

# Progetto di Tecnologie Web

Docenti:  
Prof.ssa Gabriella Trucco  
Prof. Paolo Ceravolo

---

## *El véc scàfal*

---

di Mattia Cavenaghi – matricola 736856



Anno Accademico  
2009/2010

El véc scàfal

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Il database MySQL PROGETTO_TW</b>	<b>5</b>
<b>3</b>	<b>Struttura di una pagina web</b>	<b>6</b>
3.1	Le molliche di pane . . . . .	6
3.2	File inclusi nelle pagine (intestazione, barra segnaposto, menù e piè di pagina) . . . . .	8
3.3	Il foglio di stile stile.css . . . . .	8
<b>4</b>	<b>Registrazione al sito</b>	<b>10</b>
<b>5</b>	<b>Accesso alle aree riservate del sito</b>	<b>12</b>
<b>6</b>	<b>Le funzionalità lato utente</b>	<b>13</b>
6.1	Modifica dei dati dell'utente . . . . .	13
6.2	Inserimento e modifica di una recensione . . . . .	13
6.2.1	Inserimento di una recensione . . . . .	13
6.2.2	Modifica di una recensione . . . . .	14
6.3	Inserimento e modifica di un annuncio . . . . .	15
6.3.1	Il file elencoAnnunci.xml . . . . .	15
6.3.2	Il file schemaAnnunci.xsd . . . . .	16
6.3.3	I file di trasformazione XSLT . . . . .	17
6.3.4	Inserimento di un annuncio . . . . .	18
6.3.5	Modifica di un annuncio . . . . .	19
<b>7</b>	<b>Le funzioni lato amministratore</b>	<b>20</b>
7.1	Modifica e cancellazione dei dati degli utenti . . . . .	20
7.1.1	Modifica dei dati di un utente . . . . .	20
7.1.2	Cancellazione con anteprima dei dati di un utente . . . . .	20
7.1.3	Cancellazione senza anteprima dei dati di un utente . . . . .	21
7.2	Modifica e cancellazione dei dati delle recensioni . . . . .	21
7.2.1	Cancellazione con anteprima di una recensione . . . . .	21
7.2.2	Cancellazione senza anteprima delle recensioni . . . . .	22
7.3	Modifica e cancellazione dei dati degli annunci . . . . .	22
7.3.1	Cancellazione con anteprima di un annuncio . . . . .	22
7.3.2	Cancellazione senza anteprima degli annunci . . . . .	22

<b>8 Conclusioni</b>	<b>24</b>
<b>A Figure</b>	<b>25</b>
<b>B Listati di codice</b>	<b>30</b>
B.1 Le molliche di pane . . . . .	30
B.2 Il menù di navigazione . . . . .	30
B.3 Registrazione, modifica ed eliminazione dei dati degli utenti . . .	31
B.4 Accesso al sito . . . . .	33
B.5 Inserimento, modifica e cancellazione di una recensione . . . . .	34
B.6 Il file elencoAnnunci.xml . . . . .	38
B.7 Il file listaAnnunci.xsl . . . . .	38
B.8 Il file dettaglioAnnuncio.xsl . . . . .	40
B.9 Inserimento, modifica ed eliminazione di un annuncio . . . . .	41
B.10 La lista degli utenti registrati . . . . .	44

## 1 Introduzione

L'obiettivo del progetto è quello di realizzare un sito internet che soddisfi le specifiche progettuali indicate dal docente e riassunte nei seguenti punti:

- le pagine devono essere sviluppate in formato XHTML;
- tutte le pagine XHTML devono essere validate;
- il layout delle pagine deve essere sviluppato con CSS;
- il progetto deve prevedere di rappresentare parte dei dati come documenti XML (ad esempio, prevedere la ricerca in un archivio o catalogo di prodotti, documenti, oggetti);
- tali documenti devono rappresentare istanze di strutture dati definite utilizzando XML Schema (obbligatorio) e DTD (facoltativo);
- tutti i documenti XML devono essere ben formati e validati;
- prevedere la visualizzazione di tali documenti attraverso trasformazione XSLT (da XML a XHTML);
- progettare funzionalità che richiedano l'impiego di espressioni XPath o l'uso di XQuery;
- il progetto deve implementare JSP (Java Server Pages) per la creazione dinamica di contenuti;
- le JSP devono essere utilizzate per accedere a dati residenti in un database o descritti da documenti XML;
- se non viene utilizzato un database, occorre conoscere comunque le modalità di accesso via JDBC;
- utilizzare Tomcat come JSP container e come Web Server;
- l'applicazione deve usare in maniera integrata tutte le tecnologie discusse, quali XHTML, CSS, XML, XML Schema XSLT, XPath, XQuery e JSP.

Per descrivere il funzionamento e l'implementazione del prodotto realizzato, si effettueranno delle simulazioni di utilizzo del sito da parte di un ipotetico utilizzatore, che identificheremo coi termini di *utente*, nel caso di un attore che

intenda visitare ed usufruire dei servizi implementati, e coi termini *amministratore* nel caso di un attore che detiene i diritti di cancellazione e modifica di taluni dati, descritti nel seguito di questa relazione.

Il sito denominato *El véc scaffàl*, che in dialetto lodigiano significa Il vecchio scaffale, si pone come ipotetico obiettivo di essere il punto di incontro tra gli appassionati di lettura. Nel prodotto realizzato si sono create diverse sezioni che consentono di svolgere principalmente le operazioni elencate nei seguenti punti:

- sezione utenti: dove gli utilizzatori possono registrarsi al sito e l'amministratore può gestire i dati degli utenti;
- sezione accesso: attraverso cui gli utilizzatori possono accedere al sito per poter usufruire dei servizi dedicati;
- sezione annunci: dove gli utenti possono consultare ed inserire annunci economici;
- sezione recensioni: dove gli utenti consultano ed inseriscono le recensioni delle opere lette.

## 2 Il database MySQL PROGETTO\_TW

Nel progetto sviluppato si è scelto di utilizzare un database chiamato PROGETTO\_TW, contenente due tabelle: UTENTI e RECENSIONI.

La tabella degli utenti contiene i dati di tutti gli utenti iscritti al sito, tali dati consistono nei dati anagrafici, nel nickname e password scelti dall'utente e nell'indirizzo di posta elettronica tramite cui l'amministratore può contattarlo. Un ulteriore campo contiene tutti i generi letterari di interesse da parte dell'utente, ognuno separato dal carattere “,” che vedremo sarà di utilità nel processo di visualizzazione ed elaborazione di tale dato.

La tabella delle recensioni contiene invece i dati relativi a tutte le recensioni inserite dagli utenti registrati; tali dati contemplano numero di riferimento della recensione, il titolo dell'opera, la data di pubblicazione dell'opera, l'autore dell'opera, l'editore, il testo della recensione, la data di inserimento o modifica della recensione nel database, l'autore della recensione, ed un voto da uno a cinque punti che associa l'indice di gradimento di una certa opera da parte dell'autore della stessa. Va osservato che in questa tabella viene inserito a discrezione dell'utente, il nome del file raffigurante l'opera recensita, memorizzato nell'apposita cartella RECENSIONI, sottocartella di IMMAGINI, inclusa a sua volta nella cartella WEB del progetto.

## 3 Struttura di una pagina web

Le pagine web implementate, ad esclusione dell'elenco degli annunci economici realizzato in XML (per ulteriori chiarimenti si rimanda al paragrafo 6.3.1), sono in formato JSP e presentano tutte la medesima struttura costituita da:

- dichiarazione del DTD il quale definisce la tipologia di pagina visualizzata (XHTML 1.0 transitional);
- dichiarazione delle librerie Java (o JSTL) utilizzate nella pagina;
- istanziamento delle molliche di pane (paragrafo 3.1);
- collegamento al foglio di stile CSS che definisce la formattazione dei contenuti della pagina (paragrafo 3.3);
- corpo della pagina: contenente l'head ed il body della struttura HTML.

### 3.1 Le molliche di pane

Con il termine molliche di pane (in inglese *breadcrumbs*) si designa un particolare indice che consente all'utilizzatore di un sito di identificare velocemente la sua posizione all'interno della struttura del sito stesso.

Supponiamo ad esempio che l'utente Mario Rossi, il cui nickname è mrossi, stia visitando la pagina relativa ad un annuncio economico. Come possiamo osservare nella figura 1 posta a pagina 25, nel riquadro blu sono evidenziate le molliche di pane che partendo dalla stringa a sinistra, sono così strutturate:

- etichetta e collegamento alla home page;
- carattere di separazione: nel progetto si è designato a questo scopo il carattere “»”;
- etichetta e collegamento alla pagina *indice di sezione*: con tale termine ed a seconda della tipologia di utilizzatore, si designano pagine di natura diversa. Nel caso di un utente, la pagina sarà quella di inserimento di una nuova occorrenza del tipo di oggetto che si vuole elaborare (utente, annuncio economico o recensione), mentre nel caso dell'amministratore sarà la pagina contenente l'elenco di tutte le occorrenze di una determinata categoria di oggetti;
- carattere di separazione;

- etichetta e collegamento alla *pagina corrente*: se la pagina corrente è adibita all’inserimento od alla visualizzazione di un’occorrenza, il collegamento conterrà la URL riferita alla pagina stessa, comprensiva della eventuale query. Se la pagina è risultato di una particolare azione come la modifica, cancellazione, l’accesso o la registrazione di un nuovo utente, il collegamento sarà costituito dal riferimento alla pagina di inserimento di una nuova occorrenza: si è scelto ciò per evitare che l’utente, cliccando sul collegamento in oggetto, possa eseguire due volte la stessa azione, causando così la generazione di un possibile errore, come potrebbe accadere nel caso della duplice cancellazione di una medesima occorrenza.

