Analisi e sviluppo di una web application basata sui framework JPA, JSF, CDI, per l'amministrazione di attività di trasferimento tecnologico

Tommaso Levato, Alessio Sarullo, Giulio Galvan

12 novembre 2013

Sommario

Indice

Capitolo 1

Introduzione

Capitolo 2

Analisi

2.1 Casi d'uso

Di seguito elenchiamo i principali casi d'uso per ciascun tipo di utente che interagisce col sistema.

Operatore Una rappresentazione grafica dei casi d'uso dell'Operatore è disponibile in figura 2.1

1. Inserimento di una nuova Convenzione/Contributo

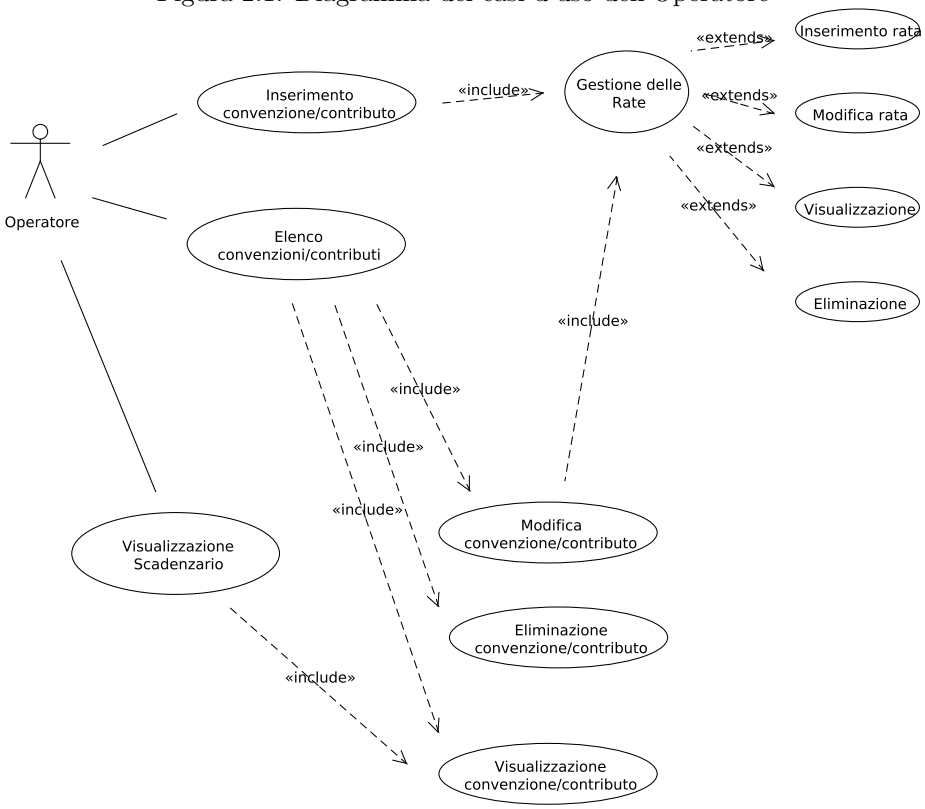
   Percorso base: l'operatore, una volta effettuato il login, clicca su "Crea una convenzione/contributo"; viene visualizzata una schermata suddivisa in varie schede, ognuna corrispondente ad un passo della procedura. E' possibile passare da una vista all'altra mediante i pulsanti "Avanti" e "Indietro". I passi sono:

   (a) Inserimento dei dati della convenzione/contributo

   In questa scheda sono elencati tutti i campi necessari per la definizione di una convenzione/contributo, che l'Operatore deve compilare. Tali campi sono:

   - Il titolo
   - Il numero di protocollo
   - L'UAR
   - La tipologia
   - Il responsabile scientifico
     Per selezionare un responsabile scientifico è possibile usare l'apposito menù a tendina o, in alternativa, qualora la persona cercata non sia nell'elenco, aggiungerla cliccando sul pulsante "Aggiungi".
   - Il referente
   - La dittà
     Per selezionare una dittà si può usare l'apposito menù a tendina

Figura 2.1: Diagramma dei casi d'uso dell'Operatore



o, se la dittà cercata non fosse presente nell'elenco, aggiungerne una nuova cliccando sul pulsante "Aggiungi".

