Alma Mater Studiorum · Università di Bologna

**SCUOLA DI SCIENZE**
**Corso di Laurea in Informatica per il management**

# MONITORING DI UN PORTALE WEB: MODELLO E IMPLEMENTAZIONE

**Relatore:**
**Dott.**
**ANGELO DI IORIO**

**Presentata da:**
**CECILIA FALCHI**

**Sessione II**
**Anno Accademico 2013/2014**

# Introduzione

Nella società dell'informazione, ogni individuo e organizzazione ha a disposizione grandi quantità di dati e informazioni relative a se stessa e all'ambiente nel quale si trova ad operare: anche l'espansione del World Wide Web ha avuto come risultato la produzione di una quantità enorme di dati e informazione che è generalmente disponibile e libera all'accesso da parte degli utenti.

Ogni giorno enormi quantità di dati sono prodotti come record dettagliati del comportamento di utilizzo del Web, ma l'obiettivo di trarne conoscenza rimane ancora una sfida. Data la quantità e l'eterogeneità dei dati, si è sempre posto il problema di estrarre ed evidenziare informazioni utili per fini commerciali ed economici, per migliorare la struttura dei siti, per ricavarne statistiche, per la personalizzazione dei siti o per classificare i contenuti e gli utenti del web.
Interessanti sono quindi gli studi del *Web usage mining*, che si pone come obiettivo quello di catturare, modellare e analizzare i pattern comportamentali e i profili degli utenti che interagiscono con un sito Web. Il *Web mining*, applicazione del *data mining* nel Web, propone soluzioni a questo problema, le quali mirano a scoprire pattern ed estrarre conoscenza dai dati salvati sul Web. I pattern rilevati sono di solito rappresentati come una collezione di pagine o risorse a cui accedono frequentemente gruppi di utenti, con interessi e necessità in comune.
Il Web usage mining [26] si occupa dell'estrazione di conoscenza a partire

dai file di log di navigazione e sono basate sulla raccolta dei dati forniti da tre fonti principali: web server, proxy server e web client. Dal momento che possono raccogliere una grande quantità di informazioni nei loro file di log, i Web server rappresentano sicuramente la più ricca e la più comune fonte di dati, ed è quella su cui si basano la maggior parte degli algoritmi di *mining* presenti nella letteratura. Ogni accesso a una pagina web viene registrati dei log di accessi del web server che la ospita. Le informazioni fornite dai server log possono essere usate per identificare e costruire diversi livelli di astrazione dei dati [27]: utenti, sessioni server, *pageview* (visualizzazione della pagina sul browser), *clickstream* (serie sequenziale di richieste di *pageview*).

Dagli anni '90 ai giorni nostri si è visto un grande aumento della letteratura riguardante la modellazione del comportamento di navigazione e molte sono le ricerche fatte in questo campo. Diversi tipi di attività di*mining* possono essere applicati nel dominio del Web e consentono di ricavare la maggiore quantità di dati possibile: regole di associazione, classificazione, *clustering*, pattern sequenziali e modelli di dipendenza. Le regole di associazione sono usate di solito per mettere in relazione pagine che compaiono frequentemente insieme nella stessa sessione; il *clustering* è la tecninca che permette di individuare gruppi, nel caso del *Web usage mining* di utenti e di pagine Web, i cui componenti hanno caratteristiche simili. La tecnica della scoperta di pattern sequenziali mira a scoprire pattern di oggetti in ordine temporale all'interno delle sessioni, con lo scopo ad esempio di predire la navigazione di un utente. L'obiettivo dei modelli di dipendenza invece è di sviluppare modelli in grado di rappresentare dipendenze significative fra le variabili (utenti e pagine Web) del dominio di Web.
Queste possono essere utilizzate per la personalizzazione, lo sviluppo di un sistema, la modifica di un sito, *business inteliigence* e caratterizzazione di utilizzo.

All'interno del monitoring della navigazione Web è interessante il moni-

toring di portali, in quanto formati da componenti indipendenti e variabili chiamate *portlet*. Le portlet sono quindi le entità principali e fondamentali che formano un portale web: queste sono componenti che migliorano lo sviluppo e la manutenzione del portale e ne consentono la riusabilità. Una portlet è una componente web basata sulla tecnologia Java e il contenuto da essa generato è denominato anche frammento: esso corrisponde a codice di markup (HTML, XHTML, WML) che soddisfa determinate regole. Il frammento di una stessa portlet può variare da un utente all'altro a seconda della configurazione apportata ad essa.
I portali Web permettono agli utenti di accedere facilmente e in maniera trasparente all'informazione: nati come evoluzione dei motori di ricerca, i portali di seconda generazione permettono invece l'accesso a multiple ed eterogenee risorse di dati e applicazioni tramite una singola interfaccia, consentendo personalizzazione da parte dell'utente nell'aggregazione di contenuti. Essi hanno associato agli strumenti tipici dei motori di ricerca (*search engine* e categorizzazione delle informazioni) altri servizi, informativi e non, allo scopo di proporsi come accesso preferenziale e guida per la navigazione via Internet.
L'esigenza di un sistema in grado di monitorare gli accessi e la navigazione degli utenti in un portale web è intrinseca nella definizione di portale e dai presupposti che esso ha come interfaccia altamente personalizzabile. I *portal framework* producono pagine web come risultato di un'aggregazione di portlet: una pagina di un portale può quindi contenere un contenuto altamente eterogeneo e che ha origine da producer di portlet differenti.
Gli strumenti e le analisi proposte finora non risultano sufficienti per una piena comprensione del comportamento dell'utente nel caso di portali Web: non considerano le relazioni fra contenuto (portlet) e contenitore (pagina), e i legami che possono esserci fra utenti e componenti (il layout di una pagina può cambiare a seconda di determinate caratteristiche dell'utente). Le comuni metriche di analisi basate sulla visita della pagina nel suo complesso non sono sufficienti a inquadrare completamente gli interessi dell'utente e il suo reale comportamento di navigazione. Partendo da misure standard

come il numero di visualizzazioni di una pagina, è necessario poi introdurre nuovi parametri di analisi *portlet-based*: la visibilità (in termine di tempo e percentuale di spazio occupato sullo schermo) che possono avere le diverse componenti durante la navigazione, le diverse interazioni dell'utente con le singole componenti anche per rilevare relazioni fra di esse e l'influenza delle singole portlet sulla performance dell'intero portale, questi sono sono esempi di indicatori essenziali per catturare conoscenza riguardo agli interessi degli utenti e per avere suggerimenti utili nella progettazione di un portale (a livello di design e di personalizzazione).

EOP (Eye-On -Portal) è il sistema esposto nei prossimi capitoli e si propone come strumento per riuscire a catturare informazioni dettagliate sulle componenti della pagina visitata dall'utente e sulle interazioni di quest'ultimo con il portale. Il framework è caratterizzato da una struttura a tre livelli: una fase di logging di *raw data*, che devono essere raccolti con performance elevate e in modo non invasivo, e forniranno la base per le analisi successive; una fase di traduzione di questi dati in dati strutturati, che richiedono quindi un'elaborazione per essere chiari e utilizzabili dall'utente per eventuali applicazioni successive (ultima fase).
I dati raccolti possono avere utilità nella personalizzazione, nello sviluppo di un sistema, nella modifica di un sito, nel campo della *business intelligence* e nella caratterizzazione di utilizzo. La soluzione proposta ha l'obiettivo di arrivare a dati più specifici e ad un livello più profondo della semplice sequenza di pagine visualizzate, la necessità è quella anche di ottenere misure client-side: tempi di visualizzazione della pagina e delle sue componenti, movimenti del mouse, scrolling della pagina e altre interazioni dell'utente con la pagina.
L'idea è nata con la collaborazione di Engineering Ingegneria Informatica Spa, dove ho svolto un tirocinio formativo nel mese di Ottobre 2014: il caso d'uso affrontato si riferisce a un portale web intranet del Comune di Bologna, il quale ha fatto emergere la necessità di un tipo di monitoraggio più appro-

fondito per permettere all'azienda di effettuare scelte in termini di struttura del portale ai fini di rendere più efficace e performante la navigazione del portale per ogni tipologia di utente.

La presente trattazione ha la seguente struttura: nel capitolo 1 verrà descritto l'attuale stato dell'arte delle ricerche nel campo del *Web usage mining*, i suoi limiti e la necessità di un nuovo modello. Si andranno poi a introdurre nel capitolo 2 i concetti fondamentali alla base del funzionamento di un portale e delle sue componenti. La descrizione del modello proposto per il monitoring portlet-based è affrontata nel capitolo 3: viene approfondita ogni fase della struttura proposta, quali obiettivi e quali dati si propone di ottenere ognuna di queste. Infine nel capitolo 4 verrà proposto un caso di studio che vedrà l'implementazione della sola parte di logging di navigazione di un utente, in un portale intranet del Comune di Bologna, sviluppato tramite la piattaforma Liferay.

# Indice

# Elenco delle figure

# Capitolo 1

# Monitoring della navigazione Web

Nel mondo del WWW enormi quantità di dati sono prodotti ogni giorno come record dettagliati del comportamento di utilizzo del Web, ma l'obiettivo di trarne conoscenza rimane ancora una sfida. Data la quantità e l'eterogeneità dei dati, si è sempre posto il problema di estrarre ed evidenziare informazioni utili per fini commerciali ed economici, per migliorare la struttura dei siti, per ricavarne statistiche, per la personalizzazione dei siti o per classificare i contenuti e gli utenti del web: tipi differenti di dati devono essere organizzati e gestiti in modo tale che utenti diversi possano avervi accesso in maniera efficiente.
Per questo, l'applicazione delle tecniche di *data mining* nel Web sono ora al centro di un numero sempre più crescente di ricerche. Per ***data mining***, che rappresenta un'area disciplinare di recente costruzione, si intende il processo di selezione, esplorazione e modellazione di grandi masse di dati, al fine di scoprire regolarità o relazioni non note a priori, e allo scopo di ottenere un risultato chiaro e utile al proprietario del database; permette quindi di raccogliere e organizzare i dati in strutture che facilitano l'accesso e il trasferimento delle informazioni verso tutti i possibili fruitori.

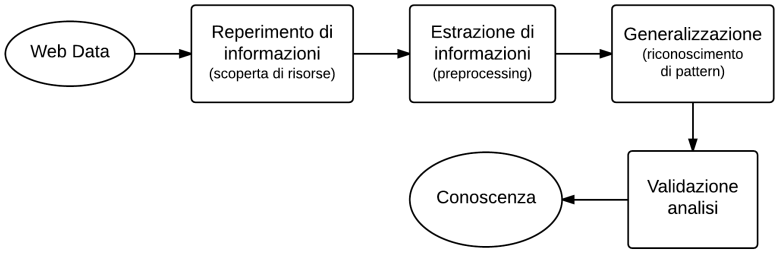


Figura 1.1: Web Mining Subtask

Diversi metodi di *data mining* sono usati per scoprire informazioni nascoste nel Web: con l'espressione ***Web mining*** si riferisce all'applicazione di procedure analoghe per estrarre automaticamente informazioni dalle risorse presenti nel Web (sia documenti che servizi) (figura 1.1). L'obiettivo del *Web mining* trova giustificazione nell'opinione diffusa che l'informazione presente nel Web sia sufficientemente strutturata da consentire un'efficace applicazione di tecniche statistiche e di apprendimento automatico.
Il *Web mining* può essere categorizzato in tre aree di interesse, in base a quale parte del Web si intende analizzare:

- ***Web content mining***: si concentra sulle informazioni grezze disponibili nelle pagine web ed ha come scopo la classificazione e l'ordinamento delle pagine in base al contenuto;

- ***Web structure mining***: è il processo che, utilizzando la teoria dei grafi, analizza i nodi e la struttura dei collegamenti di un sito web;

- ***Web usage mining*** è la sottoarea che si occupa dell'estrazione di conoscenza dai file di log della navigazione web di un utente; è la parte che è di maggiore interesse per questa trattazione.

## 1.1 Web usage mining

Il *Web usage mining* utilizza le tecniche di *data mining* per scoprire pattern di utilizzo da applicazioni web-based e pr prevedere il comportamento dell'utente nella sua interazione con il Web: si occupa dell'estrazione di conoscenza a partire dai file di log di navigazione.

Il processo di *Web usage mining* può essere considerato come un processo a tre fasi: nella prima fase i dati dei log vengono sottoposti al *preprocessing*, al fine di ricavare i *raw data* (identificare utenti, sessioni...) dalle risorse web; nella seconda fase, vengono applicati metodi statistici, regole associative, analisi di raggruppamento e classificazione per scoprire dei modelli; questi modelli vengono memorizzati in modo tale da essere utilizzati nella fase successiva di analisi dei pattern scoperti e verificarne la validità.

Le applicazioni di *Web usage mining* sono basate sulla raccolta dei dati forniti da tre fonti principali: web server, proxy server e web clients.
Dal momento che possono raccogliere una grande quantità di informazioni nei loro log file, i web server rappresentano sicuramente la più ricca e la più comune fonte di dati. Ogni accesso a una pagina web viene registrati dei log di accessi del web server che la ospita: la registrazione di un web log consiste in campi che seguono un formato predefinito (Common Log Format[1]).
I dati i utilizzo possono anche essere rintracciati a livello client (utilizzando JavaScript ad esempio): queste tecniche evitano il problema dell'identificazione delle sessioni utente e forniscono dettagliate informazioni riguardo al comportamento effettivo degli utenti.

Gli anni recenti hanno visto prosperare la ricerca nell'area del Web Mi-

[1]Il Common Log Format (`http://publib.boulder.ibm.com/tividd/td/ITWSA/ITWSA_info45/en_US/HTML/guide/c-logs.html#common`) è un formato standard di file di testo usato dai web server quando generano file di log. La sintassi è la seguente: *host ident authuser date request status bytes.*

ning e particolarmente del *Web usage mining*. Dai primi articoli pubblicati nella metà degli anni '90, c'è stata un grande aumento della letteratura riguardante la modellazione del comportamento di navigazione e molte sono le ricerche fatte in questo campo: scoprire pattern ricorrenti nei log [13], modellare i pattern di navigazione degli utenti [12], effettuare cluster di utenti di un sito specifico [18].

Ivàncsy R. [13] mostra una panoramica di come usare tecniche di mining individuando pattern frequenti nei log di navigazione web: i tre pattern da ricercare sono i set di pagine più frequenti (*itemset*), le sequenze di pagine e i pattern "ad albero" (*tree pattern*); vengono inoltre proposti algoritmi per rendere la ricerca di questi pattern il più efficiente possibile.