Tenendo presente il nostro esempio di figura 1 possiamo ora analizzare la schematizzazione del codice contenuto nel listato 1 riportato nell’apposito appendice a pagina 30. La riga 3 ci consente di identificare la matrice di stringhe che conterrà i dati delle nostre molliche, essa contiene nella colonna di indice 0 tutte le etichette che andranno stampate a video, mentre nella colonna di indice 1, saranno contenuti i collegamenti alle pagine a cui tali etichette sono riferite.

Il collegamento alla home page è sempre lo stesso, quindi è stato inserito a mano al momento dell’implementazione (riga 9), per il collegamento all’indice di sezione si è reso necessario l’utilizzo di un costrutto CHOOSE che restituisce il collegamento alla lista delle recensioni, nel caso in cui l’utilizzatore sia l’amministratore, mentre restituisce il collegamento alla pagina di inserimento dei dati di una nuova occorrenza, nel caso in cui l’utilizzatore sia un semplice utente (righe 17-23).

L’ultimo collegamento è realizzato mediante una condizione IF, la quale mantiene anche le variabili appartenenti alla query posta nella URL e per questioni di validazione della pagina XHTML, sostituisce il carattere “&” con la stringa “&amp;”, codifica del medesimo carattere di separazione delle variabili (righe 11-14).

Come ultima osservazione si vuol far notare che non tutte le pagine hanno tre indici: la home page ne contiene solamente uno, mentre la pagina di conferma del logout ne contiene quattro, che analogamente alle pagine risultato delle operazioni di inserimento, modifica e cancellazione di un’occorrenza, non rimanda a nessuna pagina, azione eseguibile mediante il link avente come riferimento il carattere “#”.



### 3.2 File inclusi nelle pagine (intestazione, barra segnaposto, menù e piè di pagina)

Onde evitare la riscrittura in ogni pagina, del codice relativo all'intestazione, alla barra segnaposto, al menù ed al piè di pagina si sono creati due file: `intestazione.jsp` che codifica l'intestazione, la barra segnaposto ed il menù e `piePagina.jsp` che codifica ovviamente il piè di pagina.

L'intestazione ed il piè di pagina sono costituite da due sezioni XHTML (dei DIV) contenenti convenevolmente del testo o delle immagini.

La barra segnaposto è anch'essa un DIV, e consente la visualizzazione delle molliche di pane analizzate nel precedente paragrafo 3.1 e del nickname dell'utilizzatore del sito: la visualizzazione di tali dati consiste nella stampa a video della variabile di sessione `MOLLICHE` istanziata all'apertura di ogni pagina, e della variabile di sessione `NICKNAME` istanziata nella fase di login, operazione che analizzeremo nella sezione 5 di questa relazione.

L'ultima sezione contenente il menù è anch'essa dinamica: a seconda del tipo di utilizzatore ed a seconda che quest'ultimo abbia effettuato l'accesso con esito positivo al sito, esso riserva certi collegamenti ad altrettante pagine. Questa operazione è resa possibile mediante dei costrutti CHOOSE schematizzati nel listato 2 a pagina 30. L'utilizzatore `AMMINISTRATORE` in sostanza potrà accedere a pagine che all'utilizzatore `UTENTE` saranno precluse, come ad esempio le pagine di gestione degli utenti. L'ospite infine avrà un accesso alla pagina di registrazione, la quale sarà invece inutile per gli utenti già registrati e quindi il relativo collegamento non verrà visualizzato nel menù di questi ultimi.

### 3.3 Il foglio di stile `stile.css`

Il file in questione serve alla formattazione dei contenuti delle pagine, la quale avviene attraverso di regole applicate ai tag XHTML (i selettori).

Il foglio può essere suddiviso in tre sezioni:

- regole della pagina: contenente le regole per la formattazione dei paragrafi, dei titoli e delle sezioni DIV principali, quali `PAGINA`, `BANNER`, `MENU` e `TESTO`;
- regole per il menù di navigazione: queste regole sono state riscritte utilizzando del codice già sviluppato, reperito alla pagina <http://ago.tanfa.co.uk/css/examples/css-dropdown-menus.html>;

- regole aggiuntive: sono regoli aggiuntive alle sezioni, o regole dedicate a sezioni utilizzate per piccole operazioni di formattazione dei dati.

## 4 Registrazione al sito

La registrazione al sito è consentita solamente agli utilizzatori non loggati al sito, difatti nel caso in cui un utente o l'amministratore che sono loggati, vogliano accedere a questa pagina, non potranno poichè il relativo collegamento nel menù viene oscurato (per ulteriori chiarimenti si rimanda al paragrafo 3.2).

Ai fini della nostra analisi e per comprendere l'implementazione della funzione di registrazione possiamo supporre di impersonare l'utilizzatore fittizio Tizio Caio che si vuole iscrivere tramite l'apposita pagina registrazioneUtente.jsp accessibile tramite la voce *Registrazione* del menù. Cliccando sul relativo link, Caio visualizzerà e successivamente compilerà la form riportata in figura 2 posta a pagina 26.

Dopo aver compilato i campi ed aver premuto il pulsante *Registra!*, i dati verranno inviati tramite il metodo POST alla pagina di elaborazione confermaRegistrazione.jsp. Prima di procedere oltre va osservato, nella figura 2, il valore della stringa posta nella barra degli indirizzi del browser: essa oltre a contenere la URL della pagina, contiene una query identificata dal carattere di separazione "?". Tale query specifica la variabile COMANDO da sottoporre sia alla pagina registrazioneUtente.jsp che alla pagina confermaRegistrazione.jsp, le quali attiveranno gli opportuni snippet di codice tramite la parola chiave INSERISCI, questo metodo è stato largamente utilizzato in questo progetto per realizzare tutte quelle funzionalità che necessitano l'attivazione di porzioni di codice all'interno delle pagine identificate dal suffisso *conferma*.

Il comando INSERISCI sottoposto alla pagina confermaRegistrazione.jsp, non fa nient'altro che attivare la condizione IF contenente la query SQL da eseguire, e mediante cui i dati dell'utente vengono salvati nell'apposita tabella del database su cui l'intero progetto poggia (per ulteriori approfondimenti consultare la sezione 2).

Nel listato 3 posto a pagina 31 è possibile osservare il codice che realizza le azioni fin qui descritte: dopo il codice necessario per l'accesso al database PROGETTO\_TW (righe 2-6), i generi letterari selezionati dall'utente vengono concatenati nella medesima stringa e separati dal carattere ";" (righe 20-25), si è scelta questa implementazione poichè istanziare nella tabella SQL UTENTI, un certo numero di campi adibito alla memorizzazione dei generi, sarebbe potuto risultare o dispersivo od insufficiente, a seconda che l'utilizzatore scelga pochi o tanti generi. L'implementazione consente quindi di elaborare un singolo campo contenente tutti i generi, il quale verrà in seguito, manipolato mediante dei

costrutti FOR.

## 5 Accesso alle aree riservate del sito

Il nostro nuovo utente Tizio Caio può ora accedere al sito in modo tale da poter usufruire dei servizi dedicati alla sua nuova categoria di appartenenza e che verranno analizzati nella prossima sezione 6.

Tizio Caio deve accedere alla pagina `accesso.jsp`, tramite la voce *Accesso* del menù, ed inserire nell'apposita form i dati forniti in fase di registrazione, nella apposita form (figura 3 a pagina 26). Cliccando sul pulsante *Accedi* ed analogamente a quanto visto per la procedura di registrazione, i dati inseriti verranno inviati alla pagina di elaborazione denominata `confermaAccesso.jsp`. Tale pagina risulta essere però più semplice rispetto alla sua controparte realizzata per la registrazione, difatti contempla solo due operazioni, identificate da due differenti comandi ossia l'accesso e l'uscita dal sito. Si osservi il listato 4 a pagina 33: dopo aver effettuato la connessione al database (righe 4-8), l'accesso consiste nella verifica da parte del server mediante una query select, che i dati passati dall'utente siano presenti nella tabella `UTENTI` (righe 10-14 e riga 20), se l'esito è positivo il nickname viene memorizzato in sessione, in caso contrario l'utente viene reindirizzato alla form di accesso (righe 20-22).

L'operazione di uscita dall'area riservata consiste invece nel cliccare sulla voce *Esci* posta a destra della barra segnaposto (figura 1 nel riquadro verde), tale azione consente l'apertura della pagina `confermaAccesso.jsp` a cui è concetenata la query `COMANDO=ESCI`, che attiva la condizione necessaria a cancellare dalla sessione l'username dell'utente (righe 25-27). Tale azione è possibile in qualsiasi pagina appartenente al sito ma si ricorda, solamente se l'utente è loggato allo stesso.

## 6 Le funzionalità lato utente

In questa sezione si analizzeranno tutte le funzionalità dedicate all'utente registrato e loggato al sito, ad esclusione delle semplici operazioni di lettura degli oggetti, che ovviamente consistono nell'apertura di singole pagine web.

### 6.1 Modifica dei dati dell'utente

Tizio una volta registrato, può accedere nuovamente alla pagina di registrazione in modo tale da modificare i propri dati, ad esclusione del nickname e della password, per tali campi è difatti necessaria l'azione dell'amministratore (per ulteriori chiarimenti consultare il paragrafo 7.1).

L'utente accedendo alla pagina `registrazioneUtente.jsp` tramite la voce *Modifica i tuoi dati* della categoria *Registrazione* posta nel menù di navigazione, visualizzerà la scheda riassuntiva contenente i dati inseriti in precedenza (figura 4 a pagina 27). Cliccando quindi sul pulsante *Modifica*, l'utente accede alla form di immissione dei dati analoga a quanto visto per l'operazione di registrazione. Una volta inseriti i dati aggiornati e cliccando sul pulsante *Modifica*, i dati inseriti verranno inviati alla pagina `confermaRegistrazioneUtente.jsp` che grazie alla parola chiave `MODIFICA`, associata alla variabile `COMANDO`, attiverà la condizione contenente le istruzioni utili ad inserire nella tabella `UTENTI` i nuovi dati, sfruttando le librerie JSTL che dopo aver effettuato l'accesso al database esegue una query SQL di tipo `update` (righe 58-67 del listato 3 a pagina 31) sull'occorrenza che presenta come chiave di selezione il nickname dell'utente loggato, che nel nostro esempio risulta essere `TCAIO`.