- Il nome del progetto CIA
- Il Repertorio
- Il totale imponibile
- L'Iva
- La data di approvazione
- La data di inizio
- La data di scadenza

Nota : i campi riguardanti l'Iva non sono presenti nel caso del contributo.

(b) Inserimento della tabella di ripartizione

Questa scheda contiene le voci della tabella di ripartizione. L'Operatore può modificare alcuni valori percentuali in base ai quali dividere l'importo totale. Le voci non modificabili sono calcolate in relazione ai campi modificabili facendo riferimento alle norme di ateneo. Le voci modificabili sono:

- Personale: stabilisce la quota destinata al personale; è l'unico campo principale che l'operatore può modificare e in base al quale vengono calcolati gli altri.
- Missioni, Materiale di consumo, etc. sono sottocampi di Beni e Servizi e servono per meglio specificare come verrà ripartita la quota destinata a "Beni e Servizi".

(c) Gestione delle rate

Questo passo della procedura è facoltativo: viene data la possibilità all'operatore di inserire delle rate per la convenzione/contributo che sta creando. Una volta inserite una o più rate queste vengono visualizzate in una tabella e l'Operatore ha la possibilità di modificare, visualizzare, eliminare una rata cliccando sui tasti "Modifica","Visualizza" ed "Elimina" che compaiono sulla destra, nella riga della tabella corrispondente alla rata in questione. Per i dettagli riquardo alle operazioni sulle rate si rimanda ai corrispettivi casi d'uso.

(d) Inserimento della documentazione relativa alla convenzione

Questa scheda elenca i documenti allegati alla convenzione. L'Operatore può aggiungere o eliminare un documento cliccando sugli appositi tasti. Premendo il tasto "Salva" la convenzione viene salvata e la procedura termina. Si ritorna alla schermata precedente.

Percorso alternativo: Durante uno qualsiasi dei passi, l'Operatore può cliccare il tasto "Annulla", che comporta, a seguito di una conferma, il ritorno alla schermata precedente senza che la convenzione/contributo venga inserita o i cambiamenti effettuati salvati. Se l'Operatore clicca sul tasto "Salva" senza aver compilato dei campi obbligatori, o avendo inserito dei valori non consentiti, viene visualizzato un messaggio di errore e il documento non viene salvato. La schermata non viene cambiata, in modo che l'Operatore possa procedere alla correzione.

2. Visualizzazione delle convenzioni/contributi

Percorso base: l'operatore clicca su "Visualizza contratti"; viene mostrata una lista delle convenzioni/contributi correntemente stipulate in riferimento al dipartimento di afferenza dell'Operatore, con opportuni filtri per agevolare la ricerca (tra cui un filtro per data di scadenza) e ordinabili secondo la data. L'Operatore può selezionare una convenzione e compiere 3 azioni, per cui si rimanda ai corrispettivi casi d'uso, mediante gli appositi pulsanti, che appaiono sulla destra portando il cursore su una convenzione:

- visualizzarla
- modificarla
- eliminarla

Percorso alternativo: Si può tornare alla pagina iniziale con l'apposito tasto.

3. Modifica di una convenzione

   Percorso base: l'Operatore a partire dalla schermata "Visualizza Contratti" clicca sul pulsante "Modifica" che appare sulla destra nella riga della tabella corrispondente alla convenzione/contributo desiderata. Viene visualizzata una schermata suddivisa analoga a quella descritta nel caso della "Creazione di una convenzione/contributo" con la differenza che in questo caso è possibile spostarsi da una scheda all'altra cliccando sulla scheda stessa. Inoltre è presente la scheda aggiuntiva "Riepilogo" che contiene alcune informazioni di riepilogo come il residuo totale della convenzione/contributo o il totale fatturato. L'Operatore effettua i cambiamenti desiderati quindi clicca sul pulsante "Salva".

   Percorso alternativo: l'Operatore può cliccare sul tasto Annulla in qualsiasi momento per tornare alla schermata Visualizzazione delle convenzioni senza salvare le modifiche effettuate. Se l'Operatore clicca su Salva ma alcuni valori immessi non sono corretti, viene visualizzato un messaggio di errore e non si torna alla schermata Visualizzazione delle convenzioni. I cambiamenti effettuati non vengono (ovviamente) salvati.