Hoxha J. [12] affronta il problema dell'eterogeneità delle informazioni disponibili sul Web, proponendo una formalizzazione semantica del comportamento dell'utente nella navigazione fra diversi siti web (*cross-site*): viene introdotto il *Web browsing Activity Model* (WAM) che permette una concettualizzazione condivisa della conoscenza fra i diversi domini dove i log di utilizzo sono registrati; il contributo degli autori si estende anche all'approccio per arricchire i log di utilizzo con la semantica, combinando le tecnologie del *Semantic Web* e le tecniche di *Machine Learning* (una delle aree fondamentali dell'intelligenza artificiale e si occupa della realizzazione di sistemi e algoritmi che si basano su osservazioni come dati per la sintesi di nuova conoscenza).

Liu Y. [18] mostra un approccio basato sul modello di Markov nell'effettuare il *clustering* degli utenti, utilizzato per studiare i pattern di navigazione degli utenti del Web in applicazioni reali. Si suppone che gli oggetti nel modello seguano una serie limitata di distribuzioni di probabilità, così che ogni distribuzione componente esprima un cluster. L'approccio basato su questo modello garantisce vantaggi rispetto ai tradizionali approcci non probabilistici, permettendo di classificare gli utenti basandosi sulle sequenze di richieste dell'utente e consentendo anche il cluster di incertezze, che è importante specialmente per quegli oggetti che sono vicini ai limiti del cluster.

Tramite il *Web usage mining* si cerca di comprendere le scelte di navigazione degli utenti: gli scopi e i mezzi di questa ricerca sono molteplici e condivisi dalla letteratura esistente riguardo questo tema. Srivastava et al. [26] afferma che analisi come regole di associazione, classificazione, clustering, pattern sequenziali e modelli di dipendenza possono essere utilizzati per la personalizzazione, lo sviluppo di un sistema, la modifica di un sito, *business inteliigence* e caratterizzazione di utilizzo. Joshi [16] enuncia che il clustering per il *Web mining* dovrebbero trovare naturali gruppi di utenti, pagine o altro; le associazioni dovrebbero essere studio di quali URL tendono ad essere richiesti insieme; e le analisi sequenziali dovrebbero rilevare l'ordine in cui gli URL tendono a ricevere l'accesso. Queste analisi possono essere utilizzati per la personalizzazione.

## 1.2 Caratterizzare l'attività dell'utente

Le informazioni fornite dalle risorse di log possono essere usate per identificare e costruire diversi livelli di astrazione dei dati: utenti, sessioni server, *clickstream*, *pageview*. Al fine di dare una consistenza al modo in cui questi termini sono necessari la W3C Web Characterization Activity (WCA) [27] ha pubblicato una stesura delle definizioni [2] dei termini rilevanti per analizzare il *Web usage*.
La W3C, è un'organizzazione non governativa internazionale che ha come scopo quello di sviluppare tutte le potenzialità del World Wide Web: al fine di riuscire nel proprio intento, la principale attività svolta dal W3C consiste nello stabilire standard tecnici per il World Wide Web inerenti sia i linguaggi di markup che i protocolli di comunicazione. L'attività di Web Characterization svolta dal W3C si occupa di osservare i pattern globali di utilizzo e struttura del Web. L'interesse si concentra sulla caratterizzazione del Web tramite la misura di aspetti come i pattern di accesso, il tipo di dati a cui si ha accesso, byte trasferiti, popolarità di risorse. Atraverso questo sguardo

[2] `http://www.w3.org/1999/05/WCA-terms/01`

sulle dinamiche del Web, il W3C si propone di poter adattare al meglio le proprie scelte per contribuire all'evoluzione del Web.
Di seguito vengono riportate le definizioni della WCA, rilevanti per analizzare il comportamento di navigazione.

- **User**: singolo individuo che accede a file da uno o più Web server attraverso un browser. In pratica è molto difficile individuare singolarmente e ripetutamente un utente: egli può accere al Web attraverso diversi computer, o usare più di un tipo di browser sullo stesso computer.

- **Pageview**: consiste in ogni file che contribuisce alla visualizzazione della pagina sul browser dell'utente in un dato momento. Il dato è associato alla visualizzazione aggregata della pagina, non alle diverse componenti.

- **Clickstream**: serie sequenziale di richieste di *pageview*. Tuttavia, i dati disponibili dal lato server non forniscono sempre abbastanza informazioni per ricostruire l'intera sequenza di clic per un sito.

- **User session**: *clickstream* per un singolo utente attraverso l'intero Web.

- **Server session**: insieme delle *pageview* in una sessione utente per un particolare sito.

Proprio per arrivare a dati più specifici e ad un livello più profondo della semplice sequenza di pagine visualizzate, la necessità è quella anche di ottenere misure client-side: tempi di visualizzazione della pagina e delle sue componenti, movimenti del mouse, scrolling della pagina e altre interazioni dell'utente con la pagina. Fenstermacher e Ginsburg [9], discutendo del monitoring client-side, suggeriscono di utilizzare le stesse tecniche di clustering egli altri approcci del *Web mining* anche per l'analisi client-side e sostengono che solo con quest'ultima sia possibile avere una panoramica più ricca e completa del comportamento dell'utente sul Web.

Inoltre un tema molto discusso e di rilievo riguardo ai portali web è la loro usabilità e di conseguenza a quella delle portlet: per andare incontro alle esigenze dell'utente e modulare la pagina web in maniera opportuna è necessario andare ad analizzare le interazioni degli utenti con ogni singola componente. Anche questo è solo possibile con un'analisi client-side.

Il caso affrontato in questa trattazione riguarda un modello di analisi specifico per pagine web generate da *portal framework*: la maggior parte degli approcci sopra citati però non tiene conto della struttura della pagina in componenti indipendenti e variabili quali le **portlet**, aggregate al momento della creazione della pagina stessa (questo meccanismo verrà approfondito nel capitolo seguente). Le tecniche di analisi proposte dalla letteratura non considerano le relazioni fra contenuto (portlet) e contenitore (pagina), e i legami che possono esserci fra utenti e componenti (il layout di una pagina può cambiare a seconda di determinate caratteristiche dell'utente): le comuni metriche di analisi basate sulla visita della pagina non sono sufficienti a inquadrare completamente gli interessi dell'utente e il suo reale comportamento di navigazione.

Il sistema esposto nei prossimi capitoli, si propone come strumento per riuscire a catturare informazioni dettagliate sulle componenti della pagina visitata dall'utente e sulle interazioni di quest'ultimo con il portale: i dati raccolti potrebbero avere utilità nell'ottimizzazione del layout e nell'usabilità del portale.

# Capitolo 2

# Portlet-based monitoring

L'esigenza di un sistema in grado di monitorare gli accessi e la navigazione degli utenti in un portale web è intrinseca nella definizione di portale e dai presupposti che esso ha come interfaccia altamente personalizzabile. Come già discusso nel capitolo precedente, *portal framework* producono pagine web come risultato di un'aggregazione di portlet: una pagina di un portale può quindi contenere un contenuto altamente eterogeneo e che ha origine da producer di portlet differenti.
Proprio per l'eterogeneità del contenuto che possono presentare, per le pagine di un portale i tradizionali indici di analisi della navigazione, come il numero di visite della pagina, sono inadeguati: il sistema proposto ha lo scopo di ottenere dati più dettagliati a livello *portlet* mirati alla comprensione degli interessi e dal comportamento dell'utente durante la navigazione. I dati ottenuti dal sistema possono avere un utilizzo in diversi campi: analisi del traffico web, del design del portale orientato alle necessità dell'utente e alla sua usabilità.

In questo capitolo analizzeremo in dettaglio la struttura di un portale e le informazioni interessanti da monitorare per comprendere al meglio il comportamento dell'utente nel *browsing*, senza tralasciare la modularità in componenti indipendenti che caratterizza i portali Web.

## 2.1 Elementi e funzionamento di un portale

In questa sezione ci si occuperà di fornire alcune nozioni fondamentali necessarie alla comprensione del lavoro svolto: verrà definito il concetto di portale, le sue componenti principali e il suo funzionamento.

### 2.1.1 Definizioni generali

I **portali web** permettono agli utenti di accedere facilmente e in maniera trasparente all'informazione: nati come evoluzione dei motori di ricerca, i portali di seconda generazione permettono invece l'accesso a multiple ed eterogenee risorse di dati e applicazioni tramite una singola interfaccia, consentendo personalizzazione da parte dell'utente nell'aggregazione di contenuti [19]. Essi hanno associato agli strumenti tipici dei motori di ricerca (*search engine* e categorizzazione delle informazioni) altri servizi, informativi e non, allo scopo di proporsi come accesso preferenziale e guida per la navigazione via Internet.
Il portale può essere quindi definito un aggregatore di informazione: il suo obiettivo primario è quello di creare un ambiente di lavoro in cui l'utente possa facilmente navigarci.
I portali di seconda generazione [2] sono ben lontani dall'essere un sistema monolitico: la loro complessità richiede un'architettura *component-oriented*, dove ogni pagina è composta da *portlet* personalizzabili.

Le **portlet** sono quindi le entità principali e fondamentali che formano un portale web: queste sono componenti che migliorano lo sviluppo e la manutenzione del portale e ne consentono la riusabilità. Una portlet è una componente web basata sulla tecnologia Java e il contenuto da essa generato è denominato anche frammento: esso corrisponde a codice di markup (HTML, XHTML, WML[1]) che soddisfa determinate regole. Il frammento di

[1]Wireless Markup Language (`http://www.wapforum.org/DTD/wml20.dtd`): basato su XML, è un linguaggio di markup sviluppato per l'implementazione delle specifiche del

una stessa portlet può variare da un utente all'altro a seconda della configurazione apportata ad essa.

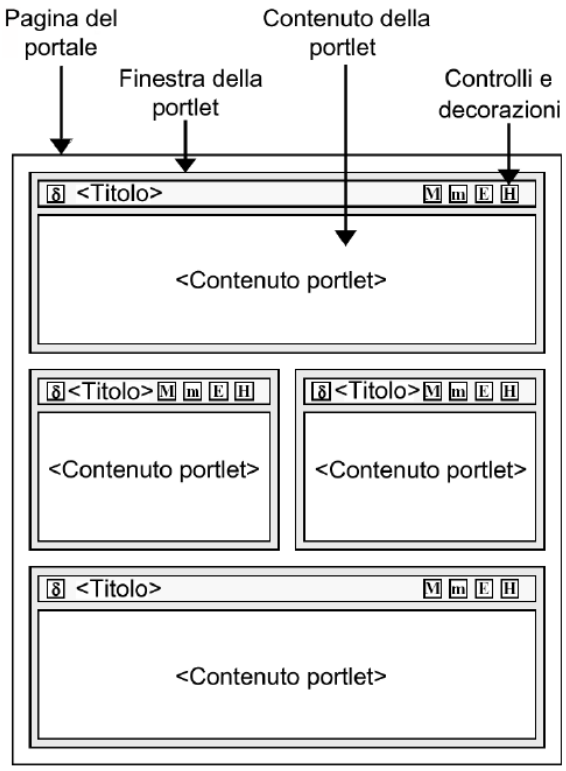


Figura 2.1: Pagina formata da portlet

Visivamente, le portlet sono porzioni di pagina e funzionalmente sono dei canali informativi. L'applicazione di questa modularità permette agli utenti di costruire una pagina per aggregazione dei singoli elementi, avendo come risultato finale una pagina web configurabile ed adattabile alle sue volontà.

La rappresentazione finale è affidata per gran parte al ***portlet container*** che, come verrà spiegato in seguito, gestisce il loro ciclo di vita. Infatti, il suo compito è di seguire le azioni dell'utente, eseguire la logica di business di tutte le portlet presenti nella pagina ed infine passare al portale i frammenti

protocollo WAP.

generati dalle stesse. In seguito il portale aggrega il risultato proveniente da ognuna di esse in un pannello tipo quello in figura 2.1.

### 2.1.2 Creazione di una pagina del portale

Il portlet container riceve i contenuti generati dalle diverse portlet. Questi vengono passati al portale, il quale crea la pagina aggregando i vari frammenti delle portlet e la invia al dispositivo client (browser) dove viene visualizzata. Il meccanismo appena descritto si può riportare nella figura 2.2.

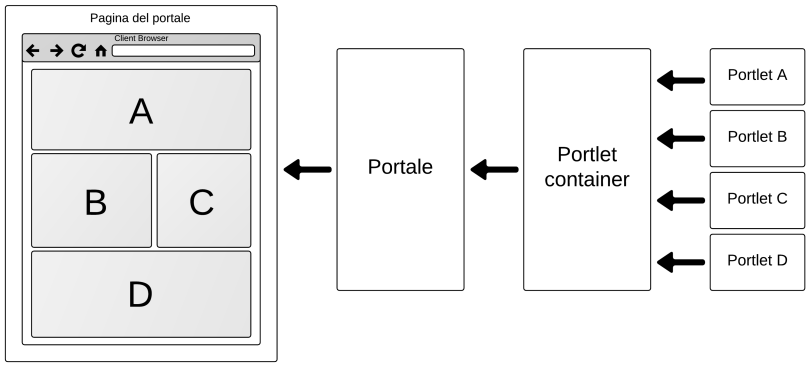


Figura 2.2: Creazione di una pagina del portale

Gli utenti possono accedere ad un portale utilizzando un comune browser web. Al momento di ricevere la richiesta HTTP della pagina, il portale determina la lista di portlet di cui necessita per soddisfare la richiesta e, attraverso il portlet container, le invoca. In tal modo, il portale crea la pagina con i frammenti generati dalle portlet che viene ritornata al client.
Il portlet container non è quindi responsabile dell'aggregazione dei contenuti generati dalle diverse portlet: questo è, infatti, compito del portale.
Portale e container possono essere integrati insieme come un componente singolo di una suite di applicazioni oppure come due parti separate di una *portal application*.

### 2.1.3 Ciclo di vita di una portlet

Una portlet è gestita attraverso un ciclo di vita ben definito che determina come è caricata, istanziata ed inizializzata, in che modo tratta le richieste che provengono dai client e come le soddisfa.

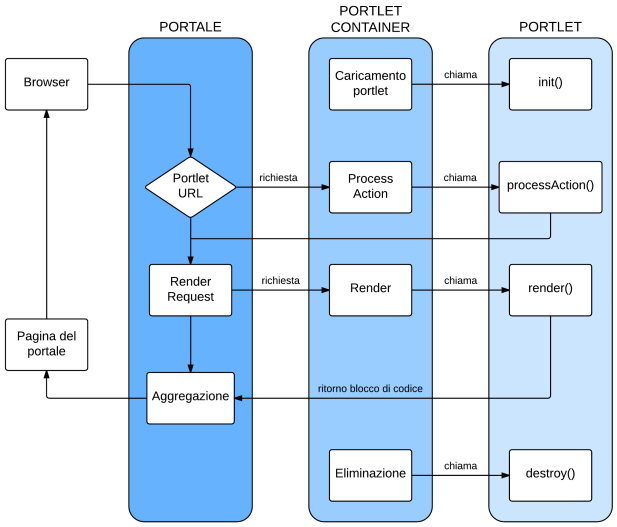


Figura 2.3: Ciclo di vita di una portlet

Le varie fasi sono individuate dai tre seguenti stati:

- caricamento delle classi e inizializzazione;
- gestione della request;
- distruzione della portlet.

