Progettazione e Realizzazione di un Modulo per i Percorsi Virtuali in una Base di Conoscenza Informatica

Danilo Aimini

Relatori:

Mario Giovanni C. A. Cimino
Giuseppe Lettieri

Indice generale

1. Specifiche

1.1. Introduzione

L'obiettivo del lavoro di questa tesi è l'implementazione di una funzionalità di gestione di tour virtuali all'interno dell'applicazione CHKB. L'iniziativa CHKB, realizzato dal Dipartimento di Ingegneria dell'Informazione dell'Università di Pisa in collaborazione con il progetto HMR del Dipartimento di Informatica dell'Università di Pisa, consiste in un'applicazione Open Source con interfaccia Web che permette la catalogazione e l'organizzazione delle informazioni e dei reperti conservati presso multiple collezioni, pubbliche o private, all'interno di un sistema unico e altamente configurabile.

La funzionalità che si intende realizzare consiste nella gestione e visualizzazione di percorsi tematici, cioè liste ordinate di reperti o schede informative salvate nella base di dati, a fine divulgativo o come aiuto per l'organizzazione di mostre. I gestori delle collezioni catalogate sulla piattaforma, o anche i normali utenti se autorizzati, possono usufruire di questa funzionalità direttamente dal pannello di gestione di CHKB, da cui sarà possibile creare e gestire tali tour virtuali tramite procedure guidate che non necessitano di particolari conoscenze tecniche.

Nel seguito saranno descritti solo tabelle, funzioni e oggetti aggiunti al progetto per l'introduzione di questa funzionalità; per la definizione delle funzionalità già presenti rimanderemo più volte all'appropriato capitolo della tesi di Angelo Biagini, “Progetto e realizzazione di CHKB, una base di conoscenza web per il Museo degli Strumenti per il Calcolo”.

1.2. Strumenti Utilizzati

L'applicazione è stata realizzata su una macchina con sistema operativo Windows 8.1 e web server Apache 2 (versione 2.4.12) con modulo PHP (versione 5.5.27) per la realizzazione della logica applicativa; il DBMS utilizzato è PostgreSQL (versione 9.4.4-1). Alcune modifiche al codice precedente si sono rese necessarie per consentire all'applicativo di funzionare correttamente sia su sistema operativo Windows che Linux, principalmente per differenze nella gestione dei path relativi sui due OS.

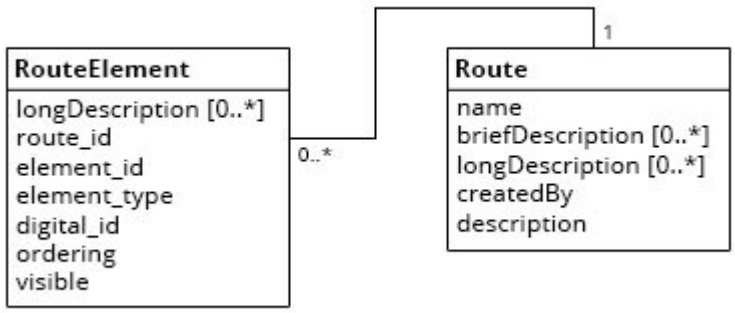
L'applicazione è stata successivamente testata su una macchina con sistema operativo Ubuntu Server 12.04 e web server Apache 2 (versione 2.2.22) con modulo PHP (versione 5.3.10) per la realizzazione della logica applicativa; il DBMS utilizzato è PostgreSQL versione 9.1.6.

I sorgenti PHP sono commentati in modo da consentire la generazione automatica della documentazione tramite lo strumento PHPDocumentor 2.

2. Requisiti

2.1. Requisiti della Base di Dati

Al fine di semplificare le operazioni di gestione dei percorsi, si è preferito dividere gli stessi in due entità separate: il Percorso (Route), che conserva informazioni quali nome, descrizione multilingua e username dell'utente che lo ha creato, e Elemento del Percorso (Route Element), che descrive il singolo reperto o scheda informativa. Si noti che descrizione e contenuto digitale mostrati per un reperto all'interno del percorso sono specifici per il percorso stesso; uno stesso elemento può comparire in diversi percorsi associato a differenti immagini/video e descrizioni, in quanto questi devono descrivere il ruolo di quell'oggetto nel percorso specifico.