4. Visualizzazione di una convenzione/contributo

   Del tutto analogo a "Modifica di una convenzione/contributo" con la differenza che in questo caso non è possibile modificare i dati della convenzione/contributo.

5. Inserimento di una rata
   Percorso base: l'operatore può inserire una rata sia in fase di creazione della convenzione/contributo sia in fase di modifica; in entrambi i casi dopo aver raggiunto la scheda "Rate" l'Operatore clicca sul pulsante "Aggiungi una rata". Viene visualizzata una finestra di dialogo suddivisa in varie schede, ognuna corrispondente ad un passo della procedura. E' possibile passare da una scheda all'altra mediante i pulsanti Avanti e Indietro. I passi sono:

   (a) Inserimento dei dati della rata

       L'operatore inserisce i seguenti campi

       - Importo
       - Iva
       - Data
       - Numero Reversale
       - Data Reversale
       - Numero di Sospeso
       - Numero di fatturato
       - Data fatturata
       - E'stata pagata la fattura?
       - Deve essere allegata la fattura?

- Note

Nota : i campi riguardanti l'Iva non sono presenti nel caso del contributo.

(b) Inserimento della tabella di ripartizione

Il procedimento è del tutto analogo a quello descritto nel caso d'uso Inserimento della tabella di ripartizione per l'inserimento di una nuova convenzione/contributo.

Le modifiche vengono salvate cliccando sul pulsante Salva. Si ritorna alla schermata precedente;

Percorso alternativo: l'Operatore clicca sul pulsante "Annulla", viene chiusa la finestra di dialogo senza che la rata sia stata inserita.

6. Modifica di una rata

Percorso base: l'Operatore accede alla schermata "Visualizza contratti" cliccando sul pulsante apposito nella pagina iniziale, quindi seleziona la convenzione/contributo alla quale la rata appartiene e clicca sul pulsante "Modifica" che compare all'interno della riga selezionata. Viene così visualizzata la schermata "Modifica di una convenzione/contributo"; l'Operatore raggiunge la scheda "rate" e clicca sul pulsante "Modifica" che compare selezionando la riga della tabella corrispondente alla rata desiderata. Viene visualizzata una finestra di dialogo composta di varie schede analoghe a quelle descritte nel caso dell'inserimento. E' possibile passare da una scheda all'altra cliccando su di esse. Dopo avere effettuato le modifiche richieste l'Operatore clicca sul pulsante "Salva", la convenzione/contributo viene aggiornata e viene visualizzata la schermata precedente.

Percorso alternativo: l'Operatore clicca sul pulsante "Annulla", le modifiche vengono scartate e si torna alla schermata precedente.

7. Visualizzazione di una rata

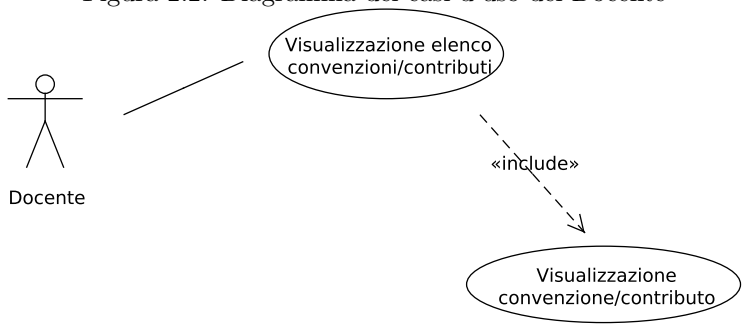
Una volta raggiunta la scheda "rate" relativa alla convenzione/contributo di interesse l'Operatore, dopo aver selezionato la rata desiderata, clicca sul pulsante "Visualizza". Viene presentata una finestra di dialogo analoga a quella descritta nel caso della modifica con la differenza che i campi non sono modificabili. E' possibile tornare alla schermata precedente cliccando sul pulsante "Indietro".

8. Eliminazione di una rata

Una volta raggiunta la scheda "rate" relativa alla convenzione/contributo di interesse l'Operatore, dopo aver selezionato la rata di interesse, clicca sul pulsante "Elimina'; appare una finestra di dialogo che chiede di confermare l'eliminazione, l'Operatore clicca "Sì", la rata viene eliminata.