L'interfaccia `Portlet` è la principale astrazione delle Portlet API (package `javax.portlet`) [15]. Tutte le portlet implementano questa interfaccia o direttamente oppure, più comunemente, estendendo una classe che la implementa. La specifica, infatti, include la classe `GenericPortlet` che implementa l'interfaccia `Portlet` e fornisce delle funzionalità di base. Gli sviluppatori

dovrebbero estendere, direttamente o indirettamente, tale classe per implementare le loro portlet.
Le fasi sopracitate sono gestite attraverso quattro metodi esposti dall'interfaccia `Portlet`:

- init(PortletConfig config), che viene chiamato dal container quando la portlet viene inizializzata; presenta come parametro di invocazione il contesto di configurazione, definito dall'interfaccia `PortletConfig` del package `javax.portlet`. L'inizializzazione avviene al momento dello startup dell'applicazione o del portal server o in alternativa al momento della prima invocazione da parte del client;

- destroy(), metodo invocato dal container quando la portlet viene distrutta. Il portlet container distrugge la portlet quando ritiene non più necessario il suo utilizzo nel container. Questa decisione viene presa se non ci sono più richieste in corso (e per un qualche motivo si deve fare shutdown dell'applicazione), quando tutti i thread di rendering o di action hanno terminato la loro esecuzione oppure quando il processo di inizializzazione si è concluso;

- processAction(ActionRequest request, ActionResponse response), chiamato dopo che l'utente ha effettuato una richiesta e serve a processare i dati avuti in input;

- render(RenderRequest request, RenderResponse response), il quale esegue la visualizzazione delle portlet.

Dopo che la portlet è opportunamente inizializzata, il container può invocarla per gestire le richieste che provengono dal client. La richiesta eseguita dal client sul portale viene tradotta in una *render request* o *action request* a seconda della tipologia di invocazione. Per questo scopo, l'interfaccia `Portlet` definisce i due metodi `render` e `processAction`. Tipicamente le richieste del client sono innescate da URL creati dalle portlet, chiamati appunto *portlet URL*. I portlet URL possono essere di due tipi: *action URL* o *render URL*.

## 2.2 Come monitorare la navigazione in un portale?

Data la grande eterogeneità di contenuto che li caratterizza, i portali necessitano di misure, di tecniche e di analisi che mirano ad analizzare il traffico web ad un livello più profondo e capillare, *portlet-based*. Gli strumenti di analisi tipici del *Web usage mining* rimangono un ottimo punto di partenza e sono in grado di estrapolare informazioni utili alla comprensione del comportamento dell'utente nel *browsing*, ma è altresì necessario non tralasciare la modularità in componenti indipendenti che caratterizza i portali Web.

### 2.2.1 Scoperta di pattern

Il *pattern discovery* si forma su metodi e algoritmi sviluppati da diversi campi della statistica, del *data mining* e *machine learning* (una delle aree fondamentali dell'intelligenza artificiale, si occupa della realizzazione di sistemi e algoritmi che si basano su osservazioni come dati per la sintesi di nuova conoscenza). Vengono ora descritte alcune di queste attività che possono essere applicate al dominio del Web, e in particolare possono essere utili per l'analisi della navigazione degli utenti.

#### Analisi statistiche

Le analisi statistiche sono il metodo più comune per estrarre conoscenza riguardo i visitatori di un sito Web. Analizzando una sessione di navigazione, si possono applicare diversi tipi di analisi statistiche (frequenza, media, mediana,...) su variabili come visualizzazione delle pagine, tempo di visualizzazione e lunghezza del percorso di navigazione. Esistono diversi tool (come Google Analytics [2], Mint [3], Spring Metrics [4]) in grado di produrre un report contente informazioni statistiche come le pagine più frequentate,

[2] http://www.google.com/analytics/
[3] http://haveamint.com/
[4] http://www.springmetrics.com/

tempo medi di visualizzazione di una pagine o la lunghezza media di un percorso di navigazione di un certo sito. Nonostante queste analisi non entrino particolarmente nel dettaglio, questo tipo di conoscenza può essere potenzialmente utile per migliorare le performance di un sistema, aumentarne la sicurezza, provvedere supporto per decisioni di marketing.

### Regole di associazione

La generazione di regole di associazione può essere utilizzata per mettere in relazione pagine Web che compaiono spesso insieme in una singola sessione. Nel contesto del *Web usage mining*, le regole di associazione si riferiscono a set di pagine a cui si accede insieme con un valore che supera una determinata soglia. Queste pagine possono anche non essere direttamente collegate fra loro: l'analisi può essere effettuata riguardo anche osservando l'interazione dell'utente con un set di portlet per scoprire connessioni fra portlet apparentemente non collegate fra loro. L'analisi può trovare applicazioni di business e di marketing, ma soprattutto la presenza o assenza di queste regole può essere utile per il desgin del portale.

### Clutering

Il *clustering* è una tecnica che si occupa di individuare gruppi di oggetti con simili caratteristiche. Nel dominio del *Web usage* ci sono due tipi di cluster interessanti da scoprire: cluster di utilizzo e cluster di pagine. Il cluster di utenti tende a stabilire gruppi di utenti che mostrano pattern simili di navigazione (fra pagine e fra portlet). Queste informazioni possono essere utili per effettuare sutdi sulla segmentazione del mercato (soprattutto in caso di e-commerce) o per fornire contenuti personalizzati agli utenti sul Web (si pensi a una pagina dinamicamente creata con portlet che riflettono gli interessi di un certo gruppo di utenti).

### Classificazione

La classificazione si pone l'obiettivo di mappare dati in diverse classi predefinite. Nel dominio del Web può essere interessante sviluppare profili di utenti che appartengono a una particolare classe o categoria: questo richiede un'estrazione delle caratteristiche che meglio descrivono le proprietà di una data classe o categoria.
La classificazione può essere effettuata tramite algoritmi di apprendimento induttivo supervisionato, una tecnica di apprendimento automatico che mira a istruire un sistema informatico in modo da consentirgli di risolvere dei compiti in maniera autonoma sulla base di una serie di esempi ideali, costituiti da coppie di input e di output desiderati, che gli vengono inizialmente forniti.

### Pattern sequenziali

La scoperta di pattern sequenziali mira a scoprire pattern di oggetti in ordine temporale all'interno delle sessioni, con lo scopo ad esempio di predire la navigazione di un utente. I risultati che si possono osservare da questa analisi, possono essere utili anche per la rilevazione di associazioni non immediate fra compontenti del portale.

### Modelli di dipendenza

La modellazione di dipendenze è un altro compito della scoperta dei pattern nel *Web mining usage*: l'obiettivo è quello di sviluppare un modello capace di rappresentare dipendenze significative fra le variabili del dominio del Web. Modellare i pattern di utilizzo del Web non solo fornisce un framework teorico per analizzare il comportamento degli utenti, ma è anche potenzialmente utile per prevedere il futuro utilizzo delle risorse sul Web.
Queste informazioni possono migliorare l'usabilità di un portale e delle sue componenti, come approfondito successivamente nella sezione 3.4.3.

### 2.2.2 Indicatori dell'interesse di un utente

La navigazione di utente nel Web lascia una traccia e si possono catturare *raw data* a riguardo, dati primari senza alcun tipo di elaborazione, che possono fornire la base per analisi successive. Di seguito vediamo alcune osservazioni che si possono fare sul comportamento di un utente durante la sua navigazione in un portale Web e come estrarre indicatori che definiscano interessi e preferenze di un utente.

#### Contatore di visualizzazioni

L'analisi più generica e semplice possibile è data dal conteggio delle visualizzazioni di una pagina: questo compito è svolto da molti tool già esistenti ma può essere un ottimo punto di partenza per capire su quali pagine concentrare l'attenzione per le successive analisi. Questa statistica è facilmente ottenibile tramite un contatore che indica quante volte compare una pagina nei log. A partire da queste informazioni si può selezionare un sottoinsieme delle pagine del portale, e da qui far partire le analisi ad un livello più profondo: un'analisi *portlet-based*.

#### Performance del portale

Calcolando ed estraendo il tempo di rendering della pagine e delle singole portlet che la compongono, è possibile avere un'idea delle performance generali del portale (tempo medio di caricamento delle pagine) ed individuare eventuali anomalie.
Si calcola il tempo medio di caricamento di ogni portlet e si confronta con quello delle altre portlet della stessa pagina: esportando graficamente questi dati è possibile evidenziare eventuali picchi di caricamento di alcune portlet che potrebbbero influire negativamente sulla navigazione dell'utente.

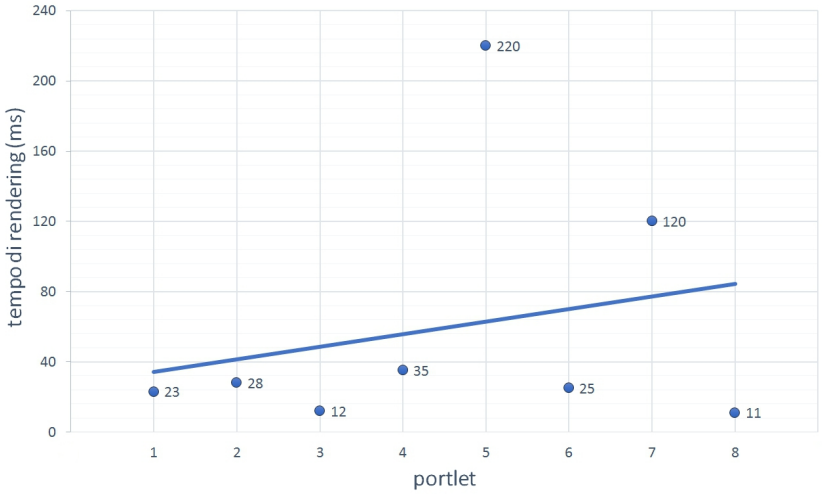


Figura 2.4: Esempio di grafico del tempo di rendering di una portlet

### Visibilità di una portlet

Un'analisi più specifica per comprendere il comportamento dell'utente durante la sua navigazione nel portale può essere ottenuta calcolando il tempo di visibilità di una portlet. Questa informazione, dopo aver selezionato quali pagine sono più visualizzate dall'utente, è molto utile per conoscere l'interesse rigurdante la singola unità di contenuto della pagina.
Il layout di un portale è solitamente organizzato per colonne (due o tre) e la pagina può contenere altri elementi oltre alle portlet (menù, header, footer): se il numero e la dimensione delle portlet è tale da eccedere la dimensione della finestra del browser, solo una parte della pagina sarà mostrata, mentre altre saranno visibili solo attraverso lo scroll della pagina. Così in ogni istante alcune portlet potrebbero avere piena visibilità e altre potrebbero essere nascoste o parzialmente visibili.
Per ricavare le informazioni utili a determinare la visibilità di una singola portlet, il *logger* ad ogni evento significativo (scroll della pagina o caricamento) registra l'istante temporale (*timestamp*) e la posizione delle portlet da monitorare: in questo modo si può calcolare quanto una portlet è rimasta visibile o meno durante la navigazione.

L'analisi proposta da [6] si sofferma anche sulla visibilità parziale di una portlet, proponendo un indicatore per quantificare la percentuale di visibilità:

$$V = \frac{\sum_i (t_i * v_i)}{T} \tag{2.1}$$

V (indicatore della visibilità) viene calcolato come la media pesata di tutti gli intervalli di tempo in cui la portlet è visibile: $t_i$ è il il tempo e $v_i$ è la percentuale di visibilità nell'i-esimo intervallo, T è il tempo totale di visita della pagina.

### Interazione con le portlet

Una portlet può essere solo informativa oppure prevedere un'interazione con l'utente (ad esempio un form). Possiamo quindi tenere traccia di tuttle le interazioni con le portlet, distinguendone i vari tipi (clic del mouse, mouseover, pressione da tastiera).
Un altro indicatore interessante da considerare, come suggerito da [6], è il tempo totale in cui il puntatore del mouse rimane all'interno di una portlet: uno studio sull'*eye tracking* [5] mostra che c'è una correlazione tra i movimenti dell'occhio e i movimenti del mouse. Tracciando quindi il percorso segnato dal puntatore del mouse, è possibile quindi ottenere il probabile percorso dell'occhio dell'utente e quindi vedere a quale parte della pagina web può essere più interessato. Ricostruire tutti i movimenti del mouse potrebbe essere molto oneroso ma è possibile calcolare quanti eventi registrati dal logger (relativi al movimento del mouse) sono avvenuti all'interno di una portlet o meno.

### Tempo speso su una pagina

Come sottolineato da [11] e accennato nel capitolo 1, le ricerche nel campo del *Web usage mining* si sono sempre molto concentrate sull'analisi dell'or-

dine e della frequenza delle pagine visitate. Molti studi però sull'*information retrival* e sull'interazione uomo-macchina hanno rilevato che il tempo speso su una pagina web (TSP) sia una misura importante per le intenzioni dell'utente e come indicatore per la rilevanza della pagina stessa. In questo caso il TSP non è un'informazione a livello di portlet ma è comunque un indicatore valido per analizzare l'interesse dell'utente anche se facilmente soggetto a fattori che lo rendono di difficile interpretazione.
Il TSP ($t_i$) per una data pagina ($p_i$) è facilmente calcolabile come la differenza fra il timestamp dell'accesso alla pagina e il timestamp della pagina precedente. È da notare che i log ovviamente non conterranno sufficienti informazioni per calcolare il TSP per l'ultima pagina visitata da un utente, ma sul lato client si può intervenire per trovare una soluzione (soprattutto nel caso in cui l'accesso al portale necessiti di autenticazione). Inoltre per rendere veramente affidabile questo indicatore è necessario considerare ed escludere dal precedente calcolo anche i tempi di caricamento della pagina e il "fattore distrazione dell'utente". Per quanto riguarda il tempo di generazione della pagina, esso è contenuto nel log e quindi può essere sottratto al TSP precedentemente calcolato.
Il fattore "distrazione" può essere determinato da una moltitudine di altri fattori e attività che portano ad un aumento del TSP (anche se l'utente non sta attivamente guardando la pagina) che non può trovare risposta nei log: in questi casi si tende ad affidarsi a euristihce basate sull'osservazione di dati reali. È norma settare un valore soglia ragionevole per il TSP (che dipende principalmente da dominio web di cui si tratta); altri studi ritengono necessario anche il definire un valore minimo per escludere possibili *outlier* (valori anomali) causati da clic accidentali o reindirizzamenti.