### 6.2 Inserimento e modifica di una recensione

#### 6.2.1 Inserimento di una recensione

L'operazione di inserimento di una recensione, così come per l'operazione di registrazione, fa uso di due pagine ossia `recensione.jsp` che contiene la form di inserimento dei dati (figura 5 a pagina 27) e `confermaRecensione.jsp` che riceve i dati dalla precedente pagina tramite la modalità `POST`, e li memorizza nella relativa tabella `RECENSIONI` del database (listato 5 a pagina 34).

Una volta che l'utente Caio ha effettuato l'accesso al sito, può aprire la pagina `recensione.jsp` tramite la voce *Inserisci una recensione* della categoria *Recensioni* posta nel menù di navigazione, e compilare la form in essa contenu-

ta. La pressione del pulsante *Inserisci* avvia la procedura di memorizzazione effettuata mediante la pagina `confermaRecensione.jsp`, che prevede l'esecuzione della condizione associata alla parola chiave `INSERISCI`, associata alla variabile `COMANDO` (riga 90). Questa condizione esegue una query SQL di tipo `insert`, con l'aggiunta di un'operazione di memorizzazione dell'eventuale file associato all'immagine dell'opera recensita (righe 68-85). Nel caso in cui l'utente scelga di non inserire un'immagine, nel database viene memorizzata la parola chiave `NESSUNO` (riga 78), utile nelle fasi di modifica ed eliminazione della recensione.

Va osservato infine che essendo contemplata l'operazione di memorizzazione di un file sul server, la form dovrà gestire l'invio di tale oggetto mediante il metodo di crittazione `multipart/form-data` che consente l'invio di allegati. Questa modalità di crittazione necessita dell'uso di uno snippet Java, il quale effettua il parsing dei dati inseriti nel corpo della pagina inviata a `confermaRecensione.jsp` (righe 58-67).

### 6.2.2 Modifica di una recensione

La modifica di una recensione necessita da parte dell'utente, la selezione della stessa dall'elenco delle recensioni, visualizzabile tramite la pagina `listaRecensioni.jsp` (figura 6 a pagina 28). Questa pagina visualizza il riferimento, l'autore, il titolo, l'anno di pubblicazione e l'autore della recensione, di tutti gli oggetti presenti nella apposita tabella e mediante una query `select`: il risultato di questa operazione viene stampato a video con l'aggiunta di un collegamento *Dettagli* tramite cui accedere alla scheda riassuntiva dei dati dell'occorrenza voluta.

Va osservato che la lettura e la modifica di un'occorrenza sono due operazioni differenti: la lista delle recensioni disponibili in lettura, contiene tutti gli oggetti presenti nella tabella `RECENSIONI` selezionati utilizzando come query `COMANDO=LEGGI`, mentre la lista delle recensioni disponibili per la modifica ritorna tutte le sole recensioni dell'utente loggato ed utilizza la query `COMANDO=MODIFICA`. Ovviamente se l'utente non ha inserito recensioni la query non ritornerà nessuna occorrenza e la lista degli oggetti modificabili conterrà il messaggio *Nessuna recensione disponibile*.

La pagina tramite cui visualizzare la form utile alla modifica dei dati di una recensione è `recensione.jsp` che grazie al comando `MODIFICA`, attiva la relativa condizione con cui vengono selezionati i dati dell'oggetto voluto che vengono a loro volta inseriti come valori di default nei campi della form stessa (figura 7 a pagina 28). Caio all'avvenuta variazione dei dati e premendo il pulsante

*Modifica*, invierà gli stessi tramite la modalità POST e la già citata crittazione multipart/form-data alla pagina confermaRegistrazione.jsp che grazie alla query SQL update, eseguita utilizzando le istruzioni riportate nel listato 5 a pagina 34 alle righe 99-159, li registra nella tabella RECENSIONI.

Nel caso in cui l'utente volesse modificare l'immagine relativa alla recensione, deve semplicemente selezionare un nuovo file tramite l'apposito campo *Immagine dell'opera recensita* (figura 7 nel riquadro blu): il codice riportato nel listato 5 a pagina 34 si occuperà di cancellare il vecchio file, caricare sul server la nuova immagine e memorizzare nel database il nome del nuovo file (righe 114-138). La cancellazione invece avviene tramite la deselezionazione della checkbox posta accanto al nome del file presente nel database, operazione che implica, una volta inviati i nuovi dati tramite il pulsante *Modifica*, la cancellazione dal server del vecchio file (righe 100-113) e la memorizzazione del valore NESSUNO, nel caso non si specifichi una nuova immagine, o dell'identificativo nuovo eventuale file.

### 6.3 Inserimento e modifica di un annuncio

Per implementare la sezione relativa agli annunci e per soddisfare i requisiti progettuali, si è scelto di adottare il linguaggio XML tramite cui creare il file database contenente gli annunci economici. Ciò implica anche l'utilizzo di un file XSD che ne definisce la struttura e di alcuni file di trasformazione XSLT, utili nella lettura e nella stampa a video dei dati contenuti nel file XML.

#### 6.3.1 Il file elencoAnnunci.xml

Osservando il listato 6 a pagina 38 è possibile osservare la struttura del file in questione: esso contiene un nodo radice chiamato ANNUNCI, che contiene tanti nodi figli quanti sono gli annunci inseriti dagli utenti ed identificati dai nodi con etichetta ANNUNCIO.

Ogni annuncio è caratterizzato da un ID, un AUTORE, una DATA di inserimento ed una TIPOLOGIA, e può contenere a sua volta i nodi figli TITOLO, TESTO, IMMAGINE, PREZZO e CONTATTO. PREZZO è caratterizzato dall'attributo VALUTA mentre CONTATTO ha un attributo TIPOLOGIA. Le caratteristiche di tali nodi e dei relativi attributi verranno analizzate nel seguente paragrafo 6.3.2 dedicato allo schema XSD associato al file XML.



### 6.3.2 Il file `schemaAnnunci.xsd`

Il file di cui parleremo in questo paragrafo serve per definire la struttura del file `elencoAnnunci.xml`; l'elenco è costituito da:

- nodo `ANNUNCI`: il nodo radice del documento XML, contiene tutti gli annunci inseriti dagli utenti può quindi essere vuoto;
- nodo `ANNUNCIO`: il nodo figlio del nodo radice, a differenza di quest'ultimo esso può comparire all'interno del documento con qualsiasi cardinalità e contiene i dati relativi ad un annuncio, caratterizzato dagli attributi:
  - `ID`: definisce l'indice numerico identificativo del singolo annuncio, nello schema tale indice è stato definito come un intero positivo, il cui inserimento è obbligatorio;
  - `AUTORE`: definisce l'autore dell'annuncio, tale attributo è definito come una stringa il cui inserimento è obbligatorio;
  - `DATA`: anch'esso è un attributo obbligatorio, codificato nello schema attraverso il tipo `data` il cui formato è di tipo `aaaa-mm-gg`;
  - `TIPOLOGIA`: quest'ultimo attributo obbligatorio è rappresentato da una stringa i cui valori ammissibili sono `cerco`, che designa annunci di ricerca, e `vendo` che designa annunci di vendita.

Il nodo `ANNUNCIO` contiene al suo interno i nodi figli:

- `TITOLO`: di tipo stringa, definisce il titolo dell'annuncio;
- `TESTO`: di tipo stringa, definisce il testo dell'annuncio;
- `IMMAGINE`: anch'esso di tipo stringa, definisce il nome del file associato all'immagine dell'oggetto relativo all'annuncio;
- `PREZZO`: rappresenta il prezzo dell'oggetto in vendita o l'importo massimo che l'utente intende spendere per l'oggetto che sta cercando. Il prezzo viene definito come un numero a virgola mobile, caratterizzato dal seguente attributo:
  - `VALUTA`: esprime la valuta associata al valore numerico espresso come prezzo, è una stringa il cui inserimento non è obbligatorio, poichè viene definito come valore di default `EURO`.

- **CONTATTO:** esprime il recapito presso cui l'utente può essere rintracciato, è quindi una stringa che può contenere al suo interno caratteri o numeri. Questo nodo ha un attributo:
  - **TIPOLOGIA:** è obbligatorio ed espresso mediante una stringa. La tipologia può essere di tre tipi: TELEFONO, MAIL e o POSTA.

### 6.3.3 I file di trasformazione XSLT

Per poter estrarre ed elaborare o visualizzare i dati contenuti nel file XML analizzato nel precedente paragrafo 6.3.1, si può procedere utilizzando le tecnologie Java o XSLT. Per poter applicare quanto appreso durante il corso e soddisfare i requisiti progettuali, si sono impiegate a seconda delle necessità entrambe le tecnologie. In questo paragrafo vedremo l'impiego delle trasformazioni XSLT, utilizzato per eseguire operazioni semplici come la ricerca e la stampa a video dei dati di determinati annunci.

**Il file listaAnnunci.xsl** Il file che per questioni di lunghezza del codice è stato schematizzato nel listato 7 a pagina 38, consente l'estrazione dei dati degli annunci in modo tale da creare un elenco breve ma esaustivo, contenente cioè i dati più significativi degli annunci inseriti dagli utenti, popolando così la lista tramite cui Tizio Caio può leggere o modificare i propri annunci. Questo file estrae a seconda della tipologia di utilizzatore e dell'azione che si vuole intraprendere sui dati, il numero di riferimento, la data di pubblicazione, il nickname del venditore ed il titolo dell'annuncio, nonché un link alla pagina con cui visualizzare tutti i dati di un certo oggetto (righe 75-63 del listato 7 a pagina 38).