Schema concettuale della Base di Dati

1. Percorsi (Route), prevedono le seguenti informazioni

    - nome (name)
    - descrizione breve (briefDescription)
    - descrizione lunga (longDescription)
    - creatore (created_by)

2. Elementi del Percorso (Route Element), prevedono le seguenti informazioni

    - descrizione lunga (longDescription)
    - percorso (route_id)
    - tipo elemento (element_type)
    - ID elemento (element_id)
    - ID contenuto digitale associato (digital_ID)
    - ordine nel percorso (ordering)
    - visibilità (visible)

2.2. Requisiti dell'Applicazione

- L'interfaccia utente deve permettere all'*utente pubblico* di visualizzare una lista dei percorsi disponibili, e di navigare gli stessi. Gli elementi del percorso sono presentati uno alla volta, nell'ordine definito nel percorso stesso. Devono essere disponibili dei pulsanti appositi per procedere all'elemento successivo, tornare al precedente o raggiungere la pagina specifica del reperto o della scheda informativa.
- L'interfaccia utente deve permettere all'*utente registrato*, se dotato dei permessi necessari, di aggiungere, modificare ed eliminare i percorsi e, per ognuno di essi, aggiungere, modificare, riordinare ed eliminare gli elementi del percorso.
- L'interfaccia utente deve essere multilingua e mostrare le descrizioni nella lingua legata all'utente che visualizza, come definito nei requisiti di CHKB al punto 2.3.2.2. G (lingua di default per l'*utente pubblico*, lingua preferita per l'*utente registrato*).
- L'interfaccia di creazione e modifica dei percorsi deve presentarsi solo agli utenti con i permessi necessari, che devono essere presenti tra le *capabilities* associabili agli utenti registrati.
- La modifica e rimozione di un percorso e degli elementi ad esso associati deve essere possibile solo per l'utente che lo ha creato. E' necessario che il sistema effettui dei controlli di sicurezza per assicurarsene.

3. Progetto della funzionalità

3.1. Progettazione della base di dati

La realizzazione della base di dati comporta una complessità per quanto riguarda la realizzazione della tabella route_element. Un elemento di percorso può infatti essere un reperto oppure una scheda informativa, ma su database questi due oggetti sono salvati su tabelle differenti ed è presente una sovrapposizione degli identificatori che risultano così non più univoci. Per ovviare a questo problema, la suddetta tabella salva la coppia element_id – element_type; in questo modo il secondo parametro, di tipo enumerato **route_element_t,** determina in quale tabella andare a cercare l'elemento di ID element_id. E' per questo motivo che la colonna element_id non è considerata come chiave esterna.

La chiave primaria di questa tabella è quindi composta dalla tripla (*element_id, element_type, route_id*), in quanto uno stesso reperto/scheda informativa può comparire una sola volta in ogni percorso.

3.2. Operazioni

In questo capitolo vengono analizzate le operazioni più significative legate alla nuova funzionalità.

In appendice A.2. sono riportate le query che implementano le operazioni descritte.

3.2.1. Recupero degli elementi appartenenti a un percorso

La cardinalità unaria dell'associazione tra *elemento del percorso* e *percorso* (*RouteElement* (0..*) - (1) *Collection*) consente di accorpare l'associazione presso l'entità *RouteElement*.

3.2.2. Inserimento di un elemento in un percorso

Un elemento del percorso può essere un oggetto di tipo Info o Physical. In entrambi i casi, è sufficiente salvare l'id dell'elemento e nella fetch si provvederà a chiamare la fetch del tipo corrispettivo.

3.2.3. Ricerca di tutti i percorsi in cui è presente un determinato elemento

Durante la visualizzazione di un elemento di un percorso, in un box apposito sono visualizzati in elenco i percorsi che contengono lo stesso elemento.

4. Codice dell'Applicazione

Il codice dell'applicazione segue le stesse specifiche definite per CHKB al paragrafo 3.2: il paradigma utilizzato è MVC (Model-View-Controller), l'architettura è orientata ai dati e le entità Route e RouteElement sono manipolabili tramite oggetti che nascondono l'implementazione e le operazioni di lettura/scrttura su database.

A. Schema della Base di Dati

Sono elencate solamente le aggiunte rispetto alla corrispondente sezione A nella tesi di riferimento.

A.1. Tipi

route_element_t
tipo di *elemento di percorso*

- *physical*
- *info*

A.2. Relazioni

Gli attributi sottolineati costituiscono la chiave primaria.

route (<u>id</u>, name, created_by, description)

route_descriptions (<u>id</u>, <u>is_long</u>, <u>language</u>, content)

route_element (eid, <u>route_id</u>, <u>element_id</u>, <u>element_type</u>, digital_id, ordering, visible)

route_element_descriptions (<u>id</u>, <u>is_long</u>, <u>language</u>, content)