# Capitolo 3

# EOP: un framework per il monitoring portlet-based

## 3.1 Architettura del framework per il monitoring

EOP(Eye-On-Portal) è un framework di monitoring che si propone di ottenere dati dettagliati sulla navigazione dell'utente nel portale e in particolare analizzare quali componenti (portlet) interessano maggiormente all'utente.
In EOP possiamo individuare tre diverse parti che lo compongono(figura 3.1):

1. logging di *raw data*;
2. traduzione da *raw data* a dati strutturati;
3. *mining* sui dati strutturati.

Il componente di logging si occupa della raccolta dei *raw data*, quindi di quei dati primari non soggetti a nessun tipo di manipolazione e processing, raccolti dalla navigazione dell'utente. Il risultato sarà un log delle azioni compiute dall'utente, con un livello di profondità di cattura dei dati che può essere configurato. Questi dati vengono poi processati e trasformati di dati strutturati per evidenziare ed estrapolare le informazioni necessarie alla comprensione

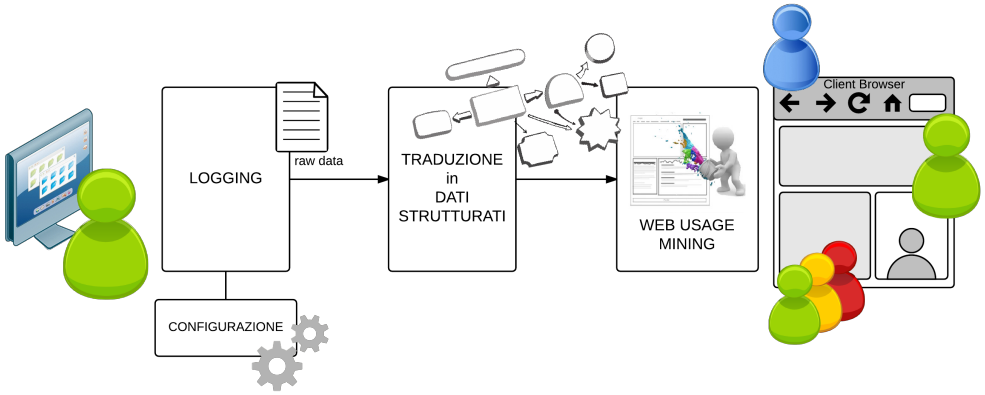


Figura 3.1: Architettura di EOP

del comportamento e delle preferenze dell'utente, e consentendo una gestione relazionale di queste. Tutte le analisi e i modelli ottenuti avranno un ruolo attivo nel miglioramento della navigazione dell'utente e nella costruzione del portale.

La caratteristica del framework di avere una struttura modulare è nata con la necessità di avere uno strumento non troppo invasivo durante la navigazione dell'utente, e questo conferisce inoltre numerosi vantaggi: la modularità rende EOP facilmente modificabile ed estendibile (con l'integrazione di altri moduli); ne facilita il riutilizzo, in una sua parte o nella sua interezza, e consente di ottenere dati che devono essere significativi poi per diversi scopi. Inoltre lo sviluppo in moduli consente una compatibilità con diversi vendor di portali (Liferay, Websphere...).

Abbiamo visto come la letteratura descriva soluzioni di monitoring lato server: nel caso dei portali web però è necessario anche considerare le relazioni fra contenuto (portlet) e contenitore (pagina), e i legami che possono esserci fra utenti e componenti: queste sono informazioni che le comuni metriche di analisi che non considerano la pagina web nella sua articolazione in diverse componenti, ma come risorsa unica, non riescono a ricavare. Dati

i limiti che i dati ottenuti lato server impongono, è necessaria quindi una soluzione che analizzi il comportamento dell'utente anche lato client, catturando informazioni pù dettagliate riguardo alle azioni compiute durante la navigazione, come quelle descritte nella sezione 2.2 del capitolo precedente.
Per raggiungere gli scopi prefissati, la raccolta e l'elaborazione dei dati utili alla conoscenza del reale comportamento dell'utente vengono affidate ai tre moduli in cui EOP si articola.

## 3.2 Logging di raw data

L'obiettivo del componente di logging è di catturare tutte le richieste alle pagine di un portale durante la navigazione e di salvare tutte le informazioni ricavate in un file di log. Dal lato client un tool si occuperà di catturare tutte le richieste di pagina effettuate da un utente e le informazioni ad esse relative che verranno inviate a intervalli regolari al componente che si occupa di salvare i dati nel file di log.
Il modello dei *raw data* raccolti è il seguente:

- timestamp della richiesta;
- URL della richiesta o identificativo della pagina richiesta;
- username dell'utente che ha effettuato la richiesta;
- tempo di rendering della pagina;
- portlet contenute nella pagina: id / nome e tempo di rendering.

Questo set di dati consente di effettuare una prima analisi sul comportamento dell'utente e sugli accessi generali al portale ma soprattutto può essere utile per analizzare le performance del portale.
Per avere una visione più completa del reale comportamento dell'utente all'interno del portale, è necessario un modulo ulteriore che si occupi di monitorare tutte le azioni dell'utente durante la navigazione. Questo componente

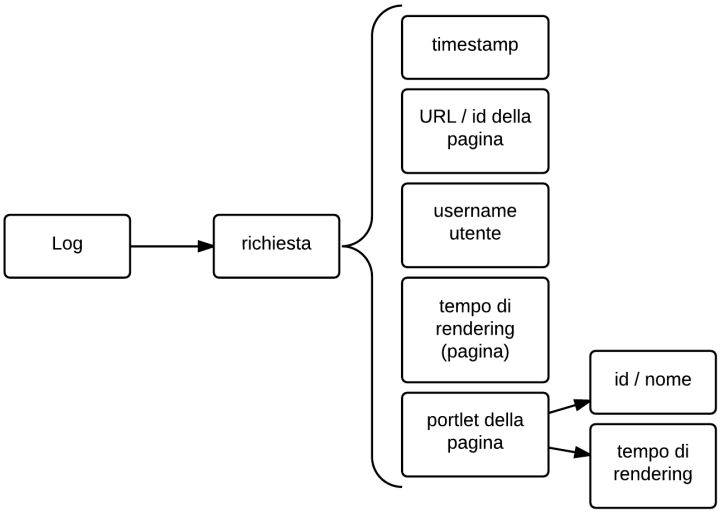


Figura 3.2: Modello dei dati di log

di analisi potrà essere attivato separatamente nel momento in cui si necessita di un'analisi più approfondita. Gli eventi catturati dal framework sono le azioni che avvengono nell'area del client (browser): pressione di un tasto, scrolling della pagina, movimenti del mouse.
I dati che riguardano le interazioni dell'utente sono i seguenti:

- timestamp dell'evento;
- tipo di evento (tasto premuto, scrolling, movimenti del mouse, clic);
- username dell'utente;
- portlet coinvolta (se l'oggetto HTML contenuto in esso è stato coinvolto nell'evento);
- coordinate del puntatore (dove è posizionato il puntatore del mouse nel momento di cui si registra l'evento);
- URL della pagina;
- portlet disponibili e relative coordinate;

- possibili ulteriori informazioni sull'evento (quale bottone è stato cliccato...).

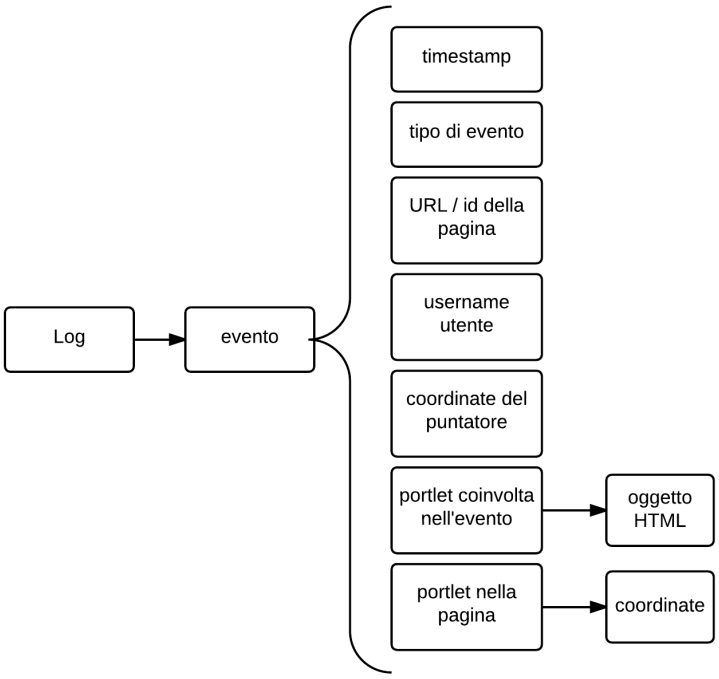


Figura 3.3: Modello dei dati di log (2)

La possibilità di catturare dati come la posizione del puntatore, la posizione delle portlet e il tipo di azione performata dall'utente, permette di poter effettuare successive osservazioni e analisi sul comportamento e le scelte che un utente effettua durante la sua navigazione nel portale, e capire quali contenuti (a livello di portlet) attirano maggiormente il suo interesse.

### 3.2.1 Configurazione

Soprattutto in caso di portali web caratterizzati da un bacino di utenti molto ampio, è necessario controllare la dimensione dei file di log. Per fare fronte a ciò si introducono parametri di configurazione per il componente di

logging che permettono di attivare/disattivare interamente o parzialmente il tool di monitoring e selezionare le componenti da analizzare. I parametri possibili sono:

- attivazione/disattivazione completa del monitoring;
- lista delle portlet da escludere dal monitoraggio;
- tempo di intervallo di scrittura del file di log;
- attivazione/disattivazione del monitoring avanzato;
- sampling degli utenti da monitorare (viene monitorato il comportamento solo di un campione casuale fra gli $n$ utenti);
- lista degli eventi da catturare;
- sensibilità di movimento del puntatore.

## 3.3 Traduzione da raw data in dati strutturati

L'analisi del comportamento degli utenti durante la loro navigazione in un portale nasce dal recupero dei log dei loro accessi e delle azioni compiute, ma, per arrivare a dati fruibili, è necessaria un'elaborazione di questi in dati strutturati tramite la scoperta e la generalizzazione di comportamenti simili fra i diversi utenti. Lo scopo è quello di riuscire ad ottenere dati dai quali è possibile dedurre i fattori descritti nella sezione 2.2.2 tramite le tecniche illustrate in 2.2.1.

### 3.3.1 Modello dell'utente

La modellazione dell'utente è parte integrante di qualsiasi sistema che fornisce informazioni personalizzate: è quindi anche il caso dei portali web, soggetti principali di questa trattazione. La definizione di un modello dell'utente

può essere descritto come un processo di costruzione delle preferenze personali degli utenti in termini di conoscenza, aspetti comportamentali, obiettivi.

Il modello di un utente in un contesto di contenuti web è genericamente rappresentato nella forma di profilo utente che cattura le preferenze in un formato processabile da una macchina. Ma trattando di sistemi personalizzati, che devono avere quindi un'intenzione adattiva secondo le esigenze dell'utente, il classico profilo utente previsto da un portale può risultare un modello chiuso e restrittivo. È necessaria anche una componente più dinamica in questo modello: la parte esplicita è definita dal profilo utente inizialmente definito al momento della registrazione e manualmente aggiornato (tramite l'editor previsto dal sistema) che cattura informazioni come username, email, indirizzo e competenze dell'utente all'interno del sistema; quest'ultima parte viene intergrata da una parte implicita che comprende il concetto di comportamento dell'utente.
Accanto alla definizione delle caratteristiche dell'utente e delle relazioni fra di essei, è quindi importante individuare:

- il dominio dell'applicazione che si sta analizzando e delinearne i principali concetti e le loro relazioni;

- definire le interazioni dell'utente con il sistema (individuate nei paragrafi precedenti) e quindi il suo comportamento che contribuirà a definire la parte implicita del suo modello: la principale fonte di dati utili per questa analisi è il log ottenuto nella parte precedente, che contiene il percorso di navigazione, le attività dell'utente, il tempo speso su una determinata risorsa.

Come già discusso nella sezione 2.2.1 nelle tecniche di scoperta di pattern di naviagzione, il *clustering* può essere utilizzato per modellare e predire pattern di navigazione di un utente. È così possibile individuare cluster di utenti basandosi sulle sequenze di richieste di pagine effettuate. Si rimanda

il lettore interessato a testi specifici sull'argomento, che propongono diversi algoritmi di *clustering*: [18], [23].

### Caso d'uso di intranet aziendale

Il caso d'uso affrontato nel prossimo capitolo tratterà l'implementazione del tool proposto per il monitoraggio di un portale intranet del Comune di Bologna: gli utente che vi accedono sono quindi dipendenti/impiegati. Individuato quindi il dominio di applicazione è necessario trovare una modellazione adatta per le caratteristiche che può avere un dipendente in ambito lavorativo.
L'ontologia descritta da [6] (EOSK) viene incontro a queste esigenze, proponendo un sistema in grado di modellare concetti di business e le relazioni fra essi in un'azienda. I concetti fondamentali dell'ontologia sono `Position`,

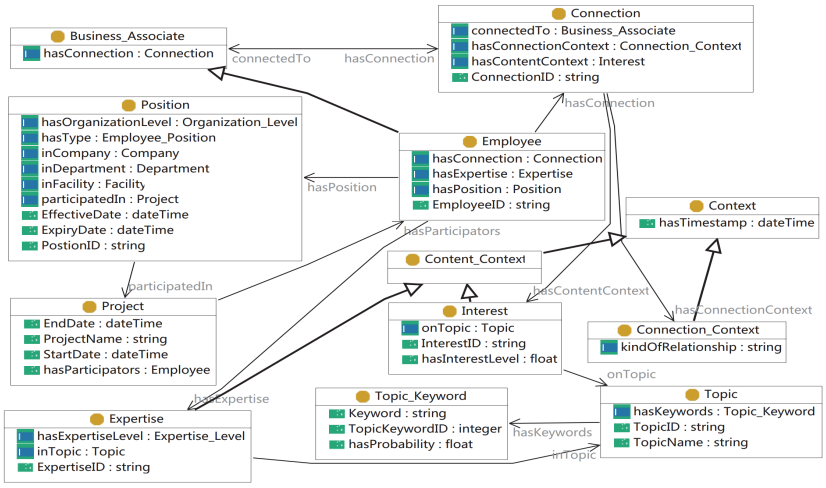


Figura 3.4: Panorama delle classi EOSK

`Expertise`, `Project` a cui ha partecipato e `Conncetion` per i differenti `Context`. Le entità `Topic` e `Expertise` fanno riferimento a informazioni estratte dal servizio di microblogging previsto dal caso d'uso analizzato da [6]. Nel nostro caso potrebbbero essere riutilizzati per indicare le portlet di maggiore interesse (id e nome) o le pagine più visitate (URL). Questo modello si differenzia dal tradizionale *user profiling*, dove le ontologie si focalizzano

su proprietà statiche come i dati personali, concentrando l'utilizzo in contesti dinamici di azienda: i dipendenti hanno una `Position` nell'azienda e ognuno di essi può partecipare a diversi `Project`.