L'utente Tizio come accennato nella sezione dedicata alla recensioni (sezione 6.2), può scegliere di leggere o modificare annunci già presenti nel database, per fare questo esiste una variabile `COMANDO` che contiene la stringa, `LEGGI` o `MODIFICA`. La pagina `listaAnnunci.jsp` che riceve questa variabile la codifica come parametro XML attraverso le librerie JSTL e le da in input al file `listaAnnunci.xsl`. In questo file si selezionano i nodi del file XML attraverso l'espressione XPath `/ANNUNCI/`, seguita dal valore `ANNUNCIO` nel caso si vogliano leggere tutti gli annunci (righe 18-22), o seguita del valore `ANNUNCIO[@AUTORE=TCAIO]` nel caso l'utente voglia modificare le proprie occorrenze (righe 23-34). Una volta popolata la lista e premuto il link dei *Dettagli*, tale comando si propagherà alla pagina `dettaglioAnnuncio.jsp` dove si potranno leggere i dati dell'annuncio od

accedere alla pagina di modifica degli stessi: i dati del singolo annuncio vengono estratti tramite il file `dettaglioAnnunci.xml` analizzato nel seguente paragrafo.

L'operazione di lettura degli oggetti coinvolge tutte le occorrenze riportate nel file XML, per questo la selezione dei nodi avverrà attraverso la condizione `otherwise` riportata nella riga 18 del listato 7. L'operazione di modifica coinvolgerà invece tutti i soli annunci appartenenti all'utente, per tale motivo l'espressione XPath corrisponderà a quanto riportato alla riga 23 del listato 7.

Concludiamo l'analisi di questo file accennando al fatto che la data di inserimento dell'annuncio viene inserita dall'utente mediante il formato GG-MM-AAAA: XML però elabora solo formati AAAA-MM-GG, si è reso necessario perciò creare un template di conversione riportato nelle righe 44-50.

**Il file `dettaglioAnnuncio.xml`** Questo secondo file XSLT il cui codice è schematizzato nel listato 8 a pagina 40, permette semplicemente la stampa di tutti i dati di un singolo annuncio, selezionato nella pagina `listaAnnunci.jsp`, il cui meccanismo di funzionamento codificato nel file `listaAnnunci.xml` è stato illustrato nel precedente paragrafo.

Tenendo presente il listato 8 a pagina 40 possiamo come prima cosa osservare l'istanziamento delle variabili necessarie all'esecuzione delle espressioni XPath (righe 2-4). Il file prosegue con le istruzioni per la stampa dei dati (righe 6-18) e della formattazione della data del tutto simile a quanto visto nel precedente paragrafo. Per la stampa dei valori relativi al contatto, al prezzo ed alla relativa valuta si sono resi necessari altrettanti template (righe 20-22 e 24-33).

#### 6.3.4 Inserimento di un annuncio

L'utente Tizio Caio oltre a consultare gli annunci già inseriti nel relativo file, può scegliere, una volta loggato al sito, di inserire una nuova occorrenza di tale oggetto: per far ciò deve accedere alla pagina `annuncio.jsp` cliccando sulla voce *Inserisci un annuncio* contenuto nella categoria *Annunci* del menù di navigazione. Quest'azione apre la form di immissione dei dati, che per questione di spazio si è scelto di non riportare nell'apposita sezione, poichè decisamente simile a quanto realizzato per la sezione *Recensioni* vista in precedenza.

Caio premendo il pulsante *Inserisci* posto in fondo alla pagina `annuncio.jsp`, spedisce tali dati alla pagina `confermaAnnuncio.jsp`: la struttura di tale pagina è anch'essa simile a quanto osservato nella sezione *Recensioni*, per cui ci si soffermerà solamente sulle istruzioni di elaborazione del file XML, rimandando per

gli approfondimenti del rimanente codice, alla sezione 6.2. Osservando quindi il listato 9 a pagina 41, alle righe 10 ed 11 notiamo l'apertura del file XML attraverso l'opportuno oggetto di manipolazione di stream. Concentriamoci ora sul codice contenuto tra le righe 50-79, attivato dalla query `COMANDO=INSERISCI`: l'operazione inizia ricercando l'ultimo nodo presente nel file XML a questo difatti andrà anteposto il nuovo nodo `ANNUNCIO` i cui figli ed attributi sono i dati inseriti dall'utente ed estratti dal corpo della pagina `annuncio.jsp`.

L'operazione di inserimento dei dati si conclude con l'indentazione del codice XML (righe 116 e 117) e con la scrittura degli stessi nel file `elencoAnnunci.xml` (righe 119-128).

### 6.3.5 Modifica di un annuncio

La modifica di un annuncio prevede la selezione dello stesso da parte del nostro utente Tizio Caio, mediante la pagina `listaAnnunci.jsp` la cui logica di funzionamento è realizzata dal file `listaAnnunci.xsl` analizzato precedentemente. Sempre prendendo ad esempio quanto visto per la sezione delle recensioni, la selezione dell'oggetto voluto aprirà la scheda dell'annuncio tramite cui, cliccando sul pulsante *Modifica* si accederà alla form utile all'operazione di modifica dei dati. Una volta effettuate le variazioni e cliccando sul pulsante *Conferma*, i nuovi dati verranno inviati alla pagina `confermaAnnunci.jsp`: i dati inseriti andranno a costituire un nuovo nodo posto nella stessa posizione del vecchio annuncio. Per fare ciò viene ricercato l'annuncio da modificare di posizione  $i$  che viene cancellato (righe 85-103 del listato 9 a pagina 41), successivamente viene inserito un nuovo elemento in posizione successiva al nodo  $i - 1$  (riga 109).

Anche in questo caso per effettuare le operazioni di apertura, scrittura e chiusura del file viene istanziato un apposito oggetto (righe 113-128).

## 7 Le funzioni lato amministratore

In questa sezione si andranno ad analizzare le sole funzioni dedicate al lato amministrativo, come la modifica e l'eliminazione di occorrenze degli oggetti fin qui presentati. Per gli altri aspetti quali la consultazione delle liste e le schede dei dettagli, si rimanda alla precedente sezione 13.

### 7.1 Modifica e cancellazione dei dati degli utenti

L'amministratore una volta loggato al sito potrà osservare nel menù di navigazione, la voce *Utenti* contenente il collegamento alla lista degli utenti registrati al sito chiamato *Gestione dei dati degli utenti*. Le azioni eseguibili dall'amministratore sono di tre tipologie che analizzeremo separatamente.

#### 7.1.1 Modifica dei dati di un utente

La pagina `listaUtentiRegistrati.jsp` è costituita da una serie di occorrenze tante quante sono gli utenti registrati sul sito, ad esclusione dell'amministratore stesso; vicino al link *Contatto* vi è il link *Dettagli* che, cliccandovi sopra, dà accesso alla scheda relativa all'utente selezionato. Nella nuova scheda realizzata grazie alla pagina `dettaglioRegistrazioneUtente.jsp` è possibile osservare in fondo alla pagina un pulsante *Modifica* che permette l'accesso alla form contenente i campi utili all'operazione che si sta analizzando.

In quest pagina possiamo osservare due nuovi campi rispetto all'interfaccia realizzata per l'utente, e sono `nickname` e `password` che consentono all'amministratore di variare i dati in questione, operazione che per motivi di sicurezza non viene consentita all'utente. La pressione del pulsante *Conferma* invia infine i nuovi dati alla pagina `confermaRegistrazioneUtente.jsp` unitamente alla query `COMANDO=MODIFICA`, la quale attiva la condizione necessaria all'esecuzione della query SQL di `update` dei dati (righe 47-56 del listato 3 a pagina 31).

#### 7.1.2 Cancellazione con anteprima dei dati di un utente

Per eseguire questo tipo di operazione l'amministratore deve accedere alla pagina `dettaglioRegistrazioneUtente.jsp` associata all'occorrenza utente, scelta nella lista degli utenti registrati (vedere il precedente paragrafo 7.1.1). Una volta aperta la scheda dell'utente voluto, e cliccando sul pulsante *Elimina*, verrà inviata la query `comando=elimina` unitamente alla query `UTENTE=NOMEUTENTE` alla pagina `confermaRegistrazioneUtente.jsp` attivando la relativa condizione `IF`

(righe 79-64 del listato 3 a pagina 31) che esegue una query SQL di tipo delete sull'occorrenza utente, identificata dal nickname passato tramite query.

### 7.1.3 Cancellazione senza anteprima dei dati di un utente

Questa modalità di eliminazione dei dati, a differenza della precedente, è direttamente accessibile dalla lista degli utenti registrati: selezionando le checkbox poste accanto al link della scheda utente (figura 8 a pagina 29, nel riquadro blu) di ogni occorrenza e cliccando sul pulsante *Elimina*, la pagina listaUtentiRegistrati.jsp invierà a se stessa la query `COMANDOLista=ELIMINA` in aggiunta al vettore `NICKNAME` contenente i nickname degli utenti da rimuovere dal database. Queste query attiveranno un ciclo for (listato 10 a pagina 44) che eseguirà un numero di volte pari alle occorrenze selezionate, la query SQL delete.

## 7.2 Modifica e cancellazione dei dati delle recensioni

La modifica di una recensione è analoga a quanto visto tra le operazioni a disposizione dell'utente, si rimanda quindi al paragrafo 6.2.2 per un'analisi di tale procedura, va aggiunto solamente che l'amministratore detiene il diritto di modificare la data di pubblicazione della recensione attraverso l'apposito campo.