Docente I casi d'uso del Docente sono rappresentati in figura 2.2

Figura 2.2: Diagramma dei casi d'uso del Docente



1. Visualizzazione dell'elenco delle convenzioni/contributi

   Il docente, dopo aver effettuato il login, può cliccare sul pulsante "Visualizzazione della lista delle convenzioni/contributi"; la schermata che viene visualizzata contiene una tabella che elenca le convenzioni/contributi del docente. E' possibile filtrare le convenzioni/contributi secondo vari criteri(data, tipo, scadenze più vicine, ...).Inoltre è possibile visualizzare i dettagli di una convenzione/contributo cliccando sul pulsante "Visualizza" che appare posizionando il puntatore su una riga della tabella.

2. Visualizzazione di una convenzione/contributo di cui il docente è responsabile scientifico

   Il docente dalla schermata "Lista delle convenzioni/contributi" può cliccare sul pulsante "Visualizza" relativo ad una convenzione/contributo; compare una schermata suddivisa in schede analoga a quella della modifica/creazione della convenzione. Il docente può navigare fra le schede cliccandoci sopra. Non è permessa nessuna modifica ai dati della convenzione/contributo tuttavia il docente può inserire degli allegati dalla scheda "Allegati". Cliccando su "Salva" gli allegati inseriti dal docente vengono memorizzati, al contrario cliccando su "Indietro" le modifiche vegono scartate.

Amministratore

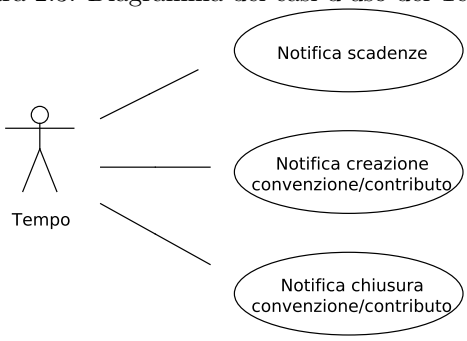
1. Inserimento di un nuovo utente
2. Visualizzazione della lista degli utenti

Tempo I casi d'uso del Tempo sono rappresentati in figura 2.2

1. Notifica delle scadenze

   Ad intervalli periodici stabiliti, i docenti che hanno convenzioni/contributi attive con rate in scadenza ravvicinata, vengono avvertiti tramite posta elettronica.

Figura 2.3: Diagramma dei casi d'uso del Tempo



2. Notifica della creazione di una nuova convenzione/contributo

   Al momento del completamento della creazione di una nuova convenzione/contributo viene inviata una email all'indirizzo di posta elettronico del responsabile scientifico indicato per la convenzione/contributo. Tale email contiene le informazioni principali che caratterizzano la convenzione/contributo.

3. Notifica della chiusura di una convenzione/contributo

   Al momento della chiusura di una convenzione/contributo (ovvero quando il fatturato è pari all'importo totale) viene inviata una email all'indirizzo di posta elettronico del responsabile scientifico che notifica la chiusura della convenzione/contributo riportando alcuni dati di questa.

Capitolo 3

Tecnologie

3.1 JPA

3.1.1 Introduzione

Le applicazioni di tipo Enterprise necessitano di raccogliere e persistere grandi quantità di informazioni, la soluzione che più si è affermata per risolvere questo problema è il database relazionale. Nell'ambiente Enterprise , in particolare per la piattaforma Java, si è sentita l'esigenza di soluzioni che permettessero l'integrazione della piattaforma Java con database di tipo relazionale, in modo naturale.

3.1.2 Mapping fra modello Relazione e a Oggetti

In un modello a oggetti,come il modello di dominio di un applicativo Java, le entità che popolano il modello sono Classi. Le relazioni fra le classi del modello sono espresse tramite riferimenti, gli attributi di una classe. Nel modello relazionale invece le entità sono chiamate Relazioni e sono collegate fra loro mediante il concetto di chiave. Effettuare un mapping fra i due modelli significa quindi avere un modo per trasferire i concetti da un modello all'altro in modo da ridurre le distanze fra i due modelli. In particolare una soluzione per il mapping dovrebbe avere le seguenti caratteristiche:

