UNIVERSITÀ DI PADOVA  FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA

IDENTITY AND ACCESS MANAGEMENT SOLUTION WITH SINGLE SIGN ON

Relatore:
Ch.mo Prof. Sergio Congiu

Laureando:
Alberto Lorenzon

Anno Accademico 2010/2011

Atutti quelli che mi hanno sostenuto

in questo percorso formativo

Indice

Cap. 1

Introduzione

1.1 Scenario attuale

Negli ultimi anni si è registrato un notevole incremento nell'utilizzo di Internet, da parte delle aziende e soprattutto dei consumatori, nelle transazioni commerciali o più semplicemente nella fruizione di servizi disponibili on-line. Le aziende tentano di acquisire il maggior numero di informazioni possibili sugli utenti, per offrire soluzioni sempre più personalizzate, sicure e semplici da utilizzare, interpretando gli interessi dei potenziali clienti. A questo proposito, l'utilizzazione dei sistemi di autenticazione on-line ha subito una brusca crescita, la registrazione del cliente consente, infatti, di raccogliere informazioni, interessi e preferenze che consentiranno, successivamente, di offrire all'utente servizi "su misura". La registrazione del cliente avviene usualmente attraverso la combinazione username/password che può però comportare problemi sia per l'utente che per le aziende, in quanto le password sono funzionali solamente se utilizzate in modo corretto e, sfortunatamente, non tutti gli utenti prestano attenzione al loro utilizzo. Si presentano quindi alcune problematiche che scaturiscono dal comportamento sbagliato degli utenti e che hanno riflessi anche sull'azienda.

Password troppo semplici : Gli utenti sono infastiditi dal dover inserire continuamente le proprie credenziali poiché tale operazione è percepita come un'interruzione della navigazione e per questo tendono a scegliere password corte, basate su parole che possono facilmente essere ricordate, permettendo così, però, ad un

malintenzionato, altrettanto facilmente, di scoprirle (ad esempio con "attacco brute force[1]").

Unica password per sistemi diversi : Per non dover ricordare tante password, gli utenti spesso utilizzano la stessa in sistemi diversi, inclusi siti non protetti, nei quali le informazioni sono inviate in chiaro. Una singola password, così, una volta scoperta, consente l'accesso agli altri sistemi.

Password accessibili : Password lunghe, contenenti differenti tipi di caratteri, sono difficili da scoprire ma sono anche difficili da ricordare e questo spinge gli utenti a scriverle da qualche parte e tenerle in luoghi accessibili e visibili da altre persone, correndo il rischio di compromissione.

Costi di gestione : Secondo uno studio effettuato della NTA Monitor Password Survey, un utente esperto in Information Tecnology utilizza, in media, 21 password, con punte che toccano il numero di 71 ed il 40% delle telefonate agli help desk (fonte: Gartner Group) riguarda lo smarrimento di tali password. Tutto questo impegna l'azienda in un notevole sforzo per salvaguardare la sicurezza con l'impiego di meccanismi di protezione che comportano riflessi anche di natura finanziaria: le società spendono circa 200-300 dollari l'anno, per utente, per mantenere la sicurezza delle password.

Per limitare questa serie di problemi nascono i sistemi Single Sign-On, i quali consentono all'utente di registrarsi una sola volta e di navigare nella rete senza dover reintrodurre continuamente le proprie credenziali. I vantaggi che tale soluzione apporta sono molteplici, in questo caso, sia per l'azienda che per l'utente. L'utente, beneficia di un aumento dell'usabilità dovuto alla riduzione del numero di autenticazioni effettuate durante la navigazione, e l'aumento della sicurezza facendo

[1]Il metodo di "forza bruta" (anche noto come ricerca esausrtiva della soluzione) è unalgoritmo di risoluzione di un problema che consiste nel verificare tutte le soluzioni teoricamente possibili fino a che si trova quella effettivamente corretta.

uso di una sola password e consentendo quindi di memorizzarla senza scriverla o tenerla in luoghi accessibili ad altri.

Per l'azienda invece, i benefici arrivano a livello di semplicità di gestione ossia la diminuzione della frammentazione delle informazioni e un minor carico sulle applicazioni le quali non devono gestire il processo di autenticazione con evidente diminuzione di costi di gestione in quanto gli utenti ed il sistema impegnano meno gli help desk e l'amministratore.

Le difficoltà che gli ideatori di tale soluzione devono affrontare concernano oltre allo sviluppo di un sistema sicuro, l'integrazione della stessa con le tecnologie e le applicazioni esistenti, evitando la necessità di creare infrastrutture ponte per la compatibilita e sviluppando un sistema scalabile e modulare, tenendo conto dei nuovi scenari aperti dalle nuove tecnologie.

1.2 Contenuto della tesi

Lo scopo di questa tesi è inanzitutto l'analisi di un portale OpenSource (Liferay Portal e i numerosi software ad esso associati) in cui verranno integrati tutti i software aziendali già in possesso dalla società dove sviluppo il tirocinio (ossia l'azienda SMC Computer di Lancenigo di Villorba TV), dopo di che l'attenzione verrà rivolta sul vero significato del termine Single Sign On e le relative problematiche riguardanti la sicureza informatica, e per finire quindi ad analizzare le varie soluzioni software a disposizione ora nel mercato che permettono la realizzazione di un servizio di SSO e la conseguente progettazione ed implementazione nel portale di alcuni di questi sistemi di autenticazione e autorizzazione.

Gli argomenti di questa tesi saranno discussi in tre diversi capitoli:

- Capitolo Secondo: Sarà presentato il significato di portale, illustrando le caratteristiche di Liferay Portal ed i software ad esso associati.

- Capitolo Terzo: Saranno illustrati i principi fondamentali riguardanti il concetto di sicurezza, dopo di che sarà analizzato in modo dettagliato il significato del termine ed il concetto di Single Sign On analizzando inoltre i software attualmente disponibili, che permettono agli sviluppatori di realizzare servizi di Autenticazione Single Sign On.
- Capitolo Quarto: In questo capitolo saranno illustrati alcuni servizi di Single Sign On che sono stati analizzati, svilupparti e integrati con il portale Liferay .