### 3.3.2 Modello di navigazione

Monitornando il comportamento di un utente durante la sua navigazione tra pagine web è necessario formalizzare alcuni concetti finora esposti.

#### Evento

Viene definito un evento di navigazione come una tupla $e = \{l, T, P, t\}$, dove $l$ è l'URL invocato, $T$ è un set di tipo di eventi che qualifica questo evento, $P$ è un set di parametri e $t$ è è l'istante (timestamp). Si denota per semplicità ogni timestamp di un evento come $e_i.t$.

#### Sessione

Si definisce una sessione come una tupla $S = \{s, T_s, T_e, U\}$ dove $s = \langle e_1, e_2, \ldots, e_n \rangle$ è una sequenza ordinata di eventi performati dall'utente $U$, così che $e_i.t \leq e_{i+1}.t$ per ogni $i$, dove $i$ denota l'ordine dell'evento nella sequenza.

Il modello di navigazione di un utente è la visualizzazione di tutti gli accessi dell'utente: la sua rappresentazione può essere quindi il log stesso ottenuto dal monitoring. Dalla figura 3.5 si può notare che:

- una pagina può riferirsi a se stessa (link autoreferenziali);
- si può accedere a una pagina attraverso percorsi alternativi;
- sono previsti anche gli accessi *backward* ("all'indietro").

I contenuti di un portale (portlet e pagine) possono essere categorizzati come segue [3]:

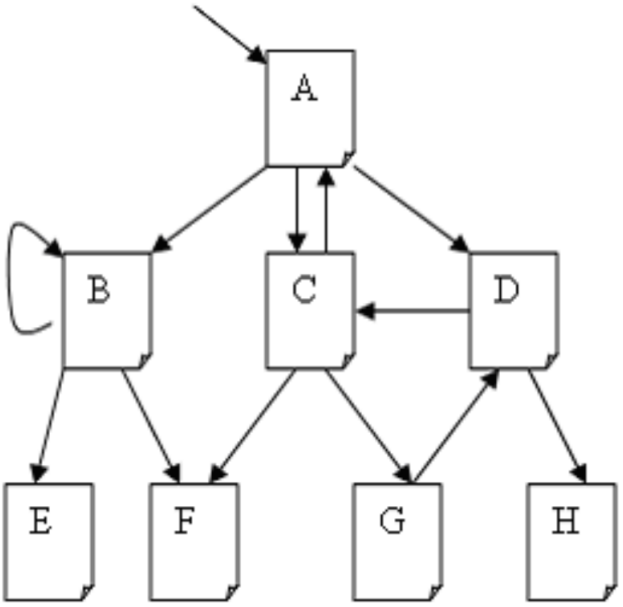


Figura 3.5: Esempio di rappresentazione di una sessione di navigazione

- **EF (*explored and fruitful*)**: questa categoria contiene le risorse che hanno avuto accessi da parte degli utenti e sono probabilmente un contenuto utile e interessante all'utente;
- **ENF (*explored and not fruitful*)**: questa categoria contiene le risorse che hanno avuto accessi da parte degli utenti ma non risultano essere interessanti;
- **UE (*unexplored*)**: queste sono le risorse a cui gli utenti non hanno mai acceduto.

Per decidere in quale categoria porre una risorsa bisogna considerare i parametri che si sono definiti nei paragrafi precedenti mentre si va ad analizzare la sessione di navigazione di un utente: se il TSP è compreso nel range di valori soglia che ne determinano la validità e si è registrata un'attività densa di eventi (movimenti/clic del mouse, scrolling...) la risorsa sarà segnata come *EF* per quel dato utente; nel caso di portlet si può considerare anche la visibilità (vedi paragrafo precedente) come indicatore di interesse. Se invece

il TSP è nettamente inferiore alla soglia minima o eccede di molto rispetto alla soglia massima, e non ci sono eventi registrati che riguardano la risorsa considerata, questa verrà categorizzata come *UE*.

| **Categoria** | **Fattore di enfasi** |
|---|---|
| EF | $\frac{2}{3}$ |
| ENF | $\frac{1}{3}$ |
| UE | 0 |

Tabella 3.1: Fattori di enfasi per le diverse categorie

Il fattore di interesse per una risorsa da parte di un utente può essere quindi definita come funzione di diversi parametri ed è data da

$$\mu = \varphi(fr, d, t_a, k) \tag{3.1}$$

dove $fr$ è la frequenza della risorsa, $d$ è la profondità alla quale si accede alla risorsa, $t_a$ è il TSP e $k$ è il fattore di enfasi corrispondente alla categoria cui appartiene la risorsa 3.1.

## 3.4 Mining sui dati strutturati

Il sistema proposto presenta diversi usi pratici in diversi campi di ricerca del Web, oltre ad integrare le già esistenti metriche. In particolare i fattori di interesse dell'utente che si sono definiti e le analisi della navigazione possono avere un ruolo attivo nel miglioramento della navigazione dell'utente e nella costruzione del portale.

### 3.4.1 Miglioramento del sistema

La performance e altri attributi di qualità del servizio sono cruciali per la soddisfazione dell'utente che deriva dall'utilizzo di servizi Web. L'analisi del

comportamento di navigazione degli utenti fornisce la chiave per comprendere il comportamento del traffico Web, che può essere utilizzato per sviluppare politiche di caching Web (si può pensare di pre-generare i contenuti in base a profili di percorsi di navigazione costruiti a partire dai log di navigazione).

### 3.4.2 Design di un portale

La posizione delle portlet nella pagina di un portale ha una grande importanza poichè alcune portlet possono avere più visibilità di altre. Alcuni studi di *eye tracking* [10, 4] sul comportamento dell'utente nella navigazione web hanno mostrato come l'attenzione dell'utente si concentri principalmente, almeno nelle prime fasi di navigazione, nelle sezioni della pagina posizionate in cima alle prima colonna sulla sinistra. È quindi consigliabile mettere in queste posizioni i contenuti che si vogliono enfatizzare: tramite il sistema appena descritto è possibile individuare le portlet che risultano maggiormente interessanti per un utente e, sulla base di questi dati, questo può aiutare l'amministratore del portale può procedere a posizionarle in modo da darle maggiore enfasi.

### 3.4.3 Personalizzazione del portale

La personalizzazione dei portali è usata di frequente per costruire servizi su misura di un utente o di un gruppo di utenti. In alcuni casi infatti l'utente ha anche la possibilià di scegliere le portlet da posizionare nella sua pagina; in altri casi il sistema descritto in questa trattazione potrebbe venire incontro all'esigenza di generare una pagina di un portale a misura dell'utente, sulla base dei suoi interessi e della sua navigazione precedente, o sulla base della navigazione di utenti che hanno mostrato pattern di navigazioni precedentemente simili. Inoltre i dati e le analisi ottenute dallo strumento di monitoring possono essere utili anche per individuare *cluster* di utenti, sulla base di interessi comuni.

### 3.4.4 Usabilità di portlet e portali

Diaz et al.[7] definiscono l'usabilità di una portlet come la capacità di una portlet di essere compresa, personalizzata o usata sotto specifiche condizioni. Gli indicatori di interesse descritti precedentemente possono aiutare nell'analisi di usabilità di una portlet.
In caso di portlet con simili caratteristiche e funzioni, il tool di monitoring può essere utile per isolare le interazioni relative a una determinata portlet per valutare la relativa usabilità.
Anche la posizione di una portlet influisce sull'usabilità del portale: supponiamo ad esempio che un'azione da eseguire necessiti l'interazione con più portlet. La poszione delle portlet coinvolte possono influire sul tempo necessario ad eseguire quel determinato compito: uno studio sull'usabilità può avere l'obiettivo di trovare la posizione migliore, analizzando i comportamenti di diversi utenti che interagiscono con le diverse configurazioni di portlet in una o più pagine. In generale l'osservare gli eventi di interazione fra portale e utente, può evidenziare se ci sono problemi da parte degli utenti nell'utilizzo del portale (ma questo può valere per un sito web generico).

# Capitolo 4

# Implementazione e caso di studio

Lo sviluppo di parte del sistema modellato nel capitolo precedente iniziato è con il tirocinio svolto presso Engineering Ingegneria Informatica nel mese di Ottobre 2014. Il caso d'uso affrontato si riferisce a un portale web intranet del Comune di Bologna.


Figura 4.1: Il portale intranet IONOI

L'intranet comunale IONOI privilegia la comunicazione, ossia la diffusione

di informazioni e contenuti sulla Intranet; privilegia la partecipazione, cioè il contributo attivo dei fruitori della Intranet, ossia i dipendenti comunali; infine privilegia l'integrazione, ossia l'utilizzo della Intranet come strumento di supporto tanto ai processi di comunicazione interna, quanto alle altre dinamiche di lavoro organizzativo. I servizi offerti dal portale sono diversi: motore di ricerca integrato, notizie dal mondo, webmail e agenda aziendali integrate, servizi di prenotazione auto e sale comunali, servizi di chat in tempo reale, video conferenza e spazi per team di progetto.

Da intranet intese come classici sistemi di gestione documentale in cui le informazioni vengono elaborate da redazioni od organi istituzionali e veicolate verso la base di utenti, oggi si parla sempre più spesso di architetture flessibili di *knowledge management* in cui il sapere viene prodotto e codificato dagli stessi utilizzatori finali che, a seconda delle specifiche competenze, permettono ai colleghi di accedere a informazioni trasmesse generalmente solo per via informale.
Proprio dagli dipendenti utenti del portale è nata la necessità di poter avere un resoconto delle performance del portale a cui accedono: il tool che si è andato a sviluppare si propone non solo di monitorare il rendering delle pagine o delle semplici richieste HTTP (cosa già realizzata da altri tool a disposizione), ma delle singole portlet che compongono la pagina e che possono differire dal tipo di utente cui vi accede. Questo tipo di monitoraggio più approfondito può permettere all'azienda di effettuare scelte in termini di struttura del portale ai fini di rendere più efficace e performante la navigazione del portale per ogni tipologia di utente.

## 4.1 La piattaforma Liferay Portal

Liferay [1]è la piattaforma di riferimento per tutti i portali web del Comune di Bologna seguiti da Engineering: è un'applicazione open source basata sul

[1]`https://www.liferay.com/`

linguaggio Java, indirizzata alla creazione e alla gestione di portali web e, di conseguenza, al controllo delle portlet.

Utilizzato da aziende in tutto il mondo, Liferay comprende una lunga lista di funzionalità che lo mettono a confronto diretto con molti portali commerciali, con il vantaggio di non avere oneri di licenza. Infatti, oltre ad una gestione ottimale delle portlet, il suo successo è dovuto alla quantità (e qualità) dei servizi integrati, un'ottima flessibilità di utilizzo e una grande capacità di organizzare e supportare la collaborazione interna.
L'applicazione fornisce un'interfaccia web unificata per le varie informazioni e gli strumenti che provengono da diverse sorgenti. All'interno del portale, come si è visto, questa interfaccia è composta da un certo numero di portlet. Dal momento che queste ultime sono sviluppate indipendentemente dal portale e che quindi sono scarsamente accoppiate con esso, si può parlare di una piattaforma basata su un'architettura SOA, della quale si parlerà più dettagliatamente in seguito.
Liferay presenta una vasta gamma di portlet liberamente disponibili per blog, forum, sondaggi, calendario, raccolta documenti, galleria immagini, feed RSS, wiki, contenuti web e così via. La piattafroma consiste di tre parti: un kernel (Liferay Portal), che funge da base per le applicazioni e i contenuti, un content management system (Liferay CMS) e una suite di applicazioni per realizzare collaboration e social networks (Liferay Collaboration).
Le portlet sono gestite da un portlet container ed un portal server, che è responsabile di servire le pagine al browser. Insieme questi due sistemi costituiscono l'infrastruttura del portale necessaria al deployment delle applicazioni.

### 4.1.1 Standard Portlet Container - Portlet

La tecnologia dei portlet e dei portali utilizza un insieme di standard allo scopo di consentire lo sviluppo di portlet portabili, ovvero che possano essere usati nel contesto di portali sviluppati con tecnologie differenti.

### JSR-168 e JSR-286

Per consentire la portabilità tra i vari portlet container commerciali, è stata fatta richiesta al Java Community Process (JCP) di formalizzare uno standard apposito per le portlet, lo standard JSR-168 (Java Portlet Specification 1.0). Quest'ultimo è stato rilasciato nel 2003 e al gruppo di lavoro hanno partecipato diverse importanti aziende tra le quali Apache Software Foundation, IBM, Sun Microsystems, Oracle.
La specifica ha definito un insieme di interfacce applicative (API) per l'interoperabilità fra un portlet container e le portlet. In particolare, ha fissato i seguenti punti:

- il ruolo e le funzionalità dei portlet container;
- il contratto tra il container e le portlet;
- la gestione del ciclo di vita delle portlet;
- il packaging per la distribuzione delle portlet;
- l'interazione con lo standard WSPR (Web Services for Remote Portlets) che definisce un protocollo standard per il dialogo tra il container e le portlet.

La specifica JSR-168 [14] presenta però diverse limitazioni (soprattutto sul lato della comunicazione inter-portlet) che hanno portato all'introduzione dello standard JSR-286 (Java Portlet Specification 2.0) [15], rilasciato nel 2008. Le maggiori novità introdotte comprendono la capacità di una portlet di ricevere ed inivare eventi, la possibilità per le portlet di condividere parametri tra loro, attraverso i *public render parameter* e i *portlet filter* (componente Java che può essere usato per mdoificare le portlet request e response prima/dopo l'accesso alla portlet desiderata).

#### WSPR

L'obiettivo dello standard WSRP (Web Services for Remote Portlets) [28] è ottenere l'interscambio dei contenuti tra portali i quali, ignorando le differenze di piattaforma e di programmazione, potranno importare ed esportare tipi etereongei di risorse da e verso qualunque sito compatibile affiliato.
WSRP è stato progettato per aggregare i contenuti dei portali in modo standard: i portali che implementano questo protocollo potranno accedere, visualizzare e utilizzare risorse (come le portlet) che risiedono su portali remoti.

La specifica JSR-168 e lo standard WSRP non sono in competizione poiché riguardano aspetti differenti delle funzionalità messe a disposizione dalle portlet. La specifica JSR-168 è una tecnologia specifica Java che mette a disposizione delle Portlet API, progettate per garantire interoperabilità tra le portlet e portlet container. Queste portlet sono locali al container il quale, provvederà alla loro gestione.
Il WSRP invece è un protocollo che permette di accedere a portlet remote in modo standard e mette a disposizione una piattaforma base per avere interoperabilità nella pubblicazione e consumo delle portlet remote.