A.3. Query

A.3.1. Recupero degli elementi appartenenti a un percorso

Nell'esempio, si recuperano gli elementi del percorso di ID 1.
Il campo eid contiene un ID univoco necessario per il salvataggio delle descrizioni multilingua; la chiave della tabella route_element è però composta dai tre campi route_id, element_id e element_type.
L'elenco degli elementi è ordinato secondo il campo ordering.

```
SELECT *
FROM route_element AS re
LEFT JOIN route_element_descriptions AS red
ON re.eid = red.id
WHERE re.route_id = 1
ORDER BY re.ordering ASC
```

A.3.2. Inserimento di un elemento in un percorso

Nell'esempio, l'elemento del percorso è un *reperto* di ID 7 che sarà inserito nel *percorso* di ID 1. Il campo *visible* è di tipo booleano e rappresenta la visibilità del percorso a cui l'elemento appartiene (funzionalità prevista ma non implementata). Il campo *digital_id* contiene l'ID del contenuto digitale che sarà visualizzato nel percorso per questo elemento. Il campo *ordering* contiene il numero d'ordine dell'elemento all'interno del percorso. Il campo *eid*, riempito automaticamente e ritornato dalla query, è un valore numerico in sequenza (auto_increment) utilizzato come ID univoco per la gestione delle descrizioni multilingua. (La chiave primaria della tabella route_element resta comunque la terna route_id, element_type, element_id).

```
INSERT INTO route_element
( "element_type", element_id, visible, digital_id, ordering, route_id )
VALUES
('physical', 7, true, 5, 2, 1)
RETURNING eid
```

A.3.3. Ricerca di tutti i percorsi in cui è presente un determinato elemento

Questa query viene eseguita per popolare il box “Percorsi Correlati” durante la visualizzazione del percorso.
La struttura della base di dati è stata scelta proprio per semplificare questo tipo di operazione, che risulterebbe molto più complessa da gestire se il percorso fosse salvato come stringa o su files esterni.

```
SELECT route_id
FROM route_element
WHERE element_type = 'physical'
AND element_id = 1
ORDER BY route_id ASC;
```

B. Organizzazione dei Sorgenti

Di seguito, la lista dei files aggiunti o modificati. Si segue la stessa organizzazione definita nella tesi di riferimento: le voci sottolineate sono cartelle, quelle in corsivo sono *files*. Sono omesse le modifiche necessarie a permettere il corretto funzionamento del software su server con sistema operativo Windows.

- database:
    - *data_definition.sql*: sono state aggiunte tutte le operazioni necessarie a inizializzare la funzionalità di gestione dei percorsi virtuali
- include
    - model
        - *Route.class.php*: percorso
        - *RouteElement.class.php*: informazioni relative a un elemento di un percorso
        - *RouteElementType.class.php*: enumerazione per il tipo di un elemento di un percorso (*physical* o *info*)
        - *User.class.php*: aggiunto l'attributo manage_routes e la corrispondente funzione getter canManageRoutes(), per definire se un utente può gestire i percorsi.
    - view
        - css
            - *common.js*: aggiunte regole css per la visualizzazione dei percorsi.
        - *italian.php*: aggiunte le stringhe di lingua per tutte le nuove operazioni e pagine legate ai percorsi virtuali
        - js: contiene sorgenti javascript per l'interfaccia utente
            - *route_form.js*

        - *routeelement.js*: contiene le funzioni javascript che si occupano di mostrare solo i menu a tendina corrispondenti alle scelte effettuate dall'utente
    - *RouteElementView.class.php*: metodi per la produzione dei form HTML e dell'ouptut per le operazioni relative agli elementi di un percorso
    - *RouteView.class.php*: metodi per la produzione dei form HTML e dell'ouptut per le operazioni relative ai percorsi
    - *UserView.class.php*: aggiunta la possibilità di gestire la capacità di gestione dei percorsi per un utente
- controller
    - *RouteController.class.php*: metodi per il controllo della validità degli input dei form per le operazioni di creazione e modifica delle informazioni generali di un percorso
    - *RouteElementController.class.php*: metodi per il controllo della validità degli input dei form per le operazioni di creazione e modifica degli elementi di un percorso
    - web
        - *moveRouteElement.php*: cambiare l'ordine degli elementi di un percorso
        - *newRoute.php*: creare un nuovo percorso
        - *newRouteElement.php*: aggiungere un elemento di un percorso
        - *removeRoute.php*: rimuovere un percorso
        - *removeRouteElement.php*: rimuovere un elemento di un percorso
        - *showRoute.php*: visualizzare un percorso
        - *showRoutes.php*: visualizzare la lista dei percorsi

- *updateRoute.php*: modificare gli elementi appartenenti a un percorso
- *updateRouteElement.php*: modificare le informazioni di un elemento di un percorso
- *updateRouteInfo.php*: modificare le informazioni generali di un percorso