Nei successivi paragrafi ci si concentrerà invece nell'operazione di eliminazione degli oggetti recensione, eseguibile dal solo amministratore e secondo due modalità distinte.

### 7.2.1 Cancellazione con anteprima di una recensione

La cancellazione con anteprima di una recensione necessita della selezione preliminare della stessa nella pagina listaRecensioni.jsp, dove l'amministratore può aprire la scheda dei dettagli di un'occorrenza, dotata dei due pulsanti *Modifica* ed *Elimina*. Cliccando su quest'ultimo pulsante l'amministratore eseguirà la pagina confermaRecensione.jsp con l'aggiunta della parola chiave `ELIMINA`, la quale andrà ad attivare la condizione IF associata (righe ???). Questa condizione ricercherà il nome del file nella tabella `RECENSIONI` (righe 19-22 del listato 5 a pagina 34), eliminerà il file immagine (righe 36-38) e successivamente eliminerà l'occorrenza della recensione nella tabella concludendo così l'operazione (righe 28-31).

### 7.2.2 Cancellazione senza anteprima delle recensioni

Questa seconda modalità di eliminazione delle recensioni è eseguibile direttamente dalla pagina `listaRecensioni.jsp`, ed è simile a quanto visto nel paragrafo 7.1.3. In tale pagina accanto al link per l'accesso alla scheda dei dettagli, è posta una checkbox che una volta attivata e premuto l'apposito pulsante *Elimina*, elimina l'occorrenza (o le occorrenze) della relativa recensione dalla tabella. Ciò è possibile poichè all'interno della pagina `listaRecensioni.jsp`, associato alle checkbox vi è una form che attiva la pagina stessa, la quale mediante la parola chiave `ELIMINA` contenuta nella variabile `COMANDOLista`, attiva una condizione IF contenente le istruzioni per l'eliminazione dell'immagine dell'opera recensita e dei dati della stessa.

## 7.3 Modifica e cancellazione dei dati degli annunci

L'operazione di modifica degli annunci è pressochè identica a quanto visto tra le operazioni eseguibili dall'utente, per ulteriori approfondimenti si rimanda perciò alla sezione6, paragrafo 6.2.2. Quello che osserveremo di seguito è come viene effettuata l'operazione di cancellazione di uno o più annunci.

### 7.3.1 Cancellazione con anteprima di un annuncio

Per cancellare un annuncio è necessario dapprima selezionare l'occorrenza voluta e visualizzarne la scheda dei dettagli selezionata nella `listaAnnunci.jsp`, aperta mediante la voce *Modifica/Elimina gli annunci*, posta nella sezione *Mercatino usato* del menù di navigazione. Fatto ciò è sufficiente premere sul pulsante *Elimina* posto sul fondo della pagina, azione che invierà la query `COMANDO=ELIMINA` alla pagina `confermaAnnuncio.jsp` (righe 16-37 listato 9 a pagina 41). Le operazioni contemplate in questo snippet consistono nella ricerca del nodo da eliminare attraverso il suo attributo ID (riga 22), nella selezione del nome dell'eventuale file associato all'annuncio (righe 28-30) e nell'eliminazione del nodo annuncio e del suddetto file (righe 34-37). Un messaggio di conferma avviserà l'amministratore dell'avvenuta operazione.

### 7.3.2 Cancellazione senza anteprima degli annunci

Questa seconda modalità di cancellazione degli annunci prevede, come nella sezione delle recensioni, l'attivazione delle opportune checkbox poste accanto

all'occorrenza degli annunci che si vogliono eliminare nella pagina listaAnnunci.jsp.

L'amministratore deve accedere a quest'ultima pagina tramite la voce *Modifica/Elimina gli annunci*, posta nella sezione *Mercatino usato* del menù di navigazione. Una volta selezionate le occorrenze da eliminare tramite le relative checkbox, l'amministratore premendo il pulsante *Elimina* invia i dati ed il COMANDOLISTA=ELIMINA alla stessa pagina listaAnnunci.jsp, ciò consentirà l'attivazione della condizione che permette la scansione del vettore contenente i riferimenti degli annunci, e la loro eliminazione dall'albero di nodi presente nel file elencoAnnunci.xml.



## 8 Conclusioni

A conclusione del lavoro svolto ci si è resi conto che molto altro si sarebbe potuto fare per rendere completo il progetto, ciò non è stato fatto per questioni di tempo di implementazione e per evitare, per perseguire ognuno dei seguenti punti, una completa ristrutturazione di quanto analizzato finora:

- controllo campi vuoti: il sito non è stato ottimizzato con gli opportuni artefatti al fine di evitare l'inserimento di valori nulli all'interno delle tabelle o del file XML;
- utilizzo delle mail di conferma: utili al fine di consentire all'utente di conoscere tutte le operazioni effettuate all'interno del sito;
- messaggi di errore: ad esclusione degli errori generati dal server, non si sono realizzati appositi messaggi di errore, come ad esempio per l'inserimento di un tipo di dato sbagliato (es. inserimento di una stringa al posto di una data);
- formattazione del file XML: la modifica e la cancellazione di annunci, operazioni che comportano la ricerca e la cancellazione di occorrenze nel file, causano l'inserimento di inopportuni spazi bianchi che non si è riusciti ad eliminare;
- aggiornamento istantaneo della lista degli annunci: la cancellazione senza anteprima di uno o più annunci, sebbene venga registrata nel file, non rende visibile in tempo reale il risultato di tale operazione nella lista associata.

## A Figure

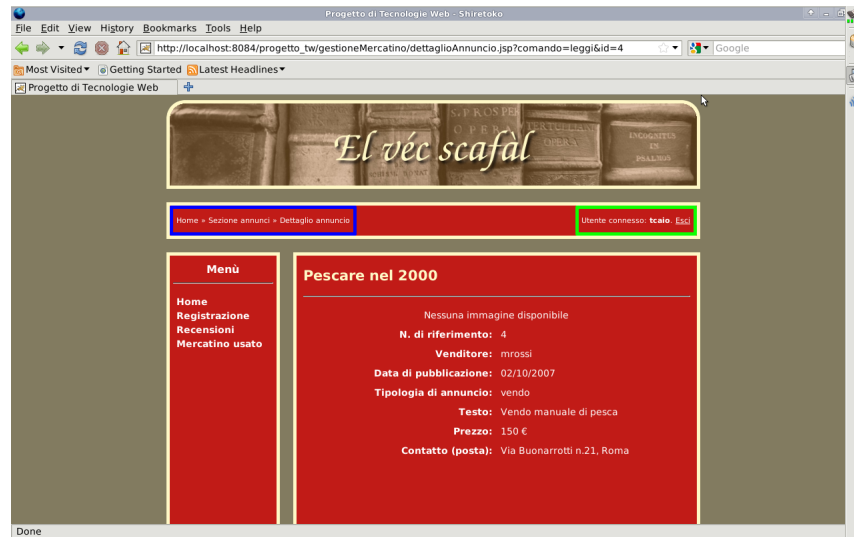


Figura 1: porzione della pagina di una recensione. Nel riquadro blu sono evidenziate le molliche di pane mentre nel riquadro verde è evidenziato il link necessario per l'uscita dall'area riservata del sito.

Progetto di Tecnologie Web - Shiretoko

File Edit View History Bookmarks Tools Help

http://localhost:8084/progetto\_tw/gestioneUtenti/registrazioneUtente.jsp?comando=inserisci

Most Visited Getting Started Latest Headlines

Progetto di Tecnologie Web

El véc scafal

Home > Sezione registrazione utenti > Inserisci dati utente Benvenuto ospite!

**Menù**

- Home
- Registrazione
- Accedi
- Recensioni
- Mercatino usato

**Crea un nuovo account**

Sei già registrato? [Accedi al sito!](#)

INSERISCI I TUOI DATI—

Nome (max 20 caratteri)  
Tizio

Cognome (max 20 caratteri)  
Caio

E-mail (max 50 caratteri)  
tcaio@mail.it

INSERISCI NICKNAME E PASSWORD—

Nickname (max 20 caratteri)  
Itcaio

Done

Figura 2: porzione della pagina `registrazioneUtente.jsp` contenente la form di registrazione. Da notare la query associata alla URL e contenuta nella barra degli indirizzi del browser.

Progetto di Tecnologie Web - Shiretoko

File Edit View History Bookmarks Tools Help

http://localhost:8084/progetto\_tw/gestioneAccesso/accesso.jsp

Most Visited Getting Started Latest Headlines

Progetto di Tecnologie Web

El véc scafal

Home > Sezione accesso > Accesso utente Benvenuto ospite!

**Menù**

- Home
- Registrazione
- Accedi
- Recensioni
- Mercatino usato

**Accesso all'area riservata agli utenti registrati**

Non sei ancora registrato? [Registrati al sito!](#)

INSERISCI NICKNAME E PASSWORD—

Nickname  
Itcaio

Password  
\*\*\*\*\*

Accedi Cancella

Done

Figura 3: porzione della pagina `accesso.jsp` contenente la form tramite cui effettuare l'accesso al sito.