### 4.1.2 Architettura SOA

L'aspetto più importante di ogni portale è senza dubbio l'architettura sottostante. Quella di Liferay, oltre ad essere adatta per le normali applicazioni, supporta la massima disponibilità per sistemi di tipo mission-critical. Infatti, è in grado di bilanciare una solida struttura che garantisce l'implementazione dei principali standard dei portali con il valore e gli standard messi a disposizione dai maggiori open frameworks. La figura 4.2 mostra i diversi strati dell'architettura e le funzionalità delle portlet.
La piattaforma è completamente basata sui principi di progettazione che fanno riferimento alla Service Oriented Architecture (SOA). Vi è infatti la necessità di avere un modo standard per esporre un software su una rete at-

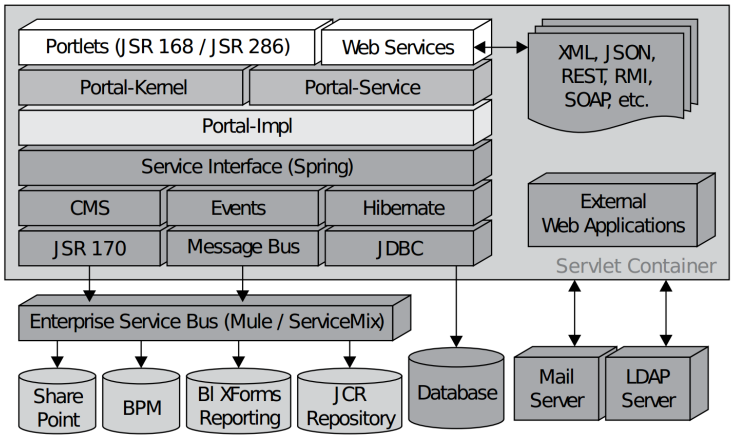


Figura 4.2: Architettura di Liferay

traverso un'interfaccia comprensibile da tutte quelle aziende che, volendo far interagire le proprie applicazioni con quelle di altre compagnie, riconoscono tale standard. SOA è un modello architetturale per la creazione di sistemi residenti su una rete che focalizza l'attenzione sul concetto di servizio. Un sistema costruito seguendo questa filosofia è costituito da applicazioni, chiamate appunto servizi, ben definite ed indipendenti l'una dall'altra, che risiedono su più computer all'interno di una rete (ad esempio la rete interna di un'azienda o una rete di connessione fra più aziende che collaborano: intracompany e intercompany network). Ogni servizio mette a disposizione una certa funzionalità e può utilizzare quelle che gli altri servizi hanno reso disponibili, realizzando, in tal modo, applicazioni di maggiore complessità. L'architettura SOA, che si può considerare quindi come una forma particolare di sistema distribuito, presenta alcune caratteristiche e proprietà orientate al riutilizzo e all'integrazione in un ambiente eterogeneo che devono essere rispettate dai servizi. In particolare un servizio dovrà:

- essere ricercabile in base alla sua interfaccia e recuperabile dinamicamente a tempo di esecuzione;

- essere ben definito, completo ed indipendente dal contesto o dallo stato

di altri servizi;

- essere definito da un'interfaccia ed indipendente dall'implementazione;
- essere debolmente accoppiato con altri servizi;
- essere reso disponibile sulla rete tramite la pubblicazione della sua interfaccia ed accessibile in modo trasparente rispetto alla sua allocazione;
- fornire un'interfaccia possibilmente a "grana grossa", con poche funzionalità in modo tale da non dover avere un programma di controllo complesso;
- essere realizzato in modo tale da permetterne la composizione con altri.

### 4.1.3 Modello degli utenti

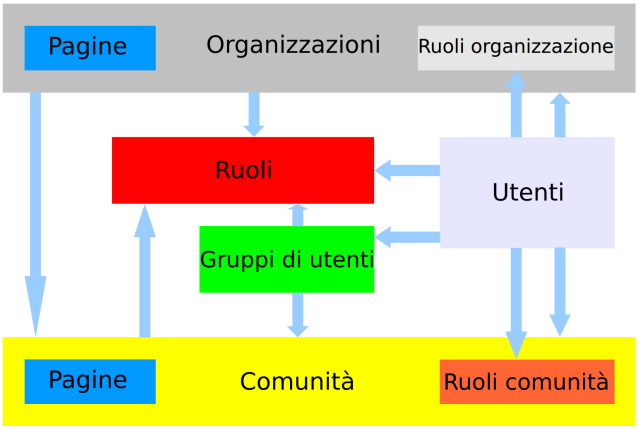


Figura 4.3: Architettura degli utenti in Liferay

In Liferay è prevista la seguente architettura nell'amministrazione degli utenti:

- USERS: rappresenta gli utenti che accedono al portale; prima di autenticarsi, ciascun utente è definito come guest ed ha un accesso limitato alle funzionalità implementate. Una volta autenticato, il client può

avere determinati privilegi a seconda della comunità a cui appartiene oppure in base all'organizzazione di cui fa parte;

- USER GROUPS: indica un insieme di utenti che condividono una serie di permessi e ruoli, gestiti dall'amministratore stesso;
- ROLES: individua i ruoli, ossia tutti quei permessi relativi all'intero portale;
- ORGANIZATIONS: raggruppa alcuni utenti strutturando il tutto in modo gerarchico;
- ORGANIZATION ROLES: specifica i ruoli relativi ad una determinata Organizzazione;
- COMMUNITIES: riunisce in gruppi gli utenti che hanno in comune un determinato interesse; questi gruppi possono essere di tre tipi:
  1. Open: permette ad un utente di entrare o uscire dalla comunità in qualsiasi momento autonomamente;
  2. Restricted: subordina l'inserimento di un utente in una comunità al controllo e alla valutazione dell'amministratore della stessa;
  3. Hidden: rappresenta una tipologia di comunità simile alla Restricted, con l'aggiunta del fatto che non è possibile visualizzarla attraverso i classici strumenti messi a disposizione dal portale;
- COMMUNITY ROLES: determina i ruoli relativi ad una particolare comunità.

La differenza strutturale tra le organizzazioni e le comunità è che le prime hanno una struttura gerarchica e un utente, una volta registrato, può far parte di molte comunità ma può essere inserito solamente in una organizzazione. Il motivo principale per il quale sono state distinte queste due categorie è che con le organizzazioni si vuole rappresentare nel miglior modo possibile il modello realmente esistente riferito all'associazione o all'ente per

cui si vuole sviluppare il portale, mentre con le comunità si vuole raffigurare un insieme di attività comuni a più utenti, che possono appartenere a diverse organizzazioni o addirittura a nessuna di queste.

## 4.2 Tool di monitoring di raw data

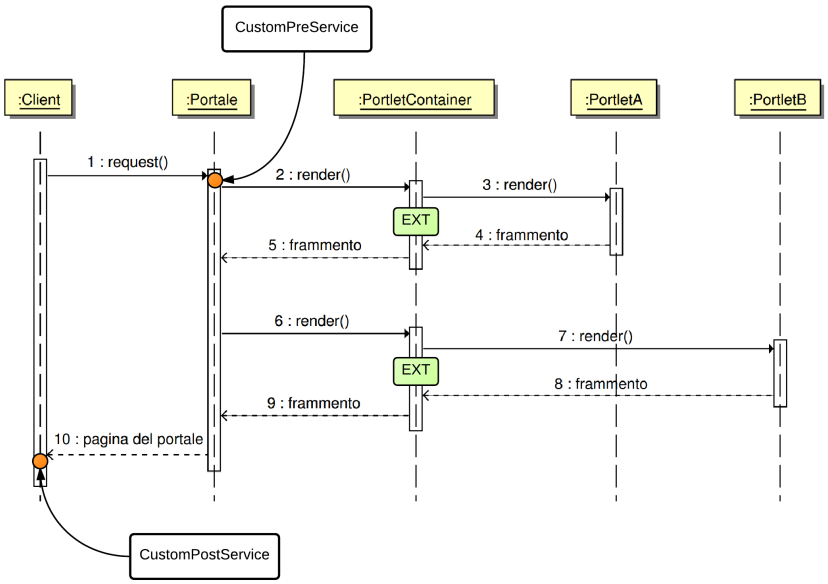


Figura 4.4: Struttura del tool

Del modello descritto nel capitolo precedente è stata implementata solo la parte iniziale relativa al monitoring degli accessi. Si è sviluppato quindi un componente che cattura tutte le richieste di pagina e le informazioni ad esse associate: l'identificativo della pagina, chi la richiede, le componenti della pagina e i tempi di caricamento di tutti gli elementi. Per ottenere queste informazioni si sono dovuti sovrascrivere i servizi richiamati a inizio e fine richiesta di pagina (a livello di plugin hook), mentre per ottenere informazioni a livello portlet si è dovuto intervenire sul *core* di Liferay (tramite l'ambiente Ext).

### 4.2.1 Tecnologie utlizzate

Caratteristica di Liferay è il supporto a più Servlet Container (Apache TomCat, JBoss, GlassFish) e il supporto a gran parte degli odierni database (MySql, PostgresSQL, Oracle, SQL Server).
Per l'implementazione del tool stati utlizzati:

- Liferay Portal 5.2.3 e relativa SDK
- JBoss 4.3
- MySql 5.5
- Java 1.6
- Apache Ant

Il tool si occuperà quindi di raccogliere i seguenti dati:

- timestamp dell'evento;
- username dell'utente che ha effettuato la richiesta (se l'utente non è autenticato verrà segnato come *guest*);
- *plid*, ovvero l'identificativo della pagina richiesta (in Liferay chiamate anche layout);
- URL della pagina;
- tempo di caricamento della pagina;
- portlet presenti nella pagina con rispettivo id, nome e tempo di rendering.

L'attività ha richiesto inizialmente lo studio della piattaforma utilizzata, la sua struttura e il suo funzionamento, in particolare il rendering di una pagina, dovendo andare a intervenire in diverse fasi del ciclo di caricamento della pagina del portale (figura 4.5).

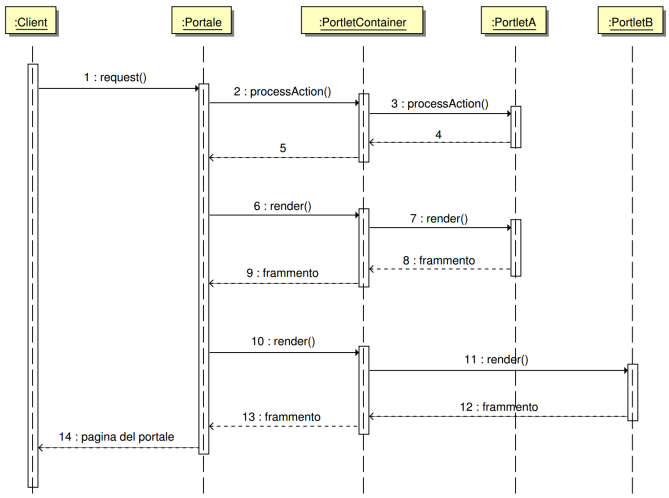


Figura 4.5: Caricamento di una pagina del portale

Le richeste arrivano tramite `com.liferay.portal.servlet.MainServlet` e diversi attributi sono salvati nella riciesta e nella sessione; una volta identificato il layout richiesto, questo viene gestito da `com.liferay.portal.action.LayoutAction`; per ogni portlet, il metodo `com.liferay.portal.util.PortalUtil.renderPortlet()` viene chiamato, il quale chiama a sua volta `/portal/portal-web/docroot/html/portal/render_page.jsp` per caricare il contenuto di ogni singola portlet.

Un portale costruito con Liferay può essere esteso secondo i seguenti tre livelli di sviluppo:

- ambiente *Plugins SDK*;
- ambiente *Extension*;
- codice sorgente di Liferay Portal.

Per gli scopi del componente che si è andato a sviluppare è stato necessario utilizzare due diversi componenti della libreria di Liferay: hook e ambiente Ext.

### 4.2.2 Hook

Gli hook sono il tipo di plugin più nuovo che Liferay Portal supporta: sono stati introdotti dalla versione 5.1.x: come le portelt, i template dei layout e i temi, essi sono creati tramite la SDK dei plugin.
L'ambiente SDK consente quindi di creare plugin hot-deployable per il portale Liferay attraverso l'invocazione di task di Ant forniti insieme all'SDK, che permettono di automatizzare attività (come ad esempio la generazione automatica della struttura del progetto o il deploy sul server) senza dover riavviare l'application server. Dunque, vengono sviluppati i plugin e successivamente vengono usati tali task per costituire dei file WAR che vengono direttamente copiati in un'apposita cartella di Auto Deploy. In seguito, il portale, assieme all'application server, individua ogni archivio WAR presente nella directory di hot-deploy ed estrae automaticamente i file al loro interno nella cartella di deployment del server.

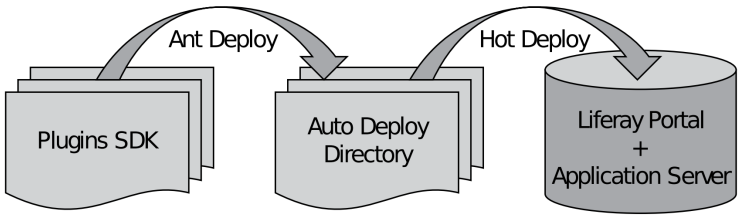


Figura 4.6: Funzionamento dell'ambiente Plugins SDK

Gli hook consentono agli sviluppatori di sovrascrivere parti del core di Liferay con le proprie implementazioni, mantenendo una netta separazione tra il codice personalizzato e il core di Liferay; inoltre sono hot-deployable, quindi aggiungere e rimuovere è un'operazione semplice. La possibilità di suddividere funzionalità complesse in più hook, sviluppati da persone diverse, rende dinamico l'ambiente di sviluppo. Gli hooks sono stati progettati per superare molte delle limitazioni dell'Ext e i punti di estensione disponibili crescono a ogni release di Liferay anche grazie ai feedback ricevuti da parte

degli utenti.