Cap. 2

Liferay Portal

2.1 Liferay

Un portale è un sito web che consente agli utenti di localizzare risorse, fruire di servizi ed utilizzare applicazioni che necessitano di internet o di intranet e forniscono servizi, contenuti e collaborazioni commerciali e non, dando la possibilità al cliente di personalizzarne i contenuti. Un portale utilizza al suo interno dei costrutti (componenti) software dinamici definiti portlet che sono in definitiva delle componenti Java . Queste componenti sono dei moduli web riusabili all'interno di un portale web e quindi una pagina di un portale è suddivisa in diversi campi il cui contenuto viene definito da un portlet specifico (ad esempio: Barra di navigazione, Calendario, Orologio, Meteo....), i quali vengono gestiti per il loro ciclo di vita da un portlet container (es. nel caso di LifeRay è Tomcat) e modificabili secondo delle richieste specifiche di un utente del portale .

Liferay Portal è un portale web open source basato sulla technologia Java (viene definito come gli altri portali come Portal Server), offre un aiuto agli utenti e sviluppatori dove si possono testare le portlet, ma anche fornendo applicazioni pronte all' uso. Utilizzato da aziende in tutto il mondo , comprende una lunga lista di funzionalita che lo mettono a confronto diretto con molti portali commerciali, con il vantaggio di non avere oneri di licenza.
Nel contesto applicativo possiamo dire che Liferay implementa le portlet, che costituiscono una interfaccia facile da modificare.

Liferay si presenta in due versioni, l'Enterprise e la Professional; la prima è stata creata per essere installata su delle application server che supporta gli EJB[1] (Enterprise Java Bean) e il deploy può essere fatto solo nei JavaEE[2] Application Server, mentre la seconda è una versione creata per supportare servizi POJO più altre funzionalità offerta da Spring (come presenta i servizi interni e gli stratti di accesso ai dati sotto forma di Plugin[3]). Il deploy può essere fatto nei servlet container (ad esempio Tomcat , Jetty..). Praticamente le due versioni sono quasi identiche dal punto di vista funzionale, il software è lo stesso , solo che nella Enterprise vengono inseriti gli EJB piuttosto che i servizi locali con la necessità di un Application Server (Jboss, WebSphere, Geronimo,..) . Bisogna quindi segliere la versione che si addatta di più alle esigenze aziendali.

2.1.1 Portlet

I portlet formano quindi delle pagine all'interno di un portale, ed hanno come scopo quello di sviluppare dei portlet portatili che possono essere usati nel contesto di portali sviluppati con tecnologie differenti.

Un portlet quindi è un archivio o una classe che implementa l'interfaccia di Java ed è inserita all'interno del portlet container .

I portlet non hanno comunicazione diretta con il browser e inoltrano richieste o scrivono markup al flusso in uscita e rappresentano singoli componenti aggregati dal portale, il quale svolge la funzione di Web container. I clienti web interagiscono attraverso il portale con i portlet, i quali hanno delle modalità predefinite (view,

[1] Gli Enterprise JavaBrean sono i componenti che implementano, lato server, la logica di business all'interno dell'architettura JAVA EE.

[2] Java EE (dall'inglese Java Enterprise Edition) è la versione enterprise della piattaformaJava.

[3] Il Plugin è un programma non autonomo che interagisce con un altro programma per ampliarne le funzioni. In alcuni casi, il plugin viene anche denominato estensione (extension).

edit, help mode) e degli stati delle finestre che ne definiscono la dimensione (minima, massima, normale). In un portale coesistono nella stessa pagina più portlet, i quali possiedono mezzi già predefiniti per la configurazione e personalizzazione dei contenuti. Liferay inoltre mette a disposizione ad utenti e ad amministratori la possibilità di aggiungere, modificare e utilizzare dei portlet già preconfigurati e pronti all'uso .

2.1.2 Concetto di Comunity

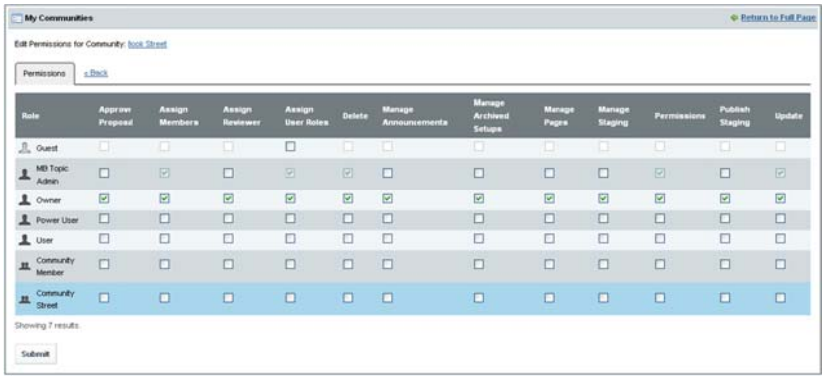
Il primo passo per la creazione di un sito web utilizzando Liferay consiste nella creazione di una community che è un insieme di utenti con un interesse comune che condivide un'area specifica del sito; quindi un sito web viene visto come un'area del portale dedicata ad una community. Successivamente alla creazione della community è possibile aggiungere vari portlet a disposizione dal menu "Aggiungi Applicazione" e la portlet appena istallata consentirà di inserire contenuti all'interno della pagina.

Gli utenti di Liferay possono accedere ai loro contenuti e alle loro applicazioni da un unico punto di accesso, il portale quindi può aggregare diversi portlet e renderli disponibili accedendo una sola volta tramite il Single Sign On. Per garantire che le persone accedano facendo uso delle sole informazioni e ai dati per le quali sono autorizzate, gli amministratori del portale possono assegnare ai singoli utenti o ai gruppi a cui gli utenti fanno parte, ruoli diversi per attribuire loro differenti livelli di accesso e differenti diritti di modifica.

