaven er det mange ting som må utforskes og vurderes underveis.

1.3.2.1 Utforske verktøy og teknologier

Et viktig delmål er å utforske hvilke forskjellige verktøy og teknologier jeg kan benytte for å implementere den ønskede funksjonaliteten på best mulig måte. Jeg skal vurdere om det er formålsmessig å definere presentasjonsmønsterspesifikasjonen som et domenespesifikt språk, eller om det finnes bedre måter å få dette til på.

1.3.2.2 Designe et godt oppsett av Eclipse

Jeg må finne ut hvilket oppsett av Eclipse som gir best oversikt over presentasjonsmønsterutviklingen, og som gir en god arbeidsfly i forhold til utviklingen. Under utviklingen av presentasjonsmønstre jobber man med mange filer samtidig, og det er viktig å kunne strukturere disse på en god måte.

1.3.2.3 Definere arbeidsområde

Jeg skal lage et verktøy for utvikling av presentasjonsmønstre. Dette er en prosess som inneholder mange verktøy og teknologier. Jeg må definere hvilke områder av dette jeg skal konsentrere meg om. Det finnes for eksempel allerede verktøy for å skrive xslt-kode, og velocity-plugin som har god nok syntaksfargelegging. Det må vurderes hvor tett disse skal kobles med verktøyet for presentasjonsmønsterutvikling.

1.4 Metode

Jeg vil benytte en kvalitativ framgangsmåte for å evaluere designet. Hovedparten vil bli gjennom «Proof of Concept», ved å vise til det utviklede verktøyet. Jeg vil utvikle flere store og små presentasjonsmønstre for så å vurdere realisasjonen av disse mot en kravspesifikasjon som er definert for prosjektet. Dokumentasjonen fra utviklingsprosessen vil også bli et viktig produkt, for å gi nye brukere en god innsikt i systemet og gjøre de i stand til å ta det i bruk for å utvikle nye presentasjonsmønstre, og også for å kunne videreutvikle verktøyet mot nye krav. Både verktøyet og dokumentasjonen kan eventuelt evalueres ved å observere nye brukere benytte verktøyet i utvikling av mønstre, og tilbakemeldinger fra disse brukerne. Det vil også være en fordel å intervjue brukere om verktøyet med hensyn til brukergrensesnittet, og arbeidsflyten.

Kapittel 2

Kravspesifikasjon

2.1 Funksjonelle krav

2.2 Ikke-funksjonelle krav

Kapittel 3

Presentasjonsmønstre i DPG 2.0

Jeg vil her gi en oversikt over hvordan presentasjonsmønsterspesifikasjonen i DPG 2.0 er definert, og hvilke verdier som kan settes for de forskjellige elementene. Dette er basert på arbeidet til Peder Skeidsvoll og meg i inf219, prosjekt i programmering, våren 2009 [2].

3.1 Presentasjonsmønsterspesifikasjonen

Presentasjonsmønsterspesifikasjonen består av fire hovedelementer

- Entity
- Entity-instance
- View
- Page

Et page-element består av ett eller flere views, og views kan forekomme i en eller flere page-elementer. En entity-instance kan forekomme i flere views, men et view kan bare inneholde ett entity-instance element. Entity-instance kan bare inneholde ett Entity-element, mens et Entity-element kan gjenbrukes i flere Entity-instance-elementer. Videre kan en Entity inneholde andre Entity-elementer.

3.1.1 Entities

En entity definerer strukturen til informasjonen som skal bli lagret, og inneholder ingen informasjon i seg selv. En entity må inneholde en unik id, og en eller flere fields. Hvert field har minimum et navn, og en attributt som deklarerer hva slags informasjon den gitte fielden inneholder.

3.1.1.1 Oversikt over fieldverdiene i DPG 2.0 for entity-elementer

string Benyttes for enkle strenger, som ikke trenger spesiell behandling. Attributter: required="true"; sikrer at verdien blir satt av bruker allowedValues;

begrenser hvilke verdier strengen kan inneholde. Våren 09 er dette ennå ikke implementert i DPG 2.0

xhtml Benyttes for tekst som skal benytte seg av formatering av tekst, lister, tabeller osv. Attributter: required="true"

date Benyttes for enkle datoer, redigeres ved hjelp av en datovelger i Presentation Content Viewer. Attributten required er ikke støttet for date.

file Benyttes for å gjøre filer tilgjengelig for nedlasting. Inneholder referanse til en fil i persistenslaget, som velges gjennom et skjema i Presentation Content Viewer Attributten required er ikke støttet for file.

plugin Benyttes for å legge inn plugins, ref. Ingvaldsen, 2008 for bruk av plugins. Attributten required er ikke støttet for plugin.

entity Benyttes for å nøste en entity inn i den gjeldende entityen. Attributten ref-id er påkrevd for å definere hvilken entity som skal nøstes. Attributten required er ikke støttet for entity.

entity-list Benyttes for å nøste en liste over entiteter inn i den gjeldende entiteten. ref-id er påkrevd. required er ikke støttet. Det er mulighet for å nøste entityer i flere nivåer.