Per lo scopo di ottenere informazioni sulle diverse richeste effettuate dagli utenti durante la navigazione, è necessario intervenire quindi subito prima del caricamento della pagina e a rendering concluso. Ciò è possibile tramite gli hook, che prevedono di poter eseguire azioni personalizzate al compiersi di alcuni eventi del portale. Si è andata a sovrascrivere la parte che intercetta la richiesta del layout (la pagina secondo Liferay): in particolare si è intervenuto sulle classi `ServicePreAction` e `ServicePostAction`, richiamate rispettivamente appena il layout da visualizzare è stato determinato e appena il rendering della pagina si è concluso.
Per comunicare a Liferay che si devono andare a sovrascrivere i servizi `servlet.service.events.pre` e `servlet.service.events.post` bisogna inserire le seguenti istruzioni

```
servlet.service.events.pre=
    com.liferay.monitorhook.CustomServicePreAction
servlet.service.events.post=
    com.liferay.monitorhook.CustomServicePostAction
```

nel file `/docroot/WEB-INF/src/portal.properties` contenuto nella cartella di sviluppo del componente hook.
Nella `ServicePreAction` vengono raccolti i dati riguardanti l'utente (username) che ha richiesto la pagina, l'id (*plid*) relativo al layout e l'URL della pagina; il timestamp salvato in questo momento viene passato come parametro della richiesta HTTP per essere poi confrontato con il timestamp ottenuto alla fine del caricamento della pagina. Inoltre viene ottenuta la lista di tutte le portlet presenti nel layout richiesto e anch'essa settata come attributo della richiesta HTTP (sarà utile successivamente per compilare il log).

Listato 4.1: CustomPreService

```
public class CustomServicePreAction extends Action {
```

```
private String filePath="/log/log.xml";
private CheckConfig config = new CheckConfig();
static int interval;
private Date now;
private Date start = new Date();
private static long MAX_INTERVAL = 20*60*1000;

@Override
public void run(HttpServletRequest request, HttpServletResponse
    response)
      throws ActionException {

   now=new Date();

   if (now.getTime()-start.getTime() > MAX_INTERVAL){
      config = new CheckConfig();
      config.check();
      start = new Date();
   }

   if(config.isLog_enabled()){
      File file = new File(filePath);
       if(!file.exists()){
         try {
            file.createNewFile();
            CreateXML.createXMLDoc();
         } catch (IOException e) {
            e.printStackTrace();
         }
       }

      try {
```

```
            servicePre(request, response);
         }
         catch (Exception e) {
            throw new ActionException(e);
         }
      }
   }

   protected void servicePre(HttpServletRequest request,
         HttpServletResponse response) {

      [...]
      //salvo come parametro il timestamp di invio richiesta
      request.setAttribute("startOfRequest", now);

      if (selectedLayout != null){
         LayoutTypePortlet layoutTypePortlet =
             (LayoutTypePortlet)selectedLayout.getLayoutType();
         List actualPortletList = null;
         try {
            actualPortletList = layoutTypePortlet.getAllPortlets();
         } catch (SystemException e1) {
            e1.printStackTrace();
         }
         request.setAttribute("list", actualPortletList);
      }
      [...]
   }
}
```

Al termine del rendering della pagina nella `ServicePostAction` viene calcolato il tempo richiesto tramite una differenza di timestamp (il timestamp di avvio della richiesta viene recuperato come parametro nella richiesta stessa);

inoltre vengono estratti anche tutti i parametri relativi ai tempi di caricamento delle singole portlet (calcolati nel componente che analizzeremo nella sezione succesiva).

Listato 4.2: CustomPostService

```
public class CustomServicePostAction extends Action{

   private CheckConfig config = new CheckConfig();

   @Override
   public void run(HttpServletRequest request, HttpServletResponse
       response)
         throws ActionException {
      if(config.isLog_enabled()){
         [..]
         //estraggo le portlet relative al layout
         List<Portlet> portlets = (List)
             request.getAttribute("list");
         for (Portlet x : portlets){
            String name = x.getDisplayName();
            //estraggo il tempo di rendering relativo alla portlet
                considerata
            String timeP = (String)
                request.getAttribute(name+"_time");
            CreateXML.appendPortletTimes(name, timeP);
         }
         if (startOfRequest != null){
            //calcolo il tempo di caricamento della pagina
            long time = now.getTime() - startOfRequest.getTime();
            System.out.println("Request: " + time + "ms");
            //aggiungo il valore al buffer di log
            CreateXML.appendTime(String.valueOf(time));
```

```
[..]
```

### 4.2.3 EXT Environment

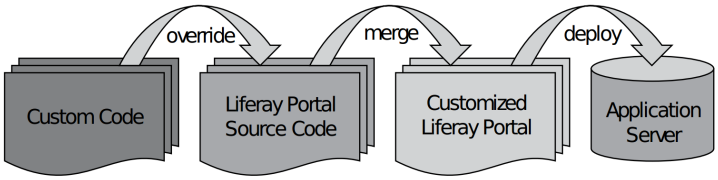


Figura 4.7: Funzionamento dell'ambiente EXT

Per quanto riguarda l'analisi dei tempi di rendering delle singole portlet che compongono una pagina si è dovuto intervenire sul *core* di Liferay ed è stato quindi necessario l'utilizzo dell'ambiente Ext.
L'ambiente Extension (EXT) permette di personalizzare completamente Liferay Portal. Infatti, estende l'ambiente di sviluppo del portale, avendo la possibilità di agire sui file di configurazione del portale e sul codice Java e JSP. È quindi un wrapper per i sorgenti del *core* di Liferay (per questo motivo la struttura delle cartelle in Ext rispecchia la gerarchia delle directory dei sorgenti, cioè `ext-impl/`, `ext-service/` e `ext-web/`): il codice sorgente non viene modificato in alcun modo, in quanto tutte le personalizzazioni sono organizzate in un progetto separato con l'obiettivo di agevolare l'upgrade a nuove versioni di Liferay in cui inserire le personalizzazioni realizzate per le precedenti versioni.
Come illustrato nella figura 4.7, il funzionamento dell'ambiente di estensione può essere così riassunto:

- il custom code sovrascrive (*override*) il codice sorgente del portale nell'ambiente Ext;
- prima del processo di deploy il custom code viene fuso (*merge*) con il codice sorgente del portale sempre nell'ambiente Ext; il risultato è

una versione custom del portale Liferay ("Customized Liferay Portal") costruita in ambiente Ext;

- del portale Liferay così personalizzato viene quindi effettuato il deploy da Ext sull'application server.

Il metodo *render()*, richiamato al momento del caricamento del contenuto di ogni singola portlet, si trova all'interno dell'implementazione della classe `InvokerPortlet` (package `com.liferay.portlet`) contenuta nella cartella `portal-impl`: è qui che ogni evento/azione che coinvolge le portlet viene gestita nella maniera opportuna a seconda del tipo. Per modificare questa classe quindi lo sviluppo si è svolto nella rispettiva cartella nell'ambiente EXT (`ext-impl/`).
Anche in questo caso viene calcolato il tempo con la differenza di due timestamp posti nel metodo che si occupa del rendering della singola portlet. Insieme a questo vengono salvati anche id e nome della portlet, ai fini di rendere il log il più comprensibile possibile, come parametri della richiesta HTTP. Questi dati vengono recuperati dal `ServicePostAction` come visto nella sezione precedente.

Listato 4.3: InvokerPortletImpl

```
public class InvokerPortletImpl implements InvokerPortlet{
   [..]
   public void render(
        RenderRequest renderRequest, RenderResponse renderResponse)
      throws IOException, PortletException {

      PortletException portletException =
         (PortletException)renderRequest.getAttribute(
            _portletId + PortletException.class.getName());

      if (portletException != null) {
         throw portletException;
      }
```

```
StopWatch stopWatch = new StopWatch();
stopWatch.start();
long time = new Date().getTime();
Timestamp ts = new Timestamp(time);
System.out.println("Start rendering for " + _portletId +": "+
    ts);

[...]

String _portletTime = String.valueOf(renderTime);

HttpServletRequest httpServletRequest =
    PortalUtil.getHttpServletRequest(renderRequest);
httpServletRequest.setAttribute(_portletName+"_time",
    _portletTime);

[...]
```

### 4.2.4 Output di log XML

Tutti i dati raccolti vengono salvati in formato XML, dove ogni entry conterrà i dati relativi alla pagina e a tutte le portlet in essa contenuta. La scrittura su file non viene eseguita ad ogni richiesta di pagina ma periodicamente viene scritto ciò che è stato salvato in un buffer di appoggio. Il monitoraggio può essere anche disabilitato/abilitato tramite un file di configurazione che periodicamente viene controllato.
Le funzioni di controllo del file di configurazione sono svolte dalla classe `CheckConfig`: tramite un parser SAX viene verificato il contenuto. In questo caso si è reso configurabile solo l'attivazione o disattivazione del monitoring, ma sarebbe possibile anche elencare i nomi delle portlet di cui non si vuole monitorare il caricamento.

Listato 4.4: File di configurazione

```
<config>
   <log>true</log>
</config>
```

La gestione della scrittura del file di log è a carico della classe `CreateXML`: una volta ricavati i dati nella `ServicePreAction` o `ServicePostAction` viene richiamato il metodo opportuno (per creare un nuovo nodo o aggiungere solo elementi a un nodo già esistente). La scrittura su file non viene effettuata a ogni chiamata di metodo ma periodicamente secondo l'intervallo settato (variabile `interval`): prima della reale scrittura su file, i dati vengono salvati in un buffer (`StringWriter`).

Listato 4.5: CreateXML

```
public class CreateXML {

   final static String filePath = "/log/log.xml";
   static File file = new File (filePath);
   static StringWriter buffer;
   static int interval;
   static DocumentBuilderFactory docFactory = DocumentBuilderFactory
          .newInstance();
   static TransformerFactory tf = TransformerFactory.newInstance();

   [...]
   public static synchronized void createElement(String username,
      String plid, String url){
      try {
         DocumentBuilder docBuilder =
             docFactory.newDocumentBuilder();
         Document doc = null;
         if (buffer != null){
            String xml = buffer.toString();
```

```
            doc = docBuilder.parse(new InputSource(new
                ByteArrayInputStream(xml.getBytes("utf-8"))));
        } else {
            doc = docBuilder.parse(file);
            buffer = new StringWriter();
        }
        Node root = doc.getFirstChild();
        Element entry = doc.createElement("entry");
        root.appendChild(entry);

        Attr attr = doc.createAttribute("date");
        attr.setValue(new Date().toString());
        entry.setAttributeNode(attr);

        Element user = doc.createElement("user");
        user.appendChild(doc.createTextNode(username));
        entry.appendChild(user);

        Element id = doc.createElement("plid");
        id.appendChild(doc.createTextNode(plid));
        entry.appendChild(id);

        Element eUrl = doc.createElement("url");
        eUrl.appendChild(doc.createTextNode(url));
        entry.appendChild(eUrl);

        buffer = new StringWriter();
        convertDocument(doc);

    } catch (SAXException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
```

```
        } catch (ParserConfigurationException e) {
          e.printStackTrace();
        }
    }
    [...]
    private synchronized static void writeDoc() {
        DocumentBuilder builder;
        try {
          builder = docFactory.newDocumentBuilder();
          String xml = buffer.toString();
          Document doc = builder.parse(new InputSource(new
              ByteArrayInputStream(xml.getBytes("utf-8"))));
          Transformer transformer = tf.newTransformer(new
              StreamSource("/home/cecilia/prettyprint.xsl"));
          DOMSource domSource = new DOMSource(doc);
          StreamResult streamResult = new StreamResult(file);
          transformer.transform(domSource, streamResult);
        } catch (Exception e) {
              e.printStackTrace();
        }
    }

    [...]
```

Un esempio di log ottenuto si può vedere in figura 4.6.

Listato 4.6: File di log

```
<entry date="Mon Nov 10 16:55:36 GMT 2014">
   <user>5</user>
   <plid>10173</plid>
   <url>/web/guest/home</url>
   <portlet name="Journal Content">10</portlet>
   <portlet name="Journal Content">20</portlet>
   <portlet name="Login">3</portlet>
```

```
   <timeLayout>37</timeLayout>
</entry>
<entry date="Mon Nov 10 16:55:38 GMT 2014">
   <user>john</user>
   <plid>10705</plid>
   <url>/web/guest/uno</url>
   <portlet name="Dictionary">54</portlet>
   <portlet name="Language">17</portlet>
   <timeLayout>104</timeLayout>
</entry>
```

L'elemento `entry` con il rispettivo attributo `date` rappresenta ogni accesso a un singolo layout, in esso sono infatti contenuti:

- l'elemento `user` che contiene lo username dell'utente che ha richiesto il layout (se l'utente non è autenticato verrà segnato con il valore di default "5 di Liferay);
- l'elemento `plid`: l'identificativo del layout;
- l'elemento `url` che contiene il *friendly URL*;
- un elemento `portlet` per ogni componente che appartiene al layout considerato, con il rispettivo attributo che indica il nome (*display name*, il nome della portlet visualizzato durante la navigazione) e il cui contenuto è il tempo di rendering della portlet in millisecondi;
- l'elemento `timeLayout`: indica il tempo di caricamento della pagina in millisecondi.

## 4.3 Valutazione

Non è stato possibile mettere in produzione ed effettuare test completi sul componente di logging che si è sviluppato. I test in locale che sono stati eseguiti hanno però avuto lo scopo di verificare la corretta estrazione dei dati

necessari.
Soprattutto per quanto riguarda lo sviluppo della parte nell'ambiente EXT è stato necessario un uso massiccio del debugger per andare a studiare il ciclo di rendering di una pagina e riuscire a intervenire quindi nel punto giusto del codice sorgente di Liferay: come supporto di verifica del log, soprattutto nelle prime fasi di sviluppo, tutti i valori estratti e i timestamp di ogni evento venivano stampati sulla console di JBoss (figura 4.8), anche per verificare la congruenza dei tempi di rendering estratti.

```
[STDOUT] End request: 2014-12-07 23:02:18.412
[STDOUT] Request: 91ms
[STDOUT] User: john
[STDOUT] Layout id: 10666
[STDOUT] Url: /web/guest/notizie
[STDOUT] Start request: 2014-12-07 23:02:18.576
[STDOUT] Start rendering for 39_INSTANCE_wqG7: 2014-
```

Figura 4.8: Esempio di verifca delle informazioni estratte

Una volta verificato di essere riuscita ad ottenere il set di *raw data* corretti, si è proceduto con lo sviluppo del componente di scrittura del file XML di log. Test minimi sono stati effettuati per verificare la corretta gestione dell'accesso concorrente al file di log, ma sicuramente sono necessari la messa in produzione e diversi giorni di test con un bacino di utenti maggiore, per evidenziare eventuali bug del componente e verficarne l'impatto sul sistema anche a livello di prestazioni.
Sono stati effettuati test sulla mia macchina personale, simulando accessi a tre pagine differenti di prova (figura 4.9), ognuna formata da due o tre portlet ciascuna (le portlet utilizzate sono quelle messe a disposizone da Liferay di default). Come output sono stati prodotti due file di log XML da circa 15 KB ciasuno, per un totale di circa 230 accessi monitorati.