- le applicazioni dovrebbero essere scritte e pensate secondo il modello di dominio e senza avere legami col modello relazione del database; dovrebbe essere possibile recuperare informazioni dal database senza dover scrivere espressioni che coinvolgano tabelle o chiavi primarie tipiche di un database.
- la soluzione dovrebbe essere non intrusiva: sebbene sia impensabile di realizzare una soluzione che renda la persistenza completamente trasparente, si può richiedere che la persistenza non "invada" il modello di domionio. In concreto le classi del dominio non devono essere obbligate ad implementare interfacce o estendere classi per poter essere rese persistenti.
- dovrebbe essere possibile integrarsi con database preesistenti

3.1.3 JPA

Nel tempo sono state sviluppate e proposte varie soluzioni, a partire da ODBC passando per JDBC e EJB fino ad arrivare a *JPA*. JPA (Java Persistence API) è una specifica per il mapping fra modello a oggetti e modello relazionale per applicazioni Java che risolvendo i problemi degli standard precedenti raggiunge gli obiettivi sopra formulati. Esistono vari prodotti che implementano JPA fra cui *Hibernate* che è stato scelto come implementazione di riferimento.

Funzionamento

Entity Un' unità che possegga una stato e che possa essere persistita viene detta Entità. Le classi Java possono essere facilmente trasformate in entità semplicemente annotando la classe e alcuni dei suoi attributi, o alternativamente fornendo dei descrittori xml. L'unico requisito che la classe deve rispettare è che deve possedere un costruttore senza parametri: questo serve affinchè Hibernate (o qualsiasi altro provider) possa ricreare l'oggetto una volta interrogato il database. Come si può notare, non è possibile rendere persistibile una classe in modo del tutto trasparente, ma di sicuro si può affermare che questo avvenga in modo non intrusivo. Allo stesso modo, sempre usando delle annotazioni o dei file di descrizione xml, è possibile mappare le relazioni che sussistono fra le entità del modello.

Entity Manager e Peristence Context Le Entità sono gestite da un Entity Manager. Un entity Manager è in grado di persistere un entità nonchè di eliminarla o recuperarla dal database. Ogni Entity Manager è associato ad un Persistence Context. Un Persistence Context è un insieme di istanze di entità. Una entità si dice "managed" se è contenuta in un Persistence Context. Se un Persistence Context, tramite un Entity Manager, partecipa ad una transazione lo stato delle entità "managed" contenuto in memoria centrale viene salvato sul database. Le entità non "managed" sono chiamate "detached" e il loro stato in memoria non viene sincronizzato col database in nessuna transazione. Come è possibile ottenere un Entity Manager? Ci sono vari tipi di Entity Manager ognuno dei quali legato a diverse esigenze applicative. Si è scelto di includere nella presente trattazione solo una sottocategoria: le Entity Manager di tipo "Container-Managed".Questa scelta è dovuta al fatto che le Entity Manager "Container-Managed" sono quelle che sono state usate nella realizzazione dell'applicativo nonchè le tipologie preferite per l'ambiente Java EE. Un' Entity Manager di questo tipo è ottenuta dal container mediante iniezione di dipendenza. Le Entity Manager di tipo "Container-Managed" sono di due tipi:

- Transaction Scoped
  Una Entity Manager di questo tipo è *stateless*, ovvero lavora con un Persistence Context che viene costruito ogni volta che comincia una transazione (JTA) e termina il proprio ciclo di vita al termine della transazione

- Extended
  Una Entity Manager di tipo Extended è usata in coppia con un bean(vedi 3.2) di tipo *statefull*. In questo caso l'Entity Manager lavora con un singolo Persistence Context il cui ciclo di vita è legato a quello del bean che potenzialmente sopravvive a più di una transazione.

L'uso del corretto tipo di Entity Manager dipende dal contesto: per esempio sarà conveniente usare un'Entity Manager di tipo transaction-scoped nel caso dell'eliminazioni di un entità dal database mentre è forse più conveniente usare un'Entity Manager di tipo extended per recuperare dal database un oggetto il cui stato viene modificato più volte e salvato solo alla fine della sessione corrente.

Query Il linguaggio in cui sono espresse le query è chiamato JPQL. Questo linguaggio ha due caratteristiche principali che vanno nella direzione indicata negli obiettivi presentati:

- Il linguaggio è indipendente dal database sottostante. Questo significa che l'applicazione non è dipendente dal particolare database usato ma al contrario è possibile con facilità migrare da una soluzione all'altra senza dover cambiare il codice.
- Sebbene JPQL sia un linguaggio dichiarativo che rassomiglia molto da vicono SQL non usa tabelle e colonne per esprimere i propri criteri di ricerca ma usa le entità del modello di dominio e i loro attributi.