3.1.2 Entity-instances

Entity definerer kun strukturen på informasjonen, og må instansieres. Dette gjøres ved hjelp av entity-instance. Når en ny presentasjon blir opprettet basert på en presentasjonsmønsterspesifikasjon vil innholdsfilene for presentasjonen bli opprettet på bakgrunn av entity-instance-elementene. Disse innholdsfilene er xml-filer som har strukturen til entity-definisjonene, og inneholder informasjonen som skal lagres, samt en del metadata, og det er disse filene systemet henter informasjonen ut fra når nettsider skal genereres.

3.1.3 Views

Et view definerer hvordan en entity-instance skal rendres til HTML. Dette gjøres ved å gi en kobling til en xslt-transformasjon. Filtypen .xslt er utelatt når transformasjonen defineres. View inneholder også en kort beskrivelse av viewet, som vises til administratoren i Presentation Manager. Flere views kan knyttes opp mot samme entity-instance, slik at den samme informasjonen kan rendres på forskjellige måter der det er ønskelig.

3.1.4 Pages

Pages definerer nettsidene som en presentasjon skal inneholde. En page består av en page-template og ett eller flere views, samt en beskrivelse av siden. De forskjellige viewene blir opprettet innenfor et composition element, som inneholder viewene, og attributter som definerer hvordan hvert view oppfører seg. Det må deklareres ett view som er default. Attributten alwaysVisible="true" sørger for at et view alltid blir rendret på siden. Et view kan gjenbrukes i flere pages, for å presentere informasjonen på forskjellige måter.

3.2 Oppbygningen av en nettside

Kapittel 4

kodetest

Listing 4.1: Eksempel på kode rett fra tex-fila

```
1  public boolean equals(Punkt3D obj) { //overkjorer equals()↩
       metoden fra Punkt2D
2  if (obj instanceof Punkt3D) { Punkt3D p = obj; return ((p.↩
       getX()==this.getX())&&(p.getY()==this.getY())&&(p.z==z));
3    }
4    else{
5      return false;
6    }
7  }
```

Listing 4.2: Eksempel på kode lest rett fra fil

```
1  package problem53;
2
3  import java.math.BigInteger;
4
5  public class K53 {
6
7        /**
8         * Tester om det funker med
9         */
10   public static void main(String[] args) {
11
12     int count = 0;
13     for(int i = 1; i <= 100; i++ ){
14       for(int j = 1; j <= i; j++){
15         if(cnr(i,j)){
16           count++;
17         }
18       }
19     }
20     System.out.println(count);
21   }
```

```
22
23    public static boolean cnr(int n, int r){
24      boolean ret = false;
25      BigInteger max = BigInteger.valueOf(1000000);
26      BigInteger a = getFac(n);
27      BigInteger b = getFac(r);
28      BigInteger c = getFac(n-r);
29      BigInteger res = a.divide(b.multiply(c));
30
31      if(res.compareTo(max) >= 1){
32        ret = true;
33      }
34      return ret;
35    }
36
37    public static BigInteger getFac(int x){
38      BigInteger n = BigInteger.ONE;
39          for (int i=1; i<=x; i++) {
40              n = n.multiply(BigInteger.valueOf(i));
41
42          }
43          return n;
44    }
45  }
```

Nå skal jeg kunne referere til samplecode 4.2 med labelnavnet.

Bibliografi

[1] 08 2009.

[2] Jostein Bjørge Peder Skeidsvoll. Utvikling av presentasjonsmønstre. 2009.

[3] Bjørn Christian Sebak. Dynamic presentation generator 2.0 – utvikling av ny dynamisk presentasjonsgenerator og presentasjonsmønsterspesifikasjon. Master's thesis, university of Bergen, 2008.

[4] Nicola L. C. Talbot and Gavin C. Cawley. A fast index assignment algorithm for robust vector quantisation of image data. In *Proceedings of the I.E.E.E. International Conference on Image Processing*, Santa Barbara, California, USA, October 1997.