La scelta di avere una struttura modulare del framework EOP viene in-

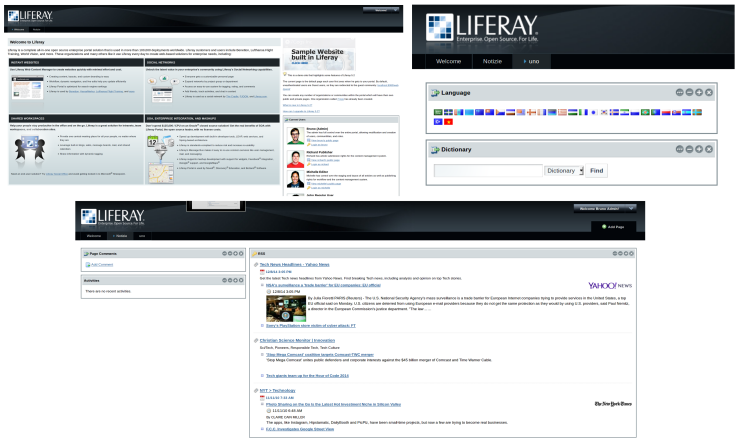


Figura 4.9: Esempi di layout su cui sono stati effettuati i test

contro alla necessità di avere uno strumento non troppo invasivo durante la navigazione dell'utente, e questo conferisce numerosi vantaggi: la modularità rende EOP facilmente modificabile ed estendibile (con l'integrazione di altri moduli); ne facilita il riutilizzo, in una sua parte o nella sua interezza, e consente di ottenere dati di facile comprensione e che possono essere significativi poi per diversi scopi.
In questa trattazione si è cercato di suggerire la direzione verso la quale dovrebbe andare il monitoring di un portale Web, proponendo modelli da seguire e fattori da considerare durante la navigazione.

## 4.4 Sviluppi futuri

In questa trattazione non si è sviluppata l'implementazione di EOP in tutte le sue fasi. Esistono comunque già diverse soluzioni e strumenti che possono aiutare nella direzione proposta.

La scelta di gestire i file di configurazione nel formato XML porta numerosi vantaggi: è una soluzione flessibile (elementi e attributi vengono scelti ad hoc per creare la struttura più efficiente per le applicazioni) e portabile,

dato il livello superiore di astrazione rispetto ad altre soluzioni. Così nel file di configurazione si può scegliere facilmente di inserire diversi parametri, come le portlet da escludere dal monitoraggio o le azioni compiute dall'utente da monitorare. Questo permette di creare uno strumento di monitoraggio a misura secondo le nostre esigenze.
Se da un lato si hanno vantaggi per la scelta di XML per un file di configurazione che non raggiungerà mai dimensioni molto elevate, per quanto riguarda la scelta di XML per i file di log, potrebbero invece emergere fattori negativi. XML infatti è sì uno standard semplice ed estendibile che garantisce interoperabilità, ma questo causa anche una discreta verbosità e un problema di occupazione della memoria. Ad esempio un altro linguaggio di *self-describing data* con gli stessi vantaggi di semplicità (anche in termini maggiori) e compatibilità è JSON (JavaScript Object Notation) [2]. In particolare, dovendo effettuare un monitoring client-side, si presuppone l'utilizzo di Javascript e JSON potrebbe essere una soluzione migliore in quanto salva i dati in array e record (mentre XML in alberi): i trasferimenti di dati potrebbero essere più facili avendoli salvati in strutture che sono familiari ai linguaggi *object-oriented*. Inoltre JSON sta cominciando ad essere sempre di più il formato usato da diversi database NoSQL (MongoDB, CouchDB).

Per il monitoring avanzato del comportamento dell'utente, e quindi delle interazioni con il portale e le sue componenti nel dettaglio, si può sfruttare la tecnologia Javascript sul lato client. Una soluzione disponibile è *isOnScreen* (`https://github.com/moagrius/isOnScreen`), un plugin jQuery che determina se un elemento è visibile nello schermo o meno; è possibile anche configurare dei parametri di soglia per determinare la visibilità sopra o sotto una certa percentuale.
Altri dati come coordinate di una portlet (e l'oggetto HTML corrispondente) e il tipo di azione eseguito da un utente sono facilmente estraibili sul lato client.

[2] `http://json.org/json-it.html`

Dell'analisi proposta da [6] che si sofferma sulla visibilità parziale di una portlet (indicatore di interesse dell'utente spiegato nella sezione 2.2), proponendo un indicatore per quantificare la percentuale di visibilità, di seguito si vede la procedura in pseudo-codice utilizzata dall'analizzatore dei log per ottenere il grafico della percentuale di visibilità delle portlet.

```
1    procedure showVisibilityChart(portletNum)
2
3       portletNames = let $x := //portlet return $x/@name;
4       coordinates = let $x := //portlet return
5            $x/@coordinates;
6       timestamps = for $x in //event where $x/@type="load"
7            or $x/@type="scroll" return $x/@timestamp;
8
9       old_time = timestamps[0];
10      T = 0;
11
12      for(j=0; j < portletNames.length; j++)
13
14          v = calculateVisibilityPercentage(coordinates[j]);
15
16          if (j % portletNum == 0)
17          i = j/portletNum + 1;
18          time = timestamps[i];
19          t = time - old_time;
20          T += t;
21          old_time = time;
22
23          sum{portletNames[j]} += v * t;
24
25      for each (portlet in sum)
26          visibility{portlet} = sum{portlet} / T;
```

```
27
28      createChart(visibility);
```

La fase di traduzione dei *raw data* estratti può essere affidata a un'applicazione stand-alone: questa dovrebbe innanzitutto occuparsi del pre-processamento dei dati (per eliminare quei valori non significativi o anomali per le analisi successive), successivamente effettuare query sui dati appena processati e successivamente restituire dei risultati. Per ottenere dati facilmente fruibili a diversi tipi di utenti, si può pensare di utilizzare i *raw data* ricavati dallo strumento di logging per ricavare statistiche utili ai fini dell'analisi di comportamento dell'utente sul Web, ad esempio tramite la libreria Java JFreeChart (`http://www.jfree.org/jfreechart/`).
Il punto più critico dell'implementazione potrebbe essere il fatto di dover far comprendere all'applicazione il modello dei dati e delle statistiche che vengono ottenute, in modo tale da ottenere risultati utili alle applicazioni nella progettazione di un portale: è necessario stabilire di quali informazioni si ha bisogno per andare ad ottenere le risposte volute sulle questioni del comportamento degli utenti nella navigazione.

La presente trattazione si propone di tracciare un modello che pone le basi per ulteriori sviluppi futuri del monitoring avanzato descritto da EOP: la sua caratteristica di essere un framework modulare permette infatti di essere facilmente estendibile e compatibile con diversi portal framework.

# Conclusioni

In questa trattazione si è presentato EOP, un framework che si pone come obiettivo la raccolta e l'analisi di dati riguardanti il comportamento degli utenti nella loro navigazione nei portali web, considerando la grande eterogeneità di contenuto che caratterizza quest'utlimi. L'idea è nata con la collaborazione di Engineering Ingegneria Informatica Spa, dove ho svolto un tirocinio formativo nel mese di Ottobre 2014: il caso d'uso affrontato si riferisce a un portale web intranet del Comune di Bologna, il quale ha fatto emergere la necessità di un tipo di monitoraggio più approfondito per permettere all'azienda di effettuare scelte in termini di struttura del portale ai fini di rendere più efficace e performante la navigazione del portale per ogni tipologia di utente.

Il modello proposto supera i limiti delle metriche basate sulla sola visita della pagina, proponendo nuovi parametri di analisi *portlet-based*: la visibilità (in termine di tempo e percentuale di spazio occupato sullo schermo) che possono avere le diverse componenti durante la navigazione, le diverse interazioni dell'utente con le singole componenti anche per rilevare relazioni fra di esse e l'influenza delle singole portlet sulla performance dell'intero portale. Questi sono sono esempi di indicatori essenziali per catturare conoscenza riguardo agli interessi degli utenti e per avere suggerimenti utili nella progettazione di un portale (a livello di design e di personalizzazione).

La caratteristica del framework di avere una struttura modulare permette

di avere uno strumento non troppo invasivo durante la navigazione dell'utente, e questo conferisce inoltre numerosi vantaggi: la modularità rende EOP facilmente modificabile ed estendibile (con l'integrazione di altri moduli); ne facilita il riutilizzo, in una sua parte o nella sua interezza, e consente di ottenere dati che devono essere significativi poi per diversi scopi. Non avendo completato l'implementazione dell'intero framework ed effettuato opportuni test, non si sono potuti verficare concretamente le performance e l'impatto sul sistema: sicuramente sono necessari la messa in produzione e diversi giorni di test con un bacino di utenti maggiore, per evidenziare eventuali bug del componente.

In questa trattazione si è cercato di suggerire la direzione verso la quale dovrebbe andare il monitoring di un portale Web, proponendo modelli da seguire e fattori da considerare durante la navigazione, ponendo le basi per sviluppi futuri, data anche la facilità di estendibilità del framework. I lavori futuri devono avere lo scopo di dimostrare anche l'utilizzo di EOP nei diversi campi di personalizzazione e usabilità di un portale Web.

# Bibliografia

[1] Baglioni, M., et al. “Preprocessing and mining web log data for web personalization” AI* IA 2003: Advances in Artificial Intelligence. Springer Berlin Heidelberg, pp. 237-249, 2003

[2] Bellas, F., “Standards for Second-Generation Portals”, IEEE Internet Computing, 8(2), Mar./Apr., 2004

[3] Bhowmick, P., Sarkar, S., Basu, A., “Ontology based user modeling for personalized information access”, International Journal of Computer Science and Applications, vol. 7, No. 1, pp. 1-22, 2010

[4] Buscher, G., Cutrell, E. and Meredith Ringel Morris, “What do you see when you’re surfing?: using eye tracking to predict salient regions of web pages”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009

[5] Chen, Mon Chu, John R. Anderson, and Myeong Ho Sohn, “What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing”, CHI’01 extended abstracts on Human factors in computing systems. ACM, 2001

[6] Costagliola, G., et al. “Logging and Analyzing User’s Interactions in Web Portals” Web Information Systems and Technologies. Springer Berlin Heidelberg, pp. 213-229, 2008

[7] Diaz, O., Rodriguez, J., "Portlet syndica tion: Raising variability concerns", ACM Transactions on Internet Technology (TOIT), vol. 5, No. 4, pp. 627-659, November 2005

[8] Etzioni, O., Perkowitz, M., "Adaptive Web Sites: Conceptual Cluster Mining", Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Vol. 99, 1999

[9] Fenstermacher, Kurt D., and Mark Ginsburg, "Client-side monitoring for Web mining", Journal of the American Society for Information Science and Technology 54.7, pp. 625-637, 2003

[10] Goldberg, Joseph H., et al,"Eye tracking in web search tasks: design implications", Proceedings of the 2002 symposium on Eye tracking research & applications. ACM, 2002

[11] Hofgesang, Peter I. "Methodology for preprocessing and evaluating the time spent on web pages" Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. IEEE Computer Society, 2006

[12] Hoxha, J, "Semantic formalization of cross-site user browsing behavior", In Research Technical Report, Archiv nr. 3025. Institut AIFB, KIT, 2012. http://www.aifb.kit.edu/web/Techreport3025/en, 2012

[13] Iváncsy, R., Vajk, I., "Frequent pattern mining in web log data", Acta Polytechnica Hungarica 3.1, pp. 77-90, 2006

[14] Java Portlet Specification 168, version 1.0, Ottobre 2003, `http://www.jcp.org/en/jsr/detail?id=168`

[15] Java Portlet Specification 286, version 2.0, Gennaio 2008, `http://www.jcp.org/en/jsr/detail?id=286`

[16] Joshi, A., "Web/data mining and personalization", University of Maryland Baltimore County (UMBC) eBiquity Re-

search Area, `http://ebiquity.umbc.edu/project/html/id/17/Web-Data-Mining-and-Personalization`, 2001

[17] Lee, S., Zufryden, F., Dreze, X., "A study of consumer switching behavior across Internet portal Web sites", International Journal of Electronic Commerce, vol.7, No. 3, pp. 39-64, 2003

[18] Liu, Y., et al. "Clustering web surfers with probabilistic models in a real application", Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence. IEEE Computer Society, 2004

[19] Mahdavi, M., Shepherd, J., Benatallah, B., "A collaborative approach for caching dynamic data in portal applications", Proceedings of the Fifteenth Conference on Australian Database, vol. 27, pp. 181–188, 2003

[20] Mobasher, B., "Web usage mining", Web Data Mining: Exploring Hyperlinks, Contents and Usage Data 12, 2006

[21] Moraga, M.Á., C. Calero, M. Piattini, and O. Diaz, "Improving a Portlet Usability Model", Submitted to: Software Quality Journal, 2005

[22] Murata, Tsuyoshi, and Kota Saito, "Extracting users' interests from web log data" Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society, 2006

[23] Nasraoui, Olfa, et al. A web usage mining framework for mining evolving user profiles in dynamic web sites. Knowledge and Data Engineering, IEEE Transactions on 20.2, pp. 202-215, 2008

[24] Razmerita, L., Angehrn, A., Maedche, A. "Ontology based user modeling for knowledge management systems", Proceedings of the 9th International Conference on User Modeling, Springer-Verlag, pp. 213–217, 2003

[25] Shahabi, Cyrus, et al. “Knowledge discovery from users web-page navigation”, Research Issues in Data Engineering, 1997. Proceedings. Seventh International Workshop on. IEEE, 1997

[26] Srivastava, Jaideep, et al, “Web usage mining: Discovery and applications of usage patterns from web data”, ACM SIGKDD Explorations Newsletter 1.2, pp. 12-23, 2000

[27] World wide web committee web usage characterization activity, `http://www.w3.org/WCA`

[28] WSRP 2.0 Specification, `http://docs.oasis-open.org/wsrp/v2/wsrp-2.0-spec-os-01.html`

[29] Wu, H., Chelmis, C., Sorathia, V., Yinuo Zhang, Patri, O.P., Prasanna, V.K., “Enriching employee ontology for enterprises with knowledge discovery from social networks” Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference, pp. 1315-1322, 25-28 Aug. 2013

# Ringraziamenti

Desidero ringraziare il dott. Di Iorio, relatore di questa tesi, per la grande disponibilità e cortesia dimostratemi, e per tutto l'aiuto fornito durante la stesura.

Un ringraziamento va inoltre a Paolo Lutterotti e Giovanni Amici, che mi hanno seguito durante il tirocinio presso Engineering Ingegneria Informatica Spa, il quale si è rivelato essere un'esperienza preziosa e pienamente formativa.

Ringrazio la mia famiglia e i miei amici più cari, capaci di essere un sostegno su cui poter contare sempre.

Un ultimo ma non meno sentito ringraziamento ai miei fidati compagni di studi: in questo percorso siete stati fedeli compagni di viaggio con i quali ho condiviso soddisfazioni e notti insonni prima delle consegne dei progetti, tante risate e tante giornate di studio in biblioteca... senza di voi sarebbe stato tutto molto più noioso!