3.2 CDI

3.3 JSF

3.4 Deltaspike

Deltaspike è un *framework* che estende le funzionalità di *CDI*. Il suo utilizzo in Jama è la gestione della sicurezza: associare ad ogni utente un insieme di permessi che determinano le azioni che può effettuare. La caratteristica fondamentale di Deltaspike è l'essere un *framework annotation-based*: funziona tramite le *Annotation* di Java - che d'ora in poi verranno chiamate semplicemente annotazioni - il che lo rende facile da utilizzare e poco invasivo.

3.4.1 Caratteristiche Generali

Il cuore della sicurezza in Deltaspike è la classe *Authorizer*: è un *bean* di CDI - tipicamente *Application Scoped* - definito dallo sviluppatore che definisce i metodi che verranno invocati durante il controllo sui permessi di un utente. Questi metodi devono essere annotati con l'annotazione *@Secures* definita da Deltaspike, che indica che essi sono i metodi da invocare per il controllo sui permessi. Non basta: devono essere annotati anche con un'altra annotazione, una *custom annotation* - cioè definita dallo sviluppatore - che associa il metodo in questione ai metodi su cui effettuare il controllo dei permessi. Quest'ultima deve essere a sua volta annotata con l'annotazione *@SecurityBindingType* di Deltaspike.
Nonostante già il controllo sui metodi possa essere sufficiente per gestire l'intera sicurezza di un'applicazione, una funzionalità interessante e molto utile di Deltaspike è la possibilità di effettuare un controllo a livello di pagina: ad esempio, far sì che solo un Amministratore possa visitare la pagina di creazione di un utente, o che solo un Operatore Amministrativo possa visitare la pagina di creazione di una convenzione. Inoltre, qualora il controllo non vada a buon fine,

si può specificare una pagina di errore diversa per ogni pagina, ed un messaggio di errore che verrà visualizzato a video dopo il redirect.
Per concludere quest'introduzione e visione d'insieme delle funzionalità di Deltaspike con gli autori hanno avuto a che fare, si ritiene comunque importante far notare che, nonostante sia un framework già usabile e utile, non sia ancora completamente maturo: la documentazione è scarsa - per non dire di peggio - e alcune funzionalità utili sono mancanti o non funzionanti - ad esempio, il redirect ad una pagina di errore anche per la sicurezza sui metodi.
Si discute adesso in dettaglio come utilizzare Deltaspike.

3.4.2 Rendere sicuro un metodo

Per spiegare come si rende sicuro un metodo, si prenderà come esempio un problema affrontato nello sviluppo di Jama: far sì che solo un Operatore Amministrativo possa eliminare una convenzione. L'eliminazione di una convenzione, in Jama, consiste in sostanza nell'invocazione di un metodo di uno dei *bean* dell'applicazione, quindi il problema si risolve impedendo l'invocazione di tale metodo da parte di utenti che non siano un Operatore Amministrativo.
Il primo passo è il definire un'annotazione, chiamata per esempio *@DeleteContractsAllowed*:

```
Retention(value = RetentionPolicy.RUNTIME)
@Target({ ElementType.TYPE, ElementType.METHOD })
@Documented
@SecurityBindingType
public @interface DeleteContractsAllowed {}
```

Si procede dunque ad annotare il nostro metodo che elimina una convenzione con l'annotazione appena definita:

```
...

@DeleteContractsAllowed
public void deleteContract() {
	//Elimina una convenzione.

	...
}
```

L'ultima cosa da fare è fornire l'Authorizer di un metodo annotato *@Secures* e *@DeleteContractsAllowed*:

```
public class Authorizer {
	@Secures
	@DeleteContractsAllowed
	public boolean canDeleteContracts {
		//Verifica che l'utente possa eliminare una convenzione.
		//Restituisce true in caso affermativo, altrimenti false.

		...
	}
```

```
}
```

Il metodo appena definito verrà invocato da Deltaspike in maniera automatica tutte le volte che si invocherà *deleteContract()*. Nel caso *canDeleteContracts()* restituisca *true*, l'eliminazione può proseguire, altrimenti viene generata un'eccezione.