Gli utenti abilitati hanno a disposizione uno spazio personale dove inserire le proprie informazioni pubbliche ed hanno anche la possibilità di renderle private, quindi gli utenti vengono raggruppati in una gerarchia di "organizzazioni" o "comunità", ad esempio i membri di una determinata area possono essere raggruppati in una organizzazione, mentre in base alle funzioni che svolgono possono essere suddivisi in varie comunità. Da quello che è stato appena illustrato si denota quindi che l'architettura del portale Liferay è stata costruita intorno alla figura degli utenti, delle Organizzazioni, delle Comunità, dei Ruoli, delle pagine e dei Portlet che permettono all'utente di creare un portale con determinate caratteristiche.

2.1.3 Portal Comunity

Una Portal Community è un insieme di pagine pubbliche e private, associate al concetto di “Community” nelle quali possono essere integrate applicazioni web con la possibilità di definire livelli differenti di privilegi legati alle singole utenze.



Liferay distingue tra utenti che possono agire attivamente nella modifica della struttura del portale e quelli che invece usufruiscono solo ed esclusivamente dei contenuti finali. La distinzione avviene a livello di ruolo utente ed è tra i profili.:

- Power User: Utente che possiede i privilegi per poter customizzare una sua pagina personale, attribuita alla creazione dell’utenza. Questa pagina fa parte di uno spazio virtuale che Liferay associa ad ogni singola utenza al momento della sua creazione.

- User: Utente che non possiede spazio personale, appartenente alla community, nella quale può solo usufruire dei servizi messi a disposizione, con privilegi definiti dall'amministratore del sistema.
- Administrator: Super utente che può definire community, creare pagine e può definire permessi per le applicazioni esposte negli spazi pubblici e privati. È lui il gestore master della struttura del CMS.
- Guest: Utente ospite del portale. Può solo navigare i contenuti pubblici messi a disposizione.

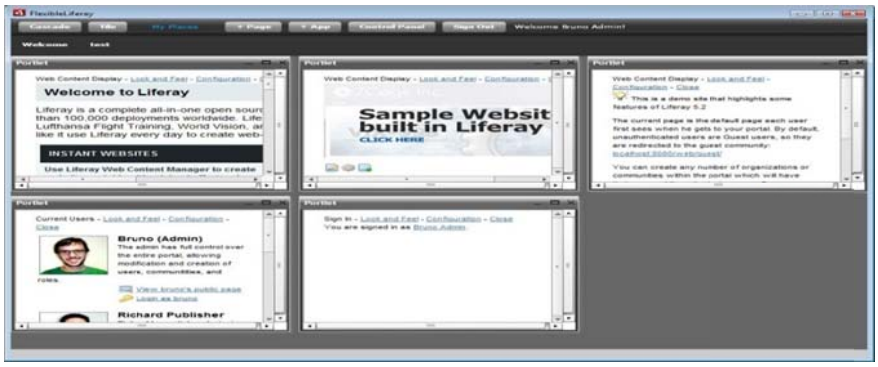
In ogni community ci sono utenti generici e amministratori, si possono specificare permessi diversi per ciascuno di questi ed inoltre un utente amministratore nella community non necessariamente deve essere anche un amministratore del portale. Gli amministratori del portale hanno un controllo assoluto sugli accessi e sui privilegi delle portlet e degli oggetti che compongono ciascuna community. Le varie modalità che consentono di assegnare i permessi agli utenti del portale sono.:

- ***Entità Resource***: Rappresenta qualsiasi oggetto all'interno del portale. Esempi di risorse sono le portlet, i documenti, le informazioni, le immagini, le cartelle etc.
- ***Permission***: Un permesso viene definito come un'azione su una particolare Risorsa; es.: View , Add-Discussion
- ***Role***: Un ruolo è una collezione di permessi i quali vengono definiti in modo inclusivo.
- ***User***: Un utente è un individuo che esegue determinate azioni all'interno del portale. A seconda dei permessi e dei ruoli assegnati a un utente può eseguire o non eseguire determinate azioni. Prima di loggarsi all'interno del portale uno user è considerato un guest ma una volta loggato viene considerato un utente e quindi può ricevere permessi nei seguenti modi:
    - Assegnazione diretta all'utenza;

    - Ereditandoli dai permessi associati alla community di appartenenza;
    - Ereditandoli dai permessi associati alla location di appartenenza;
    - Attraverso il Ruolo assegnato all'utente;
    - Attraverso il Ruolo assegnato alla community di appartenenza;
    - Attraverso il Ruolo assegnato all'organizations di appartenenza;
    - Attraverso il Ruolo assegnato alla location di appartenenza;
- ***Organization e Location***: Ruoli e permessi individuali possono essere assegnati a organizzazioni e sotto organizzazioni, di default una sotto organizzazione eredita i permessi dall'organizzazione padre.
- ***Community***: Una community è un gruppo di utenti con un interesse particolare
- ***User group***: Uno user group è un insieme di utenti dove ruoli e permessi individuali possono essere assegnati ad un determinato gruppo di utenti.

2.1.4 Web Content Management

Web Content Management rappresenta l'interfaccia web principale di gestione di Liferay e consente agli utenti di creare, modificare e pubblicare articoli e di modificarne il layout in base alle esigenze. L'elenco dei contenuti Web visualizza un elenco dinamico di tutti gli articoli per una determinata comunità e può includere gli articoli per data di creazione, data di pubblicazione, titolo o altri criteri che verranno aggiunti successivamente.



Alcuni esempi di contenuti sono :

- Web Content Display può essere utilizzato per pubblicare contenuti Web in una pagina del portale e comprende la maggior parte dei contenuti.
- Web Content è uno strumento che consente agli utenti finali di ricercare tutti i contenuti Web di un sito internet.
- Asset Editor consente agli utenti di pubblicare qualsiasi tipo di contenuto nel proprio portale, come se si trattasse di un contenuto web filtrando attraverso una serie di regole di pubblicazione o da selezione manuale.
- Blog è un portlet che include funzionalità di interazione completa, il supporto RSS, commenti dei clienti, cartellini ed etichette, i collegamenti bookmarking sociali, le notifiche e-mail di risposte sul blog, e un sistema di rating voce.
- Mail è un portlet che può essere configurato per interfacciarsi con molti popolari server di posta consentendo agli utenti di inviare e controllare la posta elettronica direttamente attraverso il portale.

2.1.5 Specifiche Tecniche

LifeRay è un prodotto che si distingue per alcune caratteristiche quali la completezza, solidità e flessibilità, dato che nel pacchetto base è presente quanto necessario per creare un portale di base completo e implementarlo con differenti

soluzioni architetturali. Liferay inoltre è un portale web open source che si propone di aiutare le organizzazioni a collaborare in modo più efficiente fornendo una serie consolidata di applicazioni pronte all'uso. È possibile sviluppare un qualsiasi Portlet che rispetti le specifiche JSR_168[1], questo vuol dire che si potranno aggiungere funzionalità al Portale scrivendo dei Portlets standard, ossia una serie di Portlets che fanno da CMS per la costruzione di un sito statico esterno al portale. Questo sistema CMS è basato su un meccanismo a template che permette tramite XML/XSL/XSLT [2] di separare completamente i dati relazionali dalle informazioni di impaginazione.

Di seguito vengono descritte in dettaglio alcune importanti caratteristiche .:

- ***SingleSignOn.*** E' possibile accedere ai propri contenuti e alle proprie applicazioni da un punto di accesso unico. Liferay Portal Server può aggregare diversi sistemi applicativi e renderli disponibili accedendo una volta sola con il massimo della riservatezza tramite il Single Sign On.
- ***ASP Model.*** Liferay è stato disegnato fin dall'inizio per essere utilizzato da Application Service Providers (APS[3]), quindi è possibile farci girare sopra più istanze di portali completamente indipendenti.
- ***Application Server Agnostic.*** Liferay può essere installato praticamente su qualsiasi servlet container o application server standard J2EE. Liferay funziona sia su Tomcat, Jetty, etc., che su application server commerciali come Borland ES, Oracle9iAS, Weblogic, etc. Ovviamente essendo scritto in Java la propria esecuzione non è legata al tipo di sistema operativo.

[1] Standard Java che definisce le specifiche utili alla creazione di un portletStandard

[2] Metalinguaggi di markup che definiscono un meccanismo sintattico, che consente di estendere o controllare il significato di altri linguaggi marcatori

[3] Modello architetturale per l'erogazione di servizi informatici che prevede una spinta remotizzazione elaborativa ed applicativa

- ***Spring, EJB, and AOP.*** Lo strato business di Liferay è scritto utilizzando Spring. Ciò permette di utilizzare le caratteristiche AOP, IOC e Proxy di Spring per personalizzare più agevolmente il codice. Utilizzando Spring si può scegliere se utilizzare gli strati di servizio POJO o gli strati EJB.
- ***Database Agnostic.*** Liferay utilizza Hibernate come tool di persistenza, quindi può girare su qualsiasi database da esso supportato. Attualmente Hibernate supporta una vasta gamma di database quali : DB2, Firebird, ypersonic, InterBase, JDataStore, MySQL, Oracle, PostgreSQL, SAP, SQL Server.
- ***Scalable N-Tier Cluster.*** Liferay utilizza OSCache per offrire un livello di cache clusterizzata. Quindi è possibile scalare i deployment senza sacrificare il livello di caching.
- ***Struts and Tiles.*** Lo stato di presentazione utilizza Struts come framework MVC. Il suo utilizzo permette agli sviluppatori di trovare un ambiente "familiare" per lo sviluppo di nuove Portlet. Il Look and Feel del portale può essere facilmente modificato grazie al fatto che la presentazione si basa su templates Tiles.
- ***Internationalization.*** Liferay è sviluppato utilizzando l'internazionalizzazione e include traduzioni per una dozzina delle lingue più diffuse al mondo.
- ***Personalization.*** Il portale permette agli utenti di creare le proprie pagine e di disporre le Portlet a piacimento all'interno di schemi pre impostati (una, due, tre colonne).
- ***Administration.*** Liferay comprende delle Portlet di amministrazione per la gestione di Utenti, Ruoli, Gruppi e Permessi. I Gruppi sono insiemi di utenti, i Ruoli sono dei permessi che possono essere assegnati a gruppi o a singoli utenti. L'accesso alle Portlet è determinato in base ai

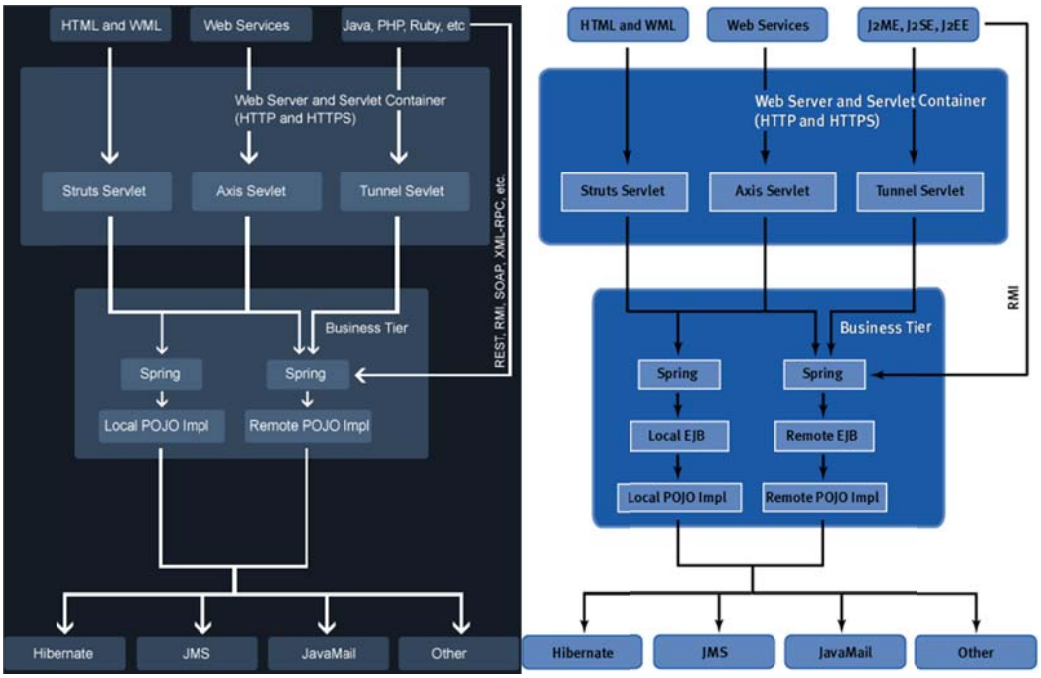
Ruoli. Gli amministratori possono gestire le pagine visibili dai vari gruppi definiti.

2.1.6 Architettura

Scendendo in specifico e facendo un'analisi a basso livello si può denotare che l'architettura di Liferay è composta in 3 settori:

1. Un primo strato di presentazione utilizza la libreria Struts per l'instradamento delle richieste, il sistema di template Tiles si occupa della creazione delle pagine;
2. Il Business Tier, basato su Spring, contiene la logica dell'applicazione ed è preposto alla comunicazione tra lo strato di visualizzazione e quello di persistenza;
3. Lo strato di persistenza si basa su Hibernate, questo consente di utilizzare la base dati in un ottica object oriented e permette un disaccoppiamento da un particolare database.

   Hibernate non è l'unico framework utilizzato, in quanto sono presenti anche Java Message Service e JavaMail per l'integrazione con server di posta.

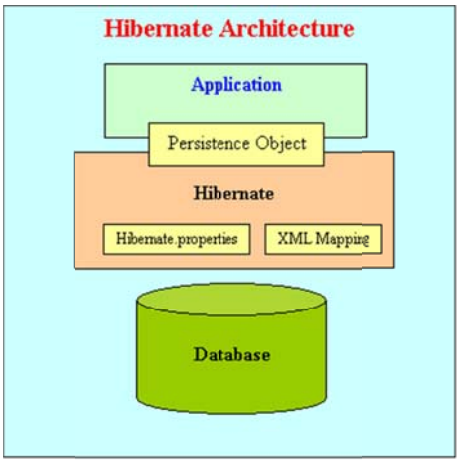


2.2 Software associati a Liferay

2.2.1 Hibernate

Hibernate utilizza il database e i dati di configurazione per fornire servizi di persistenza (e oggetti persistenti) per l'applicazione. Per utilizzare Hibernate, è necessario creare le classi Java che rappresentano la tabella del database e quindi mappare la variabile di istanza della classe con le colonne del database. Può essere utilizzato per eseguire operazioni sul database come selezionare, inserire, aggiornare e cancellare i record della tabella, ed inoltre crea automaticamente le query per eseguire queste operazioni.

L'architettura di Hibernate ha tre componenti principali, il Connection Management, il Servizio di gestione delle connessioni e l'Object relational mapping. Il Servizio di gestione delle connessioni fornisce una gestione efficiente delle connessioni al database, tenendo presente che la connessione al database è la parte più costosa dell'interazione con il database in quanto richiede molte risorse e l'apertura e la chiusura della connessione al database.



La Transaction Management rigurada il Servizio di gestione delle transazioni il quale prevede la possibilità per l'utente di eseguire più di una richiesta contemporaneamente al database. Per finire quindi, l'Object relational mapping, ossia la tecnica di mappatura della rappresentazione dei dati, da un modello a oggetti ad un modello di dati relazionale. Questa parte di Hibernate è usata per selezionare, inserire, aggiornare e cancellare i record. Quando si passa un oggetto a un metodo session.save, Hibernate legge lo stato delle variabili di tale oggetto ed esegue le query necessarie.

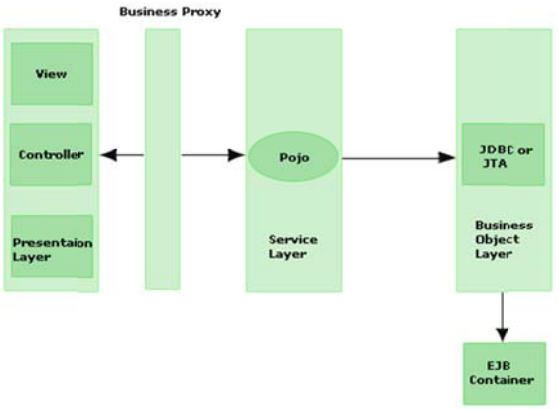
Hibernate è uno strumento molto utile per quanto riguarda il concetto di mappatura della rappresentazione dei dati, ma in termini di gestione della

connessione e la gestione delle transazioni, è privo di prestazioni e funzionalità. Solitamente viene utilizzato con altri programmi che gestiscono la connessione e di altri strumenti di gestione delle transazioni. Per esempio, Apache DBCP è utilizzato per il pool di connessioni con Hibernate.

2.2.2 POJO

I POJO (Plain Old Java Object) sono semplici oggetti Java ossia dei Javabean che encapsulano le informazioni in unità ben distinte e facilmente identificabili dal punto di vista funzionali. Non implementano alcuna interfaccia né hanno bisogno di un Application Server per poter funzionare, sono quindi semplici classi Java con le regole seguite dai Javabean.

Un EJB 2 usato per la persistenza portava con sé la problematica di essere un



oggetto remoto e quindi la necessità di esportare non l'oggetto in sé, ma il proxy che attraverso RMI avrebbe effettuato le operazioni di lettura/scrittura sul database (ad esempio un'architettura dove Web server e application server sono distribuiti). In particolare, quest'ultima situazione ha visto moltiplicarsi l'uso di oggetti POJO incapsulati in un EJB in modo da poter essere facilmente rappresentati e serializzati in fase di visualizzazione. Come ogni javabean creiamo una classe che contiene le variabili di istanza rappresentanti la nostra entità, in questo caso un id, un nome e il valore del saldo. Segue la definizione dei costruttori (quello di default deve essere sempre definito) e dei metodi getter e setter. Eventualmente è possibile inserire altri metodi, come una normale classe Java, ovviamente è necessario annotare la classe in modo che chi ne gestirà le operazioni di persistenza sappia come gestirla. La classe dovrà essere annotata come Entity e già di per sé sarebbe

sufficiente in quanto, come default, tutte le variabili vengono considerate come persistenti e quindi gestite come tali. Ci comportiamo in maniera normale, senza preoccuparci di persistenza alcuna in quanto tutto dovrà essere gestito dal layer di persistenza.

Altri framework non consentono questa funzionalità, che è una peculiarità molto importante in quanto permette a diversi team di lavorare in maniera separata ed effettuare le proprie prove.

2.2.3 Java Persistence API

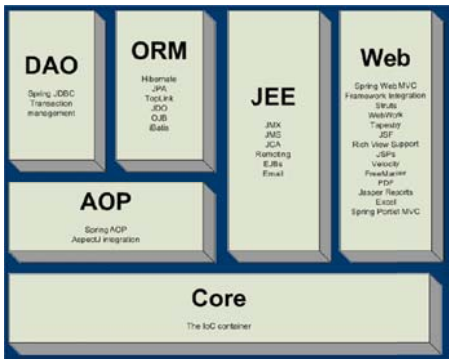
Spesso il primo passo nello sviluppo di applicazioni enterprise consiste nella creazione del Domain Model, ovvero dell'insieme di entità del dominio e delle relazioni fra esse. Il Domain Model è l'immagine concettuale del problema che il sistema deve risolvere, esso descrive oggetti e relazioni ma non si occupa di definire come il sistema agisce su tali oggetti. L' obiettivo è quello di identificare le entità che devono essere persistenti e quindi memorizzate nel database.

Gli aspetti sui quali occorre prestare attenzione sono gli Oggetti, le Relazioni, le Molteplicità delle relazioni e l'Opzionalità delle relazioni. Gli Oggetti dal punto di vista dello sviluppatore sono oggetti java con stati e comportamenti mentre le Relazioni sono rappresentate dal dal fatto che un oggetto ha al suo interno un riferimento ad un altro oggetto. Il riferimento determina la direzione della relazione che può essere unidirezionale quando un oggetto referenzia un altro ma non il contrario, oppure bidirezionale nel momento in cui due oggetti si referenziano a vinceda. Le molteplicità delle relazioni entrano in gioco quando da una parte della relazione è presente più di un oggetto, ad esempio un oggetto Categoria potrebbe mantenere il riferimento a più oggetti Articolo. Esistono quindi tre tipi differenti di relazioni, "uno a uno" in cui ogni lato della relazione ha al più un oggetto, "uno a molti" in cui una particolare istanza di un oggetto può fare riferimento a più istanze di un altro ed infine "molti a molti" qundo entrambi i lati della relazione possono fare riferimento a più istanze di un oggetto.

L'ultimo aspetto riguarda l'opzionalità delle relazioni, la quale determina se la relazione deve esistere obbligatoriamente oppure è opzionale. In EJB 3 la persistenza è gestita mediante Java Persistence API (JPA), tutto ciò che bisogna fare al fine di rendere persistenti gli oggetti del domain model è codificare il domain model in POJO e usare le annotazioni (o deployment descriptor) per fornire al persistence provider informazioni quali sono gli oggetti del domain model, come identificarli univocamente, quali relazioni esistono tra gli oggetti e come mappare un oggetto in una tabella del database.

2.2.4 Spring

Spring è un framework che semplifica lo sviluppo di applicazioni Java e in particolare J2EE, promuovendo l'uso di *best practice* di design, secondo un modello di sviluppo *POJO-based.* Non dipende quindi da nessuna API Java EE, pertanto è utilizzabile sia all'interno che all'esterno di una Application Server.



I principi architetturali su cui si basa questo framework son principalemente di due tipi, l'Inversion of Control e il Dipendency Injection. Il primo è il framework che chiama il codice e non viceversa, (L'entità più astratta invoca i metodi dell'entità più concreta), ossia con Spring è il framework che si occupa della costruzione delle istanze che vogliamo e non le istanze stesse. Le new le fa il framework e non il programmatore ed è proprio questa l'inversione di controllo. Mentre il secondo (Dependency Injection), è caratterizzato dal fatto che il singolo oggetto dichiara le proprie dipendenze, ma non le istanza/cerca, è il framework che le risolve, istanzia e ritorna.

Il modulo principale di Spring su cui si basa l'intero framework è un loC container che implementa DI osia la BeanFactory. La DI non è altro che la dipendenza tra due entità impostate nel file di configurazione xml, in altre parole

DI sono i vari *ref* di un'istanza ad un'altra. Le dipendenze vengono specificate tramite XML , Annotazioni , Codice Java.

Per quanto riguarda l'Interazione tra Spring e gli altri programmi, un' applicazione anziché creare delle istanze delle classi (Classe 1 , Classe 2 , …. , Classe N), le chiederà a Spring che farà da "intermediario" (ossia container) compiendo le seguenti procedure.:

- L'applicazione crea una nuova istanza di classe contesto Spring indicandole il file .xml da caricare per costruire i bean.
- Il contesto carica il file xml contenete le istruzioni su quali istanze caricare in esso.
- In base al file xml letto, il contesto crea una sorta di indice delle istanze che può restituire (istanze di Classe 1, Classe 2, …, Classe N e cosi via).
- L' applicazione chiede al contesto una delle istanze configurate
- Il segreto in spring è come configurare il contesto, ossia quali istanze metterci dentro e in che relazione sono le istanze tra loro

I punti di forza e i vantaggi che si possono riscontrare utilizzando questo prodotto sono inanzitutto il codice molto più modulare e sintetico, ed inoltre non dipende dal framework. Posso inoltre utilizzare le stesse classi in altri contesti e i molti moduli di Spring possono essere usati separatamente, decidendo incrementalmente quali e quando. Per finire l'applicazione risulta riconfigurabile senza bisogno di ricompilare.

2.2.5 Struts

Struts è un framework di Apache per la realizzazione di applicazioni web, è conosciuto come un framework che implementa il pattern MVC[1] per lo sviluppo unificato di Servlet e JSP, mette a disposizione utility per lo sviluppo di

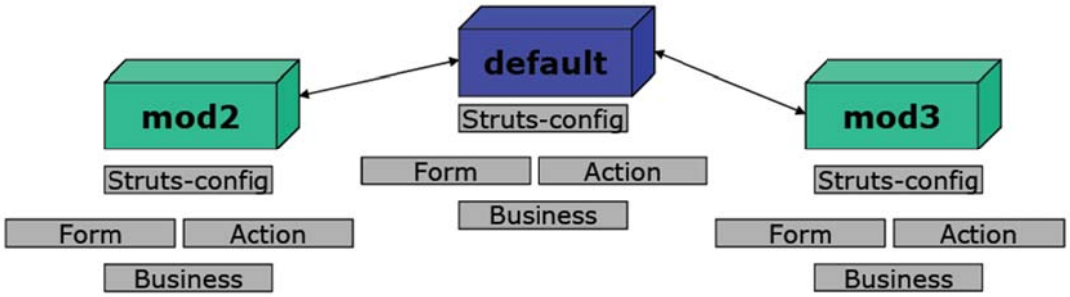
[1] Model View Controller è un pattern architetturale molto diffuso nello sviluppo di interfacce grafiche di sistemi software object-oriented. Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali, il model, il view ed il controller.

applicazioni web e una serie di librerie di tag per la gestione del rendering e mapping dell'html.

I moduli di Struts permettono di suddividere l'applicazione in sotto unità e facilitano il coordinamento del gruppo di lavoro, si possono inoltre realizzare librerie complete di controller e model con Struts-config incorporato.

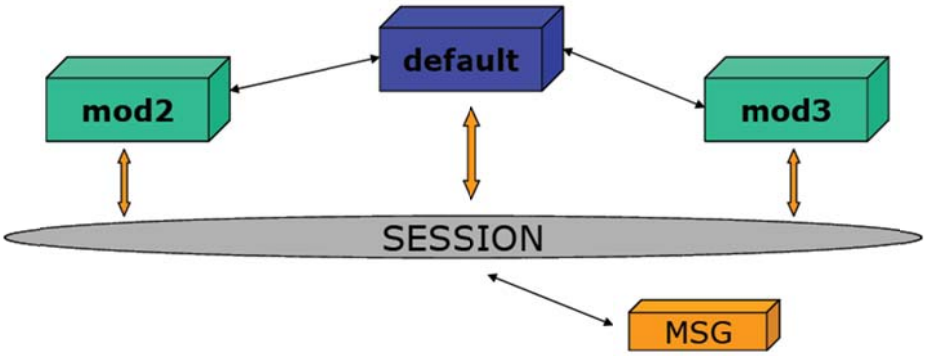
2.2.5.1 Contesto dei moduli .:

- Action di "ingresso" al modulo
- Le Action vengono cercate nello struts-config del modulo senza bisogno del prefisso del modulo



2.2.5.2 Condivisione delle informazioni tra i moduli

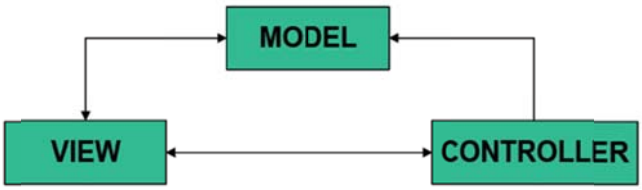
- Il contesto "session " è visibile a tutti i moduli
- Si possono creare oggetti per lo scambio di messaggi



2.2.5.3 Pattern MVC

Lo scopo del pattern è quello di disaccoppiare il più possibile le componenti dell'applicazione : controllo, logica di business e presentazione dividendo quindi le funzioni per competenza permettendo uno sviluppo e una manutenzione più

flessibile. Rende possibile sostituire uno dei tre componenti lasciando inalterati gli altri: migrazione da interfaccia web a rich-client e viceversa.



Model .: Fornisce i metodi per accedere ai dati utili all'applicazione

- Implementazione della logica business
- Accesso ai dati con pattern DAO, EJB , ecc…

View .: Visualizza i dati contenuti nel model e si occupa dell'integrazione con utenti e agenti

- Pagine JSP per gestire gli aspetti di rendering grafico
- Sezione che si occupa della presentazione dei dati all'utente

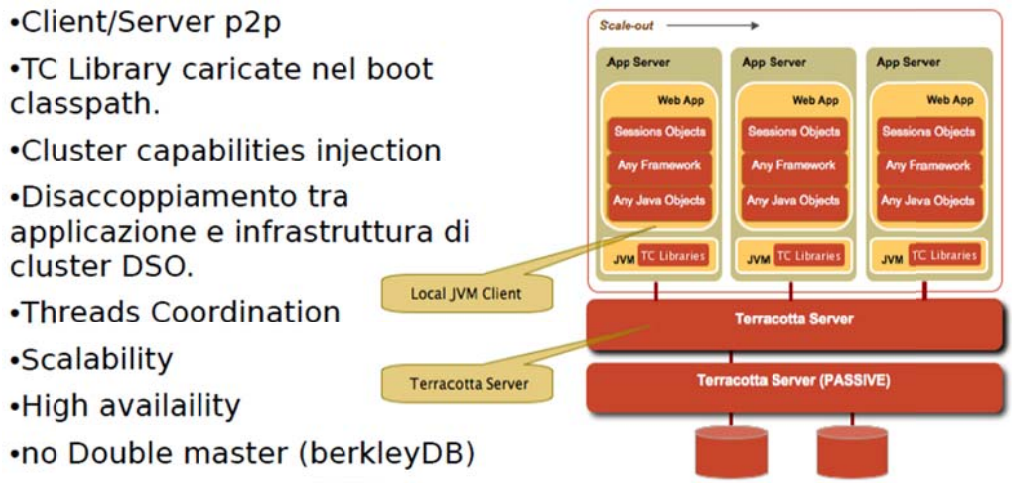
Controller .: Riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti

- Gestisce tramite Servlet le richieste dell'utente
- Centralizzazione dei percorsi di navigazione

2.2.6 Terracotta

Terracotta è una soluzione opensource per il clustering a livello JVM e soddisfa i requisiti di scalabilità e affidabilità, soddisfa inoltre il Clustering trasparente a livello applicativo e fa interagire le applicazioni distribuite come se fossero su una unica JVM.

In una JVM i threads interagiscono gli uni con gli altri attraverso il cambiamento degli oggetti residenti nell'Heap e attraverso le primtive concorrenti ('synchronized' keyword, wait(), notify() e notifyAll()). Terracotta estende il loro significato per una sincronizzazione distribuita.



I benefici che vengono riscontrati sono i seguenti .:

- Cluster non applicativo, è un cluster di JVM
- Separazione dei concetti tra Business Logic e Infrastructure Object
- Nessuna API Java nuova da imparare
- Non c'è serializzazione
- Non ci sono metodi CUSTOM da implementare per la replicazione
- Un programmatore deve semplicemente conoscere la programmazione concorrente

HTTP Session Clustering

La memoria occupata dalle sessioni cresce 1:1 con gli utenti che usano il sistema, quindi c'è bisogno di una clusterizzazione orizzontale, con un load balancer (stiky). Terracotta supporta la replicazione delle sessioni senza bisogno di API speciali e senza bisogno della serializzazione ed inoltre non introduce i noti problemi della full replication.

Terracotta preserva l'identità di un oggetto

- Non serve chiedere la copia "fresca"
- NO getAttribute() / setAttribute()
- Non ci sono copie
- Un oggetto condiviso si comporta come un qualsiasi altro oggetto
- Ogni cambiamento effettuato ad un oggetto condiviso, è disponibile su tutti gli altri oggetti che possiedono la sua reference
- Dati due ref. 'a' e 'b' che referenziano 'c',
  - a.equals(b)
  - a == b.

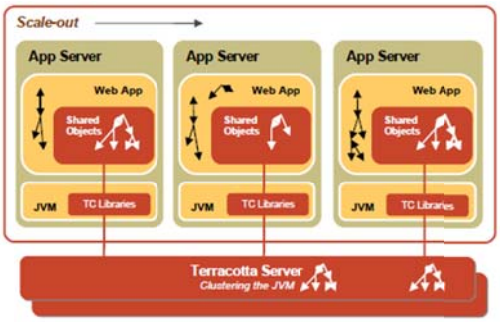
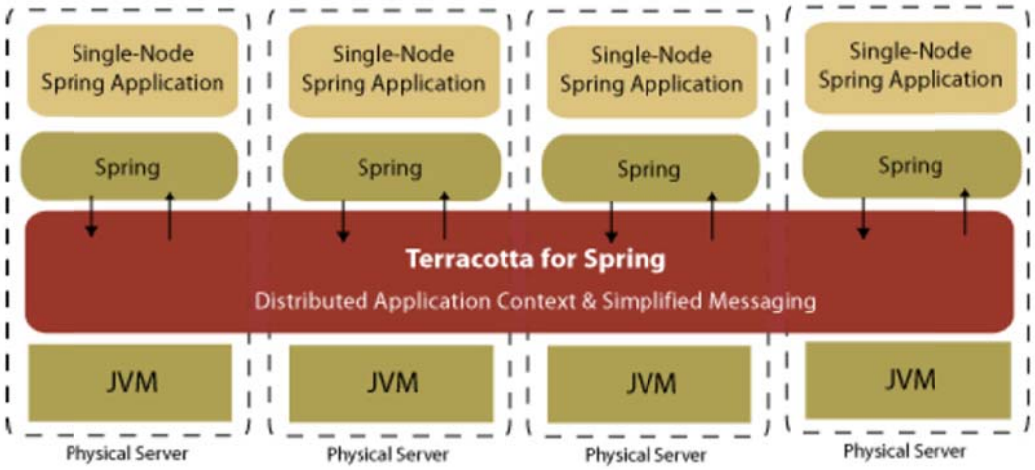


Illustrazione schematica di un'integrazione specifica di terracotta con Spring



Cap. 3

Sicurezza Informatica

Uno degli argomenti più caldi in questo periodo è la sicurezza, fortunatamente e sempre più spesso, gli sviluppatori se ne interessano e la vedono come un argomento serio che deve essere affrontato sin dalle prime fasi dell'analisi di una nuova applicazione.

La sicurezza in quanto tale deve essere pensata a tutti i livelli:

- Livello *architetturale*: la sicurezza va pensata globalmente, per tutta l'applicazione, vedendo la stessa come un corpo estraneo all'interno del nostro sistema, l'applicazione deve essere in grado di difendersi dagli attacchi che il nostro sistema può, più o meno volontariamente, portarle o, come minimo, deve sapere che determinati attacchi possono arrivare.
- Livello di *codice*: ogni singola parte del nostro codice deve partire dal presupposto che i controlli di sicurezza fatti in una determinata situazione possano fallire, possano non esserci o possano essere sbagliati.
- Livello *sociologico*: anche se abbiamo realizzato tutto secondo i canoni e i dettami più rigorosi, l'ultimo ostacolo alla vera sicurezza resta l'impatto che la nostra applicazione avrà sull'utilizzatore finale, quali passi quest'ultimo dovrà intraprendere per far si che la sicurezza venga rispettata e mantenuta, quali sforzi dovrà fare per digerire il nostro concetto di sicurezza e per digerire l'approccio e l'impronta che noi abbiamo deciso di dare alla sicurezza nella nostra applicazione.

  Se i passi e gli sforzi che l'utente dovrà fare sono o troppo difficili o comportano tempi di attesa decisamente non accettabili o, peggio, hanno conseguenze quantomeno scomode, allora significa che abbiamo fallito, la sicurezza non verrà rispettata e le falle che si creeranno nella nostra

applicazione rischieranno di diventare falle per l'intero sistema ospite, compromettendo, in questo modo, la sicurezza dell'intera infrastruttura. Cominceremo ad assistere a fenomeni ben noti a tutti come la scelta di password decisamente troppo semplici o peggio ancora lo scambio di credenziali tra gli utenti per poter eseguire operazioni che con il proprio account o non possono essere fatte o sono troppo complesse da eseguire.

- Livello *amministrativo*: quello amministrativo è un'altro punto essenziale, la sicurezza ha un costo anche in termini di manutenzione nel tempo, pensiamo solo al tempo che un amministratore di rete dedica a mantenere aggiornata la struttura di gestione degli accessi che ha in gestione, se l'infrastruttura di sicurezza introdotta dalla nