TESI DI LAUREA TRIENNALE IN INFORMATICA:
ResMan, un accesso unificato ai documenti elettronici.

Valeri Beatrice

18 agosto 2009

Indice

1 Introduzione **3**
- 1.1 Liquidpub e Resman . . . 3
- 1.2 Scopo . . . 4
- 1.3 Approccio . . . 5
- 1.4 Struttura del documento . . . 5

2 Un esempio per comprendere **6**

3 State of the Art **7**
- 3.1 Dataspaces . . . 7
  - 3.1.1 Come ottenere delle informazioni . . . 8
  - 3.1.2 I servizi forniti . . . 9
  - 3.1.3 L'importanza dell'uomo . . . 10
- 3.2 Process Spaces . . . 10
- 3.3 Lifecycle . . . 10
- 3.4 Directory Services . . . 11
- 3.5 Dinamic binding e Adapters . . . 11
- 3.6 Message Brokers . . . 11

4 Resource Management **12**
- 4.1 Esecuzione delle azioni . . . 12
- 4.2 ResmanProxy . . . 12

5 Adapter e azioni **13**

6 Tecnologie utilizzate **14**
- 6.1 Rest . . . 14
- 6.2 Hibernate . . . 14
- 6.3 Java MVC . . . 14

7 Architettura e Implementazione **15**
- 7.1 Livello Dao . . . 15
- 7.2 Livello Model . . . 15
- 7.3 Livello Service . . . 15

7.4 Livello Rest 15
7.5 Livello Controller 15
7.6 Proxy 15
7.7 Plugins 15

8 Conclusione **16**

Capitolo 1

Introduzione

1.1 Liquidpub e Resman

Liquidpub è un progetto europeo per la pubblicazione liquida di documenti scientifici. Mira alla creazione di un ambiente in cui creare, diffondere, valutare e consultare documenti scientifici.

In questo ambiente gli autori di un documento possono collaborare in modo pi'u organizzato che tramite email, come accade tutt'ora. Ad essi sono dedicati molti servizi. Possono gestire la vita del loro documento tramite l'uso di lifecycles, stabilendo quali azioni dovranno svolgere e anche le date di scadenza di ogni parte del lavoro. Potranno importare un documento già iniziato grazie al collegamento unificato coi maggiori servizi di gestione di documenti, come ad esempio Google Documents. Inoltre è fornito anche l'accesso a servizi dedicati alla gestione di progetti collaborativi, come Subversion, permettendo di mantenere un'unica versione aggiornata del progetto.

L'aspetto più importante di questa piattaforma è che fornisce diretto contatto tra autori e reviewers, velocizzando quindi il processo di analisi e correzione dei documenti per renderli pubblicabili. Viene velocizzata anche la diffusione e semplificata la consultazione dei documenti. Infatti una volta accettati dai reviewers, gli articoli vengono resi accessibili in lettura da chiunque all'interno della piattaforma e vengono divulgati tramite connessioni con servizi appositi, come per esempio i blog.

È inoltre presente un sistema di valutazione di documenti ed autori tramite indici basati sulla quantità e qualità delle pubblicazioni e sulla quantità dei riferimenti che vengono fatti a un tale articolo. Insieme a tali indici vengono calcolate anche numerose statistiche che risultano utili per comprendere la qualità degli articoli.

Resman è un progetto che si occupa della gestione delle risorse all'interno di Liquidpub. Fin dall'inizio si è posto il problema di gestire i vari tipi di risorse e le azioni disponibili su di esse in modo semplice e trasparente. Per

fare ciò, si è pensato alla creazione di una piattaforma che fornisca un'unica interfaccia per accedere a qualunque documento e per interagire con esso. Tale piattaforma, infatti, è molto utile per semplificare lo sviluppo degli altri moduli di Liquidpub al di sopra di essa, ma può anche essere utilizzata come servizio web indipendente da altre applicazioni web che intendono accedere alle risorse scientifiche.

Resman è quindi un *Resource Space Management System* (RSMS) che fornisce un accesso programmatico ed omogeneo alle risorse scientifiche astranedo le operazioni di cui il sistema può avere bisogno, astraendo i vari tipi di documenti in un unico modello concettuale e nascondendo il problema di comunicare con diverse piattaforme. Questo servizio permette agli utenti di accedere alle risorse e di eseguire azioni su di esse senza preoccuparsi del tipo di documento che esse sono. Esso si preoccupa di capire a quale servizio fà riferimento la risorsa in qestione e a inviare in modo corretto i dati per l'esecuzione dell'azione richiesta da parte del servizio opportuno.

Questa piattaforma è stata ideata per essere man mano estesa a seconda delle necessità da parte di tutti. L'obiettivo finale è infatti quello di fornire un servizio universale, che permetta l'accesso a tutte le risorse scientifiche disponibili nel web senza esclusioni. Ovviamente per fare ciò è necessario permettere che nuovi moduli vengano aggiunti ogni volta che si scoprono nuove fonti di documenti in modo da gestire anche tali documenti in modo corretto. Questo servizio deve anche risultare semplice e flessibile in quanto chiunque deve essere in grado di usarlo al meglio. La maggior parte degli utenti ha solo la capacità di usare fogli di calcolo o wikis e noi dobbiamo fare in modo che non abbiano difficoltà ad utilizzare questo servizio.

1.2 Scopo

Lo scopo di Liquidpub è la creazione di un'ambiente migliore adatto allo sviluppo e alla diffusione di documenti scientifici. Tale ambiente deve sostituire l'attuale ciclo di vita dei documenti che spesso è lungo e faticoso per gli scrittori di articoli scientifici. Questa piattaforma deve rendere più veloce e semplice la pubblicazione e diffusione del sapere scientifico. Infatti bisogna tenere conto del fatto che il sapere scientifico di oggi evolve con ritmi frenetici, quindi se, dal momento del suo completamento, un documento impiega sei mesi per essere pubblicato, il suo contenuto risulta ormai vecchio rispetto alle ricerche sviluppate in tale lasso di tempo. Quindi diventa importante ridurre il tempo necessario per la pubblicazione di documenti, in modo che il loro contenuto sia all'avanguardia al momento della loro diffusione.

Lo scopo di Resman è di fornire un servizio per la gestione di risorse scientifiche.
Continua...

1.3 Approccio

1.4 Struttura del documento

Capitolo 2

Un esempio per comprendere

Contiene un'esempio ad hoc per poter spiegare bene in particolare l'argomento approfondito.

Capitolo 3

State of the Art

3.1 Dataspaces

I *dataspaces* sono stati ideati per risolvere il problema della gestione di una grande quantità di dati separati in più fonti diverse. Al giorno d'oggi capita spesso che un'applicazione si appoggi sui dati presenti in diversi database forniti e gestiti da diversi servizi e non ideati per essere usati in correlazione. Quindi, sebbene i dati contenuti siano semanticamente legati tra loro, per un'applicazione potrebbe essere molto complicato notare tali correlazioni perchè, per esempio, due campi, su due database diversi che contengono effettivamente gli stessi dati, hanno nomi diversi poichè sono stati creati da due persone diverse.

Sono già stati sviluppati fino ad ora dei sistemi di integrazione dei dati per fornire servizi su grandi quantità di dati, ma per utilizzare tali tecniche in modo efficiente bisogna prima ottenere una piena integrazione semantica delle fonti di dati. Per fornire servizi sui *dataspace* sono state ideate le *DataSpace Support Platforms* (DSSPs, Piattaforme di Supporto ai Dataspace), che rappresentano un passo avanti rispetto ai precedenti sistemi di integrazione dei dati in quanto non necessitano dell'integrazione semantica delle fonti di dati. Un DSSP fornisce una serie di servizi e rende gli sviluppatori in grado di concentrarsi sulla propria applicazione anzichè preoccuparsi della gestione dei dati a basso livello per fornire dei servizi utili a utenti e amministratori. Un DSSP aiuta a identificare le fonti in un *dataspace* e a riconoscere i legami tra di esse, offre un meccanismo per porre domande e di introspezione sui dati ottenuti da esse. Inoltre fornisce anche meccanismi per rinforzare i "constraint" e alcune nozioni di consistenza e recovery.

I *dataspaces* usano un apporccio *data co-existence* (coesistenza dei dati). Lo scopo di un *dataspace* è di fornire le funzionalità base su tutti i dati coinvolti, senza badare a quanto le risorse siano integrate. Inoltre se è richiesto un servizio più accurato o un'operazione più sofisticata, viene applicato uno sforzo di calcolo addizionale per integrare maggiormente le fonti interessate,

secondo una tattica "pay-as-you-go". In tale modo il lavoro per l'integazione dei dati viene rimandato finchè non risulta necessario.

3.1.1 Come ottenere delle informazioni

Un *dataspace* è costituito da due tipi di elementi:

- ***participants* (partecipanti)**: sono le risorse dei dati, ognuna delle quali ha caratterisiche diverse come, per esempio, un diverso linguaggio supportato o diversi gradi di mutabilità;
- ***relationships* (relazioni)**: sono le relazioni tra due o più partecipanti al sistema, siano esse molto strette, come per esempio un mapping tra due partecipanti, o molto meno specifiche, come per esempio che due fonti sono ottenute entrambe da una terza.