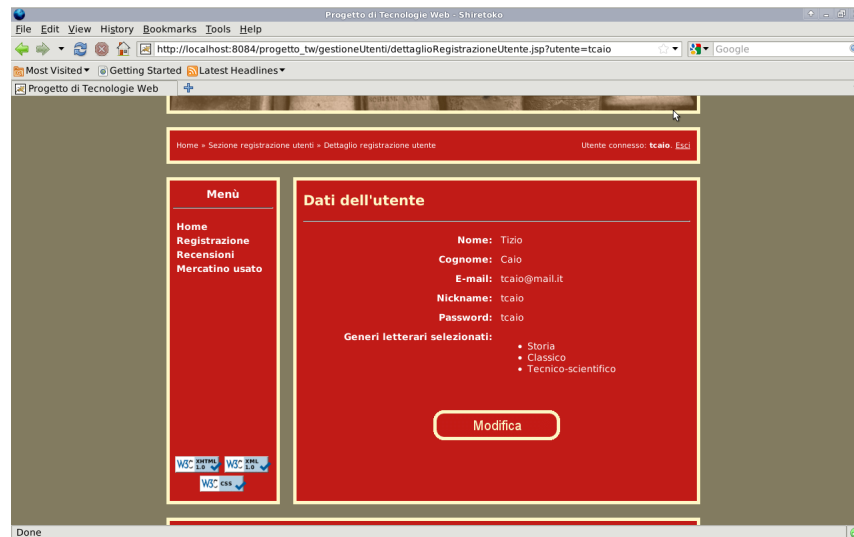


Figura 4: porzione della pagina dettaglioUtente.jsp contenente la scheda riassuntiva dei dati inseriti dall'utente.

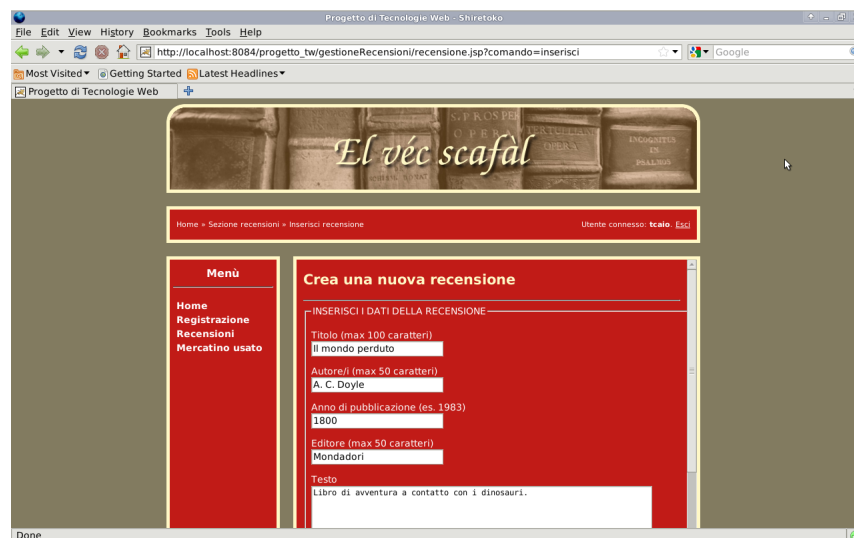


Figura 5: porzione della pagina recensione.jsp contenente la form tramite cui inserire una recensione di un libro nel database. Da notare la URL con la relativa query associata alla pagina, e posta nella barra degli indirizzi del browser.



Figura 6: porzione della pagina `listaRecensioni.jsp` contenente la lista delle recensioni inserite dall'utente e modificabili dallo stesso.

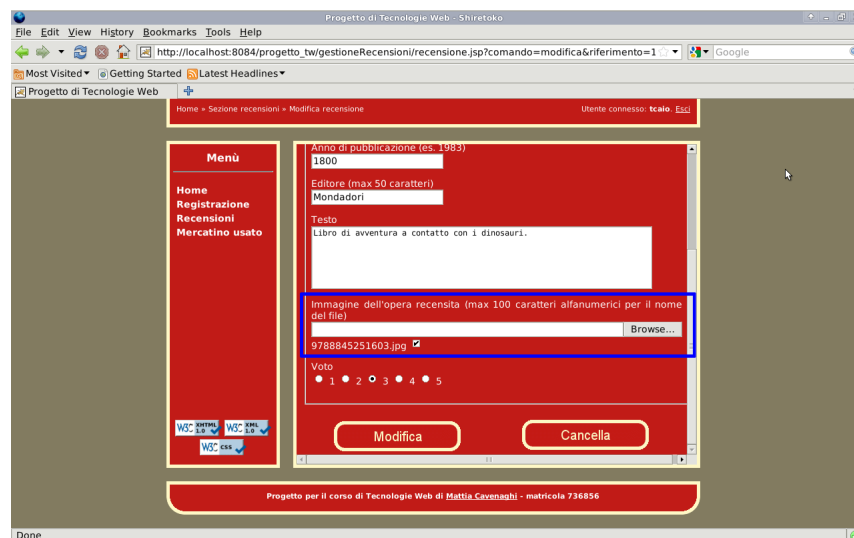


Figura 7: porzione della pagina `recensione.jsp` contenente la form tramite cui modificare la recensione di un libro memorizzata nel database. Da notare la URL con la relativa query associata alla pagina e posta nella barra degli indirizzi del browser. Nel riquadro blu è evidenziato il campo e la checkbox associati all'immagine della copertina dell'opera recensita.

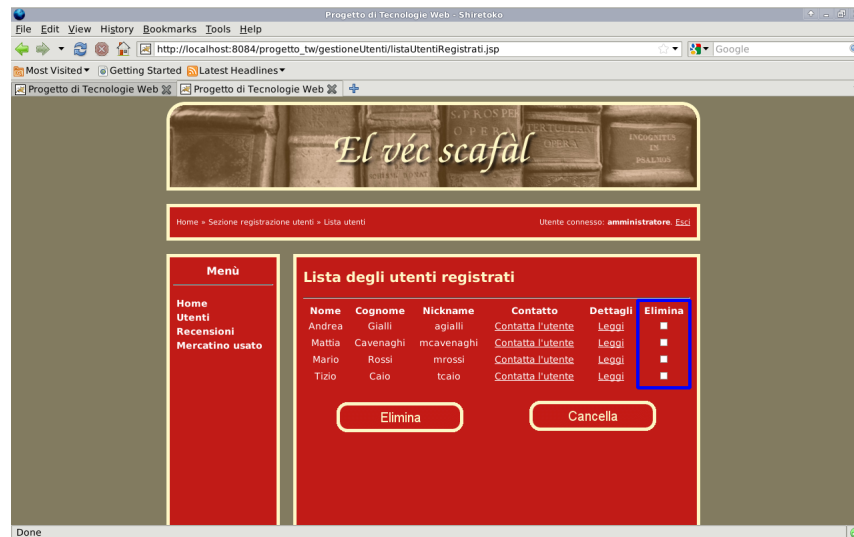


Figura 8: porzione della pagina `listaUtentiRegistrati.jsp` contenente la lista di tutti gli utenti registrati al sito. Da notare la URL con la relativa query associata alla pagina e posta nella barra degli indirizzi del browser. Nel riquadro blu sono evidenziate le checkbox per la cancellazione senza anteprima.

## B Listati di codice

### B.1 Le molliche di pane

```
1  ...
2  <%
3  String [][] molliche = new String [2][3];
4
5  molliche [0][0] = "Home";
6  molliche [0][1] = "Sezione recensioni";
7  molliche [0][2] = "Dettaglio recensione";
8
9  molliche [1][0] = "/progetto_tw/index.jsp ";
10
11 if (request.getQueryString() != null)
12     molliche [1][2] = request.getRequestURL().toString() + "?" +
13         request.getQueryString().replace("&", "&");
14 else
15     molliche [1][2] = request.getRequestURL().toString();
16 %>
17 <c:choose>
18     <c:when test="\${nickname eq 'amministratore'}">
19         <% molliche [1][1] = "/progetto_tw/gestioneRecensioni/
20             listaRecensioni.jsp "; %>
21     </c:when>
22     <c:otherwise>
23         <% molliche [1][1] = "/progetto_tw/gestioneRecensioni/recensione.
24             jsp?comando=inserisci "; %>         </c:otherwise>
25 </c:choose>
26 <% session.setAttribute("molliche", molliche); %>
27 ...
```

Listing 1: porzione di codice appartenente alla pagina dettaglioRecensione.jsp che consente la realizzazione delle molliche di pane. Questo listato è riferito alle molliche realizzate per la pagina dettaglioRecensione.jsp.

### B.2 Il menù di navigazione

```
1  ...
2  <c:choose>
3  <c:when test="\${empty nickname}">
4      Menù per l'ospite o l'utente non loggato
5  </c:when>
6  <c:otherwise>
```

```
7 <c:if test="${nickname != 'amministratore'}"
8   Menù per l'utilizzatore utente
9 </c:if>
10 <c:if test="${nickname == 'amministratore'}">
11   Menù per l'utilizzatore amministratore
12 </c:if>
13 </c:otherwise>
14 </c:choose>
15 ...
```

Listing 2: porzione di codice appartenente alla pagina intestazione.jsp che consente la creazione e visualizzazione del menù. A seconda della tipologia di utilizzatore viene stampato un differente menù.