3.4.3 Rendere sicura una pagina

La sicurezza su base pagina si implementa definendo interfacce e classi che rappresentano rispettivamente cartelle e pagine. Ad esempio, per rendere sicura la pagina *home.xhtml* che si trova dentro la cartella *pages*, si definisce un interaccia *Pages* con all'interno una classe *Home*; le classi così definite devono implementare l'interfaccia *ViewConfig* di Deltaspike. Il *path* è relativo alla cartella *webapp* dell'applicazione, quindi la classe Home contenuta nell'interaccia Pages fa riferimento alla pagina *webapp/pages/home.xhtml*. I nomi sono *case insensitive*: la classe Home fa riferimento alle pagine home.xhtml e Home.xhtml; stesso discorso vale per le interfacce.
Il metodo più facile per rendere sicuro un insieme di più pagine è creare un file Java e di definire all'interno di quest'ultimo tutta la gerarchia di interfacce e di classi - ricordando di omettere il *modifier public* per ognuna delle interfacce/-classi, altrimenti si otterrebbe un errore di compilazione.
Ognuna delle classi definite deve essere annotata con l'annotazione *@Secured*, completata con l'attributo *value* che specifica una classe definita dallo sviluppatore che implementa l'interfaccia *AccessDecisionVoter* di Deltaspike; questa classe deve essere un *bean* di CDI, tipicamente *Application Scoped* . La classe specificata deve implementare il metodo *public Set<SecurityViolation> checkPermission()*, che viene invocato da Deltaspike ogni volta che si tenta di accedere alla pagina.
Il metodo restituisce un Set di *SecurityViolation*, una *Anonymous Inner Class* definita da Deltaspike. Nel caso esso sia vuoto, l'accesso viene consentito, altrimenti viene impedito.
Come abbiamo visto nell'introduzione, una delle peculiarità più interessanti della sicurezza su base pagina è il poter specificare una pagina di errore. Per fare ciò, si arricchisce l'annotazione *@Secured* delle pagine con l'attributo *errorView*, indicando una pagina definita secondo la solita convenzione spiegata ad inizio paragrafo. Nel caso venga specificata una pagina di errore, le SecurityViolation contenute nel Set restituito da checkPermission() vengono aggiunte al *bundle* di messaggi della pagina, ovvero all'interno dell'area definita dal tag *<h:messages>* della pagina - detta in maniera meno tecnica, verrà visualizzato un messaggio a video nella pagina di errore per ogni SecurityViolation contenuta.
Il messaggio da visualizzare viene specificato nel metodo *public String getReason()* di ogni SecurityViolation; ricordando che quest'ultima è una Anonymous Inner Class, il metodo getReason() viene obbligatoriamente ridefinito ad ogni SecurityViolation creata, permettendo così di specificare il messaggio di errore più adatto in ogni occasione.

Per fissare meglio le idee, è opportuno mostrare un esempio. Per prima cosa, si creano le classi relative alle pagine da rendere sicure:

```
interface Pages {

        @Secured(value = { ViewHomeAccessDecisionVoter.class },
            errorView = Login.class)
        class Home implements ViewConfig {}


        class Login implements ViewConfig {}

}
```

Successivamente si definisce l'AccessDecisionVoter; non viene mostrata nel dettaglio nella logica che esegue il controllo, verrà invece usato uno pseudo-codice che renda l'idea di come deve funzionare la classe:

```
@ApplicationScoped
public class ViewHomeAccessDecisionVoter implements AccessDecisionVoter {
        private SecurityViolation violation = new SecurityViolation() {
                public String getReason() {
                        return "Non sei autorizzato";
                };

        public Set<SecurityViolation> checkPermission(
                                AccessDecisionVoterContext
                                    accessDecisionVoterContext) {

                Set<SecurityViolation> violations = new HashSet<>();

                if( user cannot see home )
                        violations.add(violation);
                return violations;
        }
}
```

Il gioco è fatto: chiunque tenti di visitare la pagina *home* senza averne i permessi - ad esempio, un utente che senza fare il login tenta di andare subito alla home - viene reindirizzato alla pagina di login, dove potrà vedere un bel messaggio che recita: *Non sei autorizzato!*

Capitolo 4

Utilizzo