Poichè i database che costituiscono un *dataspace* possono necessitare di diversi linguiaggi per porre le query, la DSSP deve essere in grado di tradurre le richieste in ogni linguaggio necessario in modo da poter fornire una risposta completa. Quando un utente pone una domanda nei termini di quache schema di una particolare fonte, il DSSP deve in ogni caso cercare la risposta in tutto il *dataspace*, perchè l'utente si aspetta comunque di ricevere le risposte anche dalle altre fonti.

Le query poste ad un *dataspace* devono avere caratteristiche diverse da quelle poste ad un database tradizionale:

- **classificazione**: la classificazione delle risposte è necessaria in particolare quando le traduzioni per le altre risorse di dati possono essere approssimative;
- **eterogeneità**: le risposte giungono da molte fonti e si troveranno in diversi schemi e modelli di dati, quindi la classificazione deve tenere conto dell'eterogeneità degli elementi che deve ordinare;
- **indicazione della fonte per ogni risposta**: insieme alle risposte, il DSSP deve restituire per ogni risposta anche un puntatore al posto in cui è stata trovata e dove si possono trovare ulteriori informazioni.

Quindi abbiamo bisogno di un nuovo modello formale per definire domande e risposte. Bisogna ancora creare un modello per ottenere le risposte da un *dataspace*, anche quando le domande richiedono di prendere in considerazione una sequenza di domande poste precedentemente. Bisogna creare algoritmi che classifichino le fonti dei dati in base alla probablilità che hanno di fornire la risposta migliore e metodi che classifichino le risposte ottenute da fonti eterogenee. Inoltre c'è la necessità di creare comandi che includano più operazioni di basso livello su un *dataspace*.

Poichè le fonti di dati sono eterogenee sia al livello dello schema che al livello dei dati, c'è la necessità di sviluppare dei metodi per rispondere alle domande che non dipenda unicamente dall'applicare un insieme di mappature semantiche corrette. Bisogna appoggiarsi alle seguenti idee combinandole quando necessario:

- usare diverse mappature approssimate o incerte e paragonare le risposte ottenute;
- usare domande per parole chiave per ottenere dati o costanti che permettano di affinare le mappature esistenti o di crearne di nuove;
- esaminare le domande e le risposte precedenti per dedurre le relazioni che ci sono tra le varie fonti di dati, in particolare quando le domande attraversano numerose risorse di dati.

Inoltre, ci serve un modello formale per creare mappature approssimate e per misurare l'accuratezza delle risposte ottenute. Per la natura eterogenea delle fonti di dati può capitare che due insiemi di dati contengano gli stessi termini, per esempio come nomi di tabelle, ma dati diversi. Quindi è necessario sviluppare dei metodi per tradurre al meglio le domande da un insieme di dati all'altro.

3.1.2 I servizi forniti

Uno dei servizi fondamentali di un *dataspace* è la catalogazione degli elementi dei dati forniti dai partecipanti. Nel catalogo sono presenti tutte le risorse, associate alle proprie informazioni basilari, come per esempio la fonte che le fornisce, il nome, la data di creazione e così via. Il catalogo è principalmente una struttura di supporto per gli altri servizi forniti dal dataspace, ma può anche supportare un'interfaccia che permette agli utenti di effetuare ricerche sul *dataspace*.

Altri servizi molto importanti che un DSSP deve supportare sono la ricerca e l'interrogazione. Un DSSP deve permettere ad un utente di effettuare una query al dataspace e successivamente rifinirla in modo da ottenere risposte più interessanti e meno vaghe. Il punto fondamentale nella ricerca in un *dataspace* è che essa deve permettere di ottenere informazioni da tutte le fonti, nonostante l'eterogeneità delle loro strutture e dei loro contenuti.
Questi servizi devono essere disponibili anche sui meta-dati, oltre che sui dati. Gli utenti devono essere in grado di effettuare ricerche riguardo alle fonti dei dati e alle loro caratteristiche principali. Infatti è utile sapere quanto spesso viene aggiornata una fonte, se i suoi contenuti sono completi e corretti per poter valutare l'affidabilità delle informazioni che essa fornisce.