### B.3 Registrazione, modifica ed eliminazione dei dati degli utenti

```
1 ...
2 <sql:setDataSource
3   url="jdbc:mysql://localhost/progetto_tw"
4   driver="com.mysql.jdbc.Driver"
5   user="root"
6   password="" />
7
8 <c:if test="${param.comando eq 'inserisci' || param.comando eq '
   modifica'}">
9 <%
10 Enumeration parametri = request.getParameterNames();
11 int i = 0;
12
13 while(parametri.hasMoreElements()) {
14   i++;
15
16   if(parametri.nextElement().equals("generePreferito")) {
17     String [] generi = request.getParameterValues("generePreferito");
18     String generiConcat = null;
19
20     for(int j = 0; j < generi.length; j++) {
21       if(j == 0)
22         generiConcat = (generi[j] + ",");
23       else
24         generiConcat += (generi[j] + ",");
25     }
26
27     pageContext.setAttribute("generiConcat", generiConcat);
28   }
```



```
29 }
30 %>
31 </c:if>
32
33 <c:if test="${param.comando eq 'inserisci'}">
34   <sql:update>
35     INSERT INTO utenti (nome, cognome, email, nickname, password,
36       generi) VALUES (?, ?, ?, ?, ?, ?)
37   <sql:param value="${param.nome}"/>
38   <sql:param value="${param.cognome}"/>
39   <sql:param value="${param.email}"/>
40   <sql:param value="${param.nickname}"/>
41   <sql:param value="${param.password}"/>
42   <sql:param value="${param.generiConcat}"/>
43 </sql:update>
44 </c:if>
45
46 <c:if test="${param.comando eq 'modifica'}">
47   <sql:update>
48     <c:if test="${nickname eq 'amministratore'}">
49       UPDATE utenti SET nome = ?, cognome = ?, nickname = ?, password
50         = ?, email = ?, generi = ? WHERE nickname = ?
51     <sql:param value="${param.nome}"/>
52     <sql:param value="${param.cognome}"/>
53     <sql:param value="${param.nickname}"/>
54     <sql:param value="${param.password}"/>
55     <sql:param value="${param.email}"/>
56     <sql:param value="${param.generiConcat}"/>
57     <sql:param value="${param.utente}"/>
58   </c:if>
59   <c:if test="${nickname ne 'amministratore'}">
60     UPDATE utenti SET nome = ?, cognome = ?, email = ?, generi = ?
61       WHERE nickname = ?
62   <sql:param value="${param.nome}"/>
63   <sql:param value="${param.cognome}"/>
64   <sql:param value="${param.email}"/>
65   <sql:param value="${param.generiConcat}"/>
66   <sql:param value="${param.utente}"/>
67 </c:if>
68 </sql:update>
69 </c:if>
70
71 <c:if test="${param.comando eq 'elimina'}">
72   <sql:update>
73     DELETE FROM utenti WHERE nickname = ?
```

```
72 <sql:param value="{param.utente}"/>
73 </sql:update>
74 </c:if>
75 ...
76 Messaggi di avvenuta operazione.
77 ...
```

Listing 3: porzione di codice appartenente alla pagina confermaRegistrazione.jsp che consente l'inserimento dei dati di un nuovo utente nella relativa tabella.

## B.4 Accesso al sito

```
1 ...
2 <c:choose>
3 <c:when test="{empty nickname}">
4 <sql:setDataSource
5 url="jdbc:mysql://localhost/progetto_tw"
6 driver="com.mysql.jdbc.Driver"
7 user="root"
8 password="" />
9
10 <sql:query var="query">
11 SELECT nickname, password FROM utenti WHERE nickname = ? AND
    password = ?
12 <sql:param value="{param.nickname}"/>
13 <sql:param value="{param.password}"/>
14 </sql:query>
15 </c:when>
16 </c:choose>
17 ...
18 <c:choose>
19 <c:when test="{empty nickname && param.comando eq 'accedi'}">
20 <c:if test="{query.rowCount eq 1}">
21 <c:set var="nickname" value="{param.nickname}" scope="session"
    "/>
22 </c:if>
23 </c:when>
24
25 <c:when test="{!empty nickname && param.comando eq 'esci'}">
26 <c:remove var="nickname" scope="session"/>
27 </c:when>
28 </c:choose>
29 ...
30 Stampa dei messaggi di benvenuto o di errore riscontrato durante la
    fase di login.
```

31 ...

Listing 4: porzione di codice appartenente alla pagina confermaAccesso.jsp. Il codice schematizza la struttura ed i costrutti che realizzano il processo di accesso ed uscita dall'area riservata del sito da parte di un utilizzatore.

## B.5 Inserimento, modifica e cancellazione di una recensione

```
1 ...
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
4 <%@ page import="java.io.File , org.apache.commons.fileupload .
    servlet.ServletFileUpload , org.apache.commons.fileupload.disk .
    DiskFileItemFactory , org.apache.commons.fileupload.* , java.sql
    .*, java.text.*, java.util.*" %>
5 ...
6 <%
7 ...
8 Istanza delle variabili di servizio contenenti i dati della
    recensione ed i comandi.
9 ...
10
11 boolean isMultipart = ServletFileUpload.isMultipartContent(request
    );
12
13 if (!isMultipart) {
14 %>
15 ...
16 Connessione al database.
17 ...
18 <c:if test="${param.comando eq 'elimina'}">
19     <sql:query var="query">
20         SELECT nome_file FROM recensioni WHERE riferimento = ?
21     <sql:param value="${param.riferimento}"/>
22 </sql:query>
23
24 <c:forEach items="${query.rows}" var="risultato">
25     <c:set var="nomeFile" value="${risultato.nome_file}" scope="page
        "/>
26 </c:forEach>
27
28 <sql:update>
29     DELETE FROM recensioni WHERE riferimento = ?
30 <sql:param value="${param.riferimento}"/>
31 </sql:update>
```

```
32  ...
33  Messaggio di avvenuta cancellazione della recensione da parte dell
    'amministratore.
34  ...
35  <%
36  if (!pageContext.getAttribute("nomeFile").equals("nessuno")) {
37      File file = new File(config.getServletContext().getRealPath("/")
        + "../immagini//recensioni//" + pageContext.getAttribute("
        nomeFile"));
38      file.delete();
39  }
40  %>
41  </c:if>
42  <%
43  }
44  else {
45      FileItemFactory factory = new DiskFileItemFactory();
46      ServletFileUpload upload = new ServletFileUpload(factory);
47      List items = null;
48
49      try {
50          items = upload.parseRequest(request);
51      }
52      catch (FileUploadException e) {
53          e.printStackTrace();
54      }
55
56      Iterator itr = items.iterator();
57
58      while (itr.hasNext()) {
59          FileItem item = (FileItem) itr.next();
60
61          if (item.isFormField()) {
62              String name = item.getFieldName();
63              String value = item.getString();
64              ...
65              Parsing dei dati della recensione.
66              ...
67          }
68          else {
69              try {
70                  String itemName = item.getName();
71
72                  if (!itemName.isEmpty()) {
73                      File savedFile = new File(config.getServletContext().
                        getRealPath("/") + "../immagini//recensioni//" + itemName);
```

```
74         item.write(savedFile);
75         nomeFile = itemName;
76     }
77     else {
78         nomeFile = "nessuno";
79     }
80 }
81 catch (Exception e) {
82     e.getMessage();
83 }
84 }
85 }
86 ...
87 Apertura della connessione al database.
88 ...
89
90 if(comando.equals("inserisci")) {
91     ...
92     Query di inserimento dei dati.
93     ...
94     %>
95     ...
96     Messaggio di avvenuta registrazione dei dati.
97     <%
98     }
99     else if(comando.equals("modifica")) {
100         if(!immagineCheck.equals("") && !immagineCheck.equals("
            immagineMem")) {
101             String nomeFileCancellare = "";
102
103             query = "SELECT nome_file FROM recensioni WHERE riferimento = '"
                + riferimento + "'";
104             risultato = statement.executeQuery(query);
105             risultato.beforeFirst();
106
107             while(risultato.next())
108                 nomeFileCancellare = risultato.getString("nome_file");
109                 risultato.close();
110
111             if(!nomeFileCancellare.equals("nessuno")) {
112                 File file = new File(config.getServletContext().getRealPath("/")
                    + "../immagini//recensioni//" + nomeFileCancellare);
113                 file.delete();
114             }
115         }
116     }
```

```
114     else if(nomeFile.equals("nessuno") && immagineCheck.equals("
        immagineMem")) {
115         query = "SELECT nome_file FROM recensioni WHERE riferimento = '"
            + riferimento + "'";
116         risultato = statement.executeQuery(query);
117         risultato.beforeFirst();
118
119         while(risultato.next())
120             nomeFile = risultato.getString("nome_file");
121         risultato.close();
122     }
123     else if(nomeFile.equals("nessuno") && immagineCheck.equals("") &&
        immagineCheck.equals("immagineMem")) {
124         String nomeFileCancellare = "";
125
126         query = "SELECT nome_file FROM recensioni WHERE riferimento = '"
            + riferimento + "'";
127         risultato = statement.executeQuery(query);
128         risultato.beforeFirst();
129
130         while(risultato.next())
131             nomeFileCancellare = risultato.getString("nome_file");
132         risultato.close();
133
134         if(!nomeFileCancellare.equals("nessuno")) {
135             File file = new File(config.getServletContext().getRealPath("/")
                + "../immagini/recensioni/" + nomeFileCancellare);
136             file.delete();
137         }
138     }
139
140     if(session.getAttribute("nickname").equals("amministratore")) {
        sdf.applyPattern("yyyy-MM-dd");
141         ...
142         Query di modifica dei dati da parte dell'amministratore.
143         ...
144     }
145     else {
146         ...
147         Query di modifica dei dati da parte dell'utente.
148         ...
149     }
150 %>
151     ...
152     Messaggio di avvenuta modifica dei dati.
```

153 ...

Listing 5: porzione di codice appartenente alla pagina confermaRecensione.jsp. Il codice schematizza la struttura ed i costrutti che realizzano il processo di inserimento dei dati relativi ad una recensione inserita da un utente, ed il processo di modifica e cancellazione dei dati relativi ad una recensione da parte dell'amministratore o di un utente.

## B.6 Il file elencoAnnunci.xml

```
1 ...
2 <annunci xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="schemaElencoAnnunci.xsd">
3   <annuncio id="1" autore="mrossi" data="2005-03-02" tipologia="
     vendo">
4     <titolo>Enciclopedia Britannica</titolo>
5     <testo>Vendo i primi dieci volumi dell'enciclopedia britannica.</
       testo>
6     <immagine>encbritannica.jpg</immagine>
7     <prezzo valuta="euro">1500</prezzo>
8     <contatto tipologia="telefono">1234567890</contatto>
9   </annuncio>
10 ...
11 </annunci>
```

Listing 6: porzione del file elencoAnnunci.xml, contenente gli annunci economici inseriti dagli utenti.

