Una metodologia di progettazione di applicazioni web centrate sui dati

ALBERTO BELUSSI
ANNO ACCADEMICO 2010/2011

Progettazione **logica** di un sito web centrato sui dati

Si propone la seguente metodologia di progetto. L'obiettivo è quello di specificare in modo formale il contenuto informativo e la struttura dell'ipertesto che costituisce il sito. Inoltre, visto che l'informazione viene prelevata da una base di dati, viene specificato anche come (con quali interrogazioni SQL) il dato viene estratto dal DBMS per essere presentato nelle pagine web del sito.

Fasi della metodologia

La metodologia proposta è strutturata in 4 fasi:

1. Organizzazione in pagine web del contenuto informativo
2. Definizione della navigazione tra le pagine web (specifica dei LINK)
3. Corrispondenza tra il contenuto delle pagine e la basi di dati (interrogazioni SQL)
4. Presentazione (aspetto grafico delle pagine: fogli di stile CSS)

Progettazione logica - FASE 1

La struttura delle pagine web e il loro contenuto informativo possono essere specificati definendo SCHEMI di PAGINA.

Linguaggio per la specifica di schemi di pagina:

```
page schema <nomeSchema> [unique](
          <nomeAttributo>: <TIPO>;
          …
          <nomeAttributo>: <TIPO>;
    )
<TIPO> :== {string, integer, date, time,
            real, <LISTA>}
<LISTA> :== list_of (<nomeAttr>: <TIPO>; …
                     <nomeAttr>: <TIPO>;)
```

Progettazione logica – FASE 2

La specifica dei legami tra pagine si ottiene aggiungendo i **link** tra gli schemi di pagina. In particolare, un link è un tipo specifico che si aggiunge agli altri possibili tipi di attributo.

```
<TIPO> :== {string, integer, date, time, real,
           <LISTA>, <LINK>}
<LINK> :== {link(<ETICHETTA>; *<nomeSchema>)|
            link(<ETICHETTA>; url(<nomeURL>)|
            link(<ETICHETTA>;
                url(<nomeURL>,<nomeAttrib>))}
<ETICHETTA> :== {<nomeEtichettaFissa>|
     <nomeAttrib>: {string|integer|date|time}}
```

Progettazione logica – FASE 3

La terza fase prevede la specifica della **corrispondenza** tra la base di dati e il contenuto informativo degli schemi di pagina. Nella sintassi proposta tale corrispondenza si precisa per ogni schema di pagina *<nomeSchema>* come segue:

```
DB to page schema <nomeSchema>
parameter(<nomepar>, …,<nomepar>)
(
      <nomeAttr>, …,<nomeAttr>: <QUERY_SQL>;
      …
      <nomeAttr>, …,<nomeAttr>: <QUERY_SQL>;
)
```

dove i parametri dovranno comparire nelle interrogazioni SQL con la sintassi *?nomepar?* e gli attributi *<nomeAttr>* devono essere tutti e soli gli attributi dello schema di pagina *<nomeSchema>*.

Risultato della progettazione logica

- Schema logico della base di dati:
  - schema relazionale: tabelle + vincoli di integrità
- Elenco di schemi di pagina:
  - schema del sito: page schema + link
- Specifica della corrispondenza con la basi di dati:
  - Interrogazioni SQL sullo schema relazionale mappati sul contenuto informativo degli schemi di pagina.

Progettazione logica – FASE 2 e form

Può essere necessario introdurre una form in uno schema di pagina. Tale specifica si ottiene aggiungendo allo schema di pagina un attributo che descrive una form. A tale scopo si aggiunge un nuovo tipo di attributo.

```
<TIPO> :== {string, integer, date, time, real,
            <LISTA>, <LINK>, <FORM> }
<FORM> :== form(<nomeAttrib_1>: <TIPOINPUT>;
                …
               <nomeAttrib_n>: <TIPOINPUT>;)
<TIPOINPUT> :== { text | radio [(v_1, …, v_n)]
                | password | date | time
                | checkbox [(v_1, …, v_n)]
                | select [(v_1, …, v_n)]
                | submit(*<nomeSchema>)}
```

Ogni form deve contenere sempre un attributo **submit**.

Progettazione logica – FASE 3 e form

In presenza di form la terza fase prevede la possibilità di specificare delle operazioni da eseguire sulla base di dati con le informazioni raccolte da ciascuna form dello schema. Nella sintassi proposta tali operazioni si precisano per ogni schema di pagina *<nomeSchema>* e per ogni form *<nomeForm_i>* dello schema come segue:

```
DB from page schema <nomeSchema>
(
      <nomeForm_1>: <OPERAZIONE_1>;
      …
      <nomeForm_k>: <OPERAZIONE_k>;
)
```

dove gli attributi della form dovranno comparire nei comandi SQL che implementano le operazioni con la sintassi *?nomepar?*. Infine le form *<nomeForm_i>* devono essere tutte e sole le forms dello schema *<nomeSchema>*.

Progettazione logica – FASE 3 e form

Infine un'operazione può essere:

```
<OPERAZIONE> :== { <COMANDO_SQL> |
            if <INTERROGAZIONE_SQL> <CONFRONTO> <CONST>
            then *<nomeSchema> else *<nomeSchema> end }

<CONFRONTO> :== { = | < | > | <= | >= }
<CONST> :== "costante compatibile con il risultato
            dell'interrogazione SQL"
```

Si noti che uno schema di pagina SP ha certamente una specifica "DB to page-schema SP …", e può avere in aggiunta una specifica "DB from page-schema SP…" qualora contenga delle forms.

Infine se l'operazione associata alla form è del tipo "**if** …" allora l'attributo **submit** può omettere lo schema di pagina target.

Esempio di schema con form

```
page-schema InsertDip unique (
       datiNewDip: form(Codice: text;
              Nome: text;
              Cognome: text;
              DataNascita: date;
              Qualifica: select;
              Inserisci: submit(*Esito);); )
DB to page-schema InsertDip
parameter()
(      Qualifica: SELECT nome FROM Qualifica;  )
DB from page-schema InsertDip
(      datiNewDip: INSERT INTO Dipendente
              (cod, nome, cogn, datan, qual) VALUES
              (?Codice?,?Nome?,?Cognome?,?DataNascita?,
               ?Qualifica?);)
```

Esempio di schema con form

```
page-schema Esito unique (
 esitoInserimento: integer;
 nuovoInserimento: link("Nuovo dipendente",*InsertDip);
)
DB to page-schema Esito
parameter(Codice)
(      esitoInserimento: SELECT count(*)FROM Dipendente
                         WHERE cod = ?Codice?; )
```

Implementazione?

- Quale tecnologia usare?
- Quale corrispondenza tra la progettazione logica e l'implementazione?

Tecnologie disponibili: **Servlet, JSP**, PHP, ASP, **postgresql**, Mysql, ecc…

Architetture software: approccio Model-View-Controller, **MCV-2 servlet-centric**.

Architetture software per applicazioni web

Approccio banale:

- Una servlet (o JSP) per ogni schema di pagina che si occupa di tutti gli aspetti:
  - Elaborazione della richiesta HTTP.
  - Connessione alla base di dati ed estrazione delle informazioni.
  - Generazione del codice HTML da restituire al browser.

Approccio più evoluto:

- Architettura MVC-2 (servlet centric): distribuire le varie funzionalità a diversi "attori" (moduli software = classi JAVA).