Un DSSP deve anche fornire un supporto per l'aggiornamento dei dati, sempre rispettando i vincoli sulla modifica dei dati imposti dalle fonti.
Altri servizi utili sono il monitoring, l'uso di tecniche per tenere traccia degli

eventi e un supporto per i workflows complessi.
Può anche essere necessario estendere le fonti di dati, per esempio nel caso in cui esse non tengano traccia dei meta-dati dei propri documenti. In tal caso è necessario usare un deposito indipendente per tali meta-dati.

3.1.3 L'importanza dell'uomo

Il modo migliore per raccogliere informazioni semantiche sul *dataspace* è controllando l'attività degli utenti. Uno dei prossimi passi avanti della ricerca consiste nel sviluppare metodi che tengano traccia delle attività svolte dagli utenti per analizzarle e creare nuove relazioni tra le risorse del *dataspace*. Bisogna inoltre analizzare le sequelze di domande poste e le risposte ottenute per creare nuove mappature attraverso le fonti di dati. Un modo migliore per ottenere informazioni semantiche sui dati nel *dataspace* consiste nel creare, analizzando le relazioni conosciute, delle domande da porre direttamente all'utente.

3.2 Process Spaces

3.3 Lifecycle

Le persone che collaborano nella creazione di un documento hanno necessità di coordinare il proprio lavoro e di sapere quali azioni dovranno essere eseguite per iniziare, migliorare e completare tale oggetto. Il *lifecycle* non è altro che uno schema che indica le varie azioni a cui un documento dovrà essere sottoposto per essere completato. I vari gruppi che ci lavorano possono avere diversi ruoli e diversi livelli di visibilità e accesso. Attualmente l'organizzazione nei lavori di gruppo è un pò carente in quanto i cicli di vita non vengono definiti formalmente e tipicamente ci si coordina tramite email, rendendo quindi difficile la visione d'insieme del lavoro in un determinato stato.

C'è quindi bisogno di uno strumento che aiuti nella gestione dei lifecycle. Esso deve permettere di definire il lifecycle dei file trattati in un progetto, assegnare tali lifecycle a risorse e persone tenendne presente i diversi ruoli, controllare lo stato dei lifecycle. Le persone sono molto coinvolte nel processo di vita dei documenti a cui lavorano, quindi il passaggio da uno stato al un altro, l'avvio dell'esecuzione di un'azione, in modo assistito dal tool di gestione, e la modifica del lifecycle per adattarlo alle reali necessità della risorsa a cui esso è legato sono azioni che devono essere eseguite manualmente da un utente che ne ha il permesso. Tutte queste funzioni devono essere fornite in modo che il lifecycle sia il più indipendente possibile dalla risorsa a cui sarà legato, in modo da poter essere poi "riciclato" per altre risorse simili. Il lifecycle deve essere anche flessibile e robusto. Deve essere permessa sempre

la modifica del modello per essere aggiornato e adattato alle reali necessità, che possono anche cambiare nel corso della vita del documento a cui esso si riferisce. Un lifecycle può essere usato per gestire la vita di un documento anche se è incompleto. Inoltre il tool di supporto deve essere molto sempice e intuitivo da usare in quanto chiunque dovrà essere in grado di usarlo senza distogliere troppa attenzione dal documento a cui deve lavorare.

Un lifecycle è costituito da un nodo d'inizio e uno di fine, dalle fasi che contengono le azioni che vanno eseguite in un determinato momento e che sono connesse da freccie direzionali. Tali frecce indicano le possibili transazioni di fase, ovvero le possibili evoluzioni del sistema. Inoltre ad ogni lifecycle è associato il ruolo che l'utente deve avere per poter eseguire le azioni contenute nelle fasi.

3.4 Directory Services

3.5 Dinamic binding e Adapters

3.6 Message Brokers

Capitolo 4

Resource Management

Spiegazione di Resman più approfondita. Descrizione degli oggetti coinvolti e delle varie interfacce.

4.1 Esecuzione delle azioni

4.2 ResmanProxy

Capitolo 5

Adapter e azioni

Spiegazione dettagliata del funzionamento di Adapters ed Actions

Capitolo 6

Tecnologie utilizzate

Approfondimento teorico sulle tecnologie utilizzate per l'argomento affrontato dettagliatamente.

6.1 Rest

6.2 Hibernate

6.3 Java MVC

Capitolo 7

Architettura e Implementazione

Spiegazione del progetto a livello codice.

7.1 Livello Dao

7.2 Livello Model

7.3 Livello Service

7.4 Livello Rest

7.5 Livello Controller

7.6 Proxy

7.7 Plugins

Capitolo 8

Conclusione

Risultati ottenuti e ringraziamenti.