## B.7 Il file listaAnnunci.xsl

```
1 ...
2 <xsl:param name="comando"/>
3 <xsl:param name="utente"/>
4
5 <xsl:template match="/annunci">
6   <xsl:choose>
7     <xsl:when test="not(annuncio)">
8       ...
9       Stampa del messaggio di assenza di annunci nel file XML.
10      ...
11    </xsl:when>
12    <xsl:otherwise>
13      ...
14      Stampa dell'intestazione delle colonne.
15      ...
16
```

```
17 <xsl:choose>
18 <xsl:when test="($utente='amministratore') or (($utente!=
    amministratore') and ($comando!='modifica'))">
19 <xsl:for-each select="annuncio">
20 <xsl:call-template name="stampaDati"/>
21 </xsl:for-each>
22 </xsl:when>
23 <xsl:when test="($utente!='amministratore') and ($comando=
    modifica')">
24 <xsl:choose>
25 <xsl:when test="not(annuncio[@autore=$utente])">
26 <tr><td colspan="6">Non ci sono annunci nel database</td></
    tr>
27 </xsl:when>
28 <xsl:otherwise>
29 <xsl:for-each select="annuncio[@autore=$utente]">
30 <xsl:call-template name="stampaDati"/>
31 </xsl:for-each>
32 </xsl:otherwise>
33 </xsl:choose>
34 </xsl:when>
35 </xsl:choose>
36
37 ...
38 Comandi dedicati all'amministratore.
39 ...
40 </xsl:otherwise>
41 </xsl:choose>
42 </xsl:template>
43
44 <xsl:template name="formattaData">
45 <xsl:param name="data" />
46 <xsl:variable name="anno" select="substring-before($data, '-')"/>
47 <xsl:variable name="mese" select="substring-before(substring-after
    ($data, '-'), '-')"/>
48 <xsl:variable name="giorno" select="substring-after(substring-
    after($data, '-'), '-')"/>
49 <xsl:value-of select="concat($giorno, '/', $mese, '/', $anno)"/>
50 </xsl:template>
51
52 <xsl:template name="stampaDati">
53 ...
54 Tabella contenente i dati dell'annuncio selezionato.
55 ...
56
57 <xsl:call-template name="formattaData">
```



```
58 <xsl:with-param name="data" select="@data" />
59 </xsl:call-template>
60 ...
61 Stampa dei link "Leggi" e delle checkbox dei eliminazione.
62 ...
63 </xsl:template>
64 ...
```

Listing 7: schematizzazione del file listaAnnunci.xsl, contenente i template per la stampa della lista degli annunci.

## B.8 Il file dettaglioAnnuncio.xsl

```
1 ...
2 <xsl:param name="comando"/>
3 <xsl:param name="utente"/>
4 <xsl:param name="id"/>
5
6 <xsl:template match="/annunci">
7   <xsl:for-each select="annuncio[@id=$id]">
8     ...
9     Stampa dei dati dell'annuncio.
10    ...
11
12    <xsl:apply-templates select="prezzo"/>
13    ...
14    <th>Contatto
15      (<xsl:apply-templates select="contatto"/>):</th><td><xsl:value-
16        of select="contatto"/></td>
17    ...
18  </xsl:for-each>
19 </xsl:template>
20
21 <xsl:template match="contatto">
22   <xsl:value-of select="@tipologia"/>
23 </xsl:template>
24
25 <xsl:template match="prezzo">
26   <xsl:choose>
27     <xsl:when test="@valuta='euro'">
28       &#8364;
29     </xsl:when>
30     <xsl:when test="@valuta='dollaro'">
31       &#36;
32     </xsl:when>
33   </xsl:choose>
```

```
33 </xsl:template>
34
35 ...
36 Template di formattazione della data.
37 ...
```

Listing 8: porzione di codice del file dettaglioAnnuncio.xsl. In questo listato vengono schematizzate le operazioni eseguite per la stampa dei dati relativi ad un annuncio.

## **B.9 Inserimento, modifica ed eliminazione di un annuncio**

```
1 ...
2 Istanza delle molliche di pane.
3 ...
4
5 <%
6 ...
7 Istanza delle variabili utili alle operazioni di parsing e di
  esecuzione delle query SQL.
8 ...
9
10 DocumentBuilderFactory factory2 = DocumentBuilderFactory.
   newInstance();
11 Document doc = factory2.newDocumentBuilder().parse(new File(config
   .getServletContext().getRealPath("/") + "gestioneMercatino//
   elencoAnnunci.xml"));
12
13 boolean isMultipart = ServletFileUpload.isMultipartContent(request
   );
14
15 if (!isMultipart) {
16   if (request.getParameter("comando").equals("elimina")) {
17     NodeList nodiAnnuncio = doc.getElementsByTagName("annuncio");
18
19     for(int i=0; i<nodiAnnuncio.getLength(); i++) {
20       NamedNodeMap attributi = nodiAnnuncio.item(i).getAttributes();
21
22       if (attributi.getNamedItem("id").getNodeValue().equals(request.
         getParameter("riferimento"))) {
23         NodeList nodiFigli = nodiAnnuncio.item(i).getChildNodes();
24
25         for(int j=0; j<nodiFigli.getLength(); j++) {
26           Node nodo = nodiFigli.item(j);
27
28           if (nodo.getNodeName().equals("immagine"))
```

```
29         if (nodo.hasChildNodes())
30             nomeFile = nodo.getFirstChild().getNodeValue();
31     }
32
33     ...
34     Cancellazione del file.
35     ...
36
37     doc.getDocumentElement().removeChild(nodiAnnuncio.item(i));
38 }
39 }
40 ...
41 Messaggio di operazione completata.
42 ...
43 else {
44     ...
45     Salvataggio del file sul server, parsing dei dati ed istanza
        della connessione SQL.
46     ...
47
48     Element root = doc.getDocumentElement();
49
50     if (comando.equals("inserisci")) {
51         ...
52         Ricerca del contatto dell'utente nel database.
53         ...
54
55         Node lastChild = root.getLastChild();
56         NamedNodeMap attributi = lastChild.getPreviousSibling().
            getAttributes();
57         int attributoId = Integer.parseInt(attributi.item(2).
            getNodeValue()) + 1;
58
59         Element annuncio = doc.createElement("annuncio");
60         annuncio.setAttribute("autore", session.getAttribute("nickname
            ").toString());
61         annuncio.setAttribute("data", session.getAttribute("
            dataPubblicazione").toString());
62         annuncio.setAttribute("id", String.valueOf(attributoId));
63         annuncio.setAttribute("tipologia", tipologiaAnnuncio);
64
65         Element nodoTitolo = doc.createElement("titolo");
66         annuncio.appendChild(nodoTitolo);
67         Text titoloTesto = doc.createTextNode(titolo);
68         nodoTitolo.appendChild(titoloTesto);
69
```

```
70     ...
71     Istanza dei rimanenti nodi.
72     ...
73
74     root.insertBefore(annuncio, lastChild);
75
76     ...
77     Messaggio di completamento dell'operazione.
78     ...
79 }
80 else if(comando.equals("modifica")) {
81     NodeList nodiAnnuncio = doc.getElementsByTagName("annuncio");
82     Node nodoPrecedente = null;
83     String autore = "";
84
85     for(int i=0; i<nodiAnnuncio.getLength(); i++) {
86         NamedNodeMap attributi = nodiAnnuncio.item(i).getAttributes()
87             ;
88
89         if(attributi.getNamedItem("id").getNodeValue().equals(
90             riferimento)) {
91             nodoPrecedente = nodiAnnuncio.item(i).getPreviousSibling();
92             NodeList nodiFigli = nodiAnnuncio.item(i).getChildNodes();
93
94             if(session.getAttribute("nickname").equals("amministratore")
95                 )
96                 autore = attributi.getNamedItem("autore").getNodeValue();
97             else
98                 autore = session.getAttribute("nickname").toString();
99
100             ...
101             Elaborazione del file immagine.
102             ...
103
104             doc.getDocumentElement().removeChild(nodiAnnuncio.item(i));
105         }
106     }
107
108     ...
109     Creazione del nuovo nodo.
110     ...
111     root.insertBefore(annuncio, nodoPrecedente);
112 }
```

```
113 doc.normalize();
114 TransformerFactory factory = TransformerFactory.newInstance();
115 Transformer transformer = factory.newTransformer();
116 transformer.setOutputProperty(OutputKeys.INDENT, "yes");
117 transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-
    amount", "4");
118
119 StringWriter sw = new StringWriter();
120 StreamResult result = new StreamResult(sw);
121 DOMSource source = new DOMSource(doc);
122 transformer.transform(source, result);
123 String xmlString = sw.toString();
124
125 FileWriter fstream = new FileWriter(config.getServletContext().
    getRealPath("/") + "gestioneMercatino//elencoAnnunci.xml");
126 BufferedWriter file = new BufferedWriter(fstream);
127 file.write(xmlString);
128 file.close();
129 %>
130 ...
```

Listing 9: porzione di codice del file confermaAnnuncio.jsp. In questo listato vengono schematizzati costrutti necessari ad inserire, modificare ed eliminare gli annunci inseriti nel file elencoAnnunci.xml.

## B.10 La lista degli utenti registrati

```
1 ...
2 Istanza delle molliche e creazione della connessione al database.
3 ...
4
5 <c:if test="${param.comando eq 'elimina' and !empty param.comando
    }">
6   <c:forEach items="${paramValues.nickname}" varStatus="i">
7     <sql:update>
8       DELETE FROM utenti WHERE nickname = ?
9       <sql:param value="${paramValues.nickname[i.index]}" />
10    </sql:update>
11  </c:forEach>
12 </c:if>
13 ...
14 Corpo della pagina.
15 ...
```

Listing 10: porzione di codice del file ListaUtentiRegistreat.jsp. In questo listato è riportata la condizione IF che consente l'eliminazione di una o più occorrenze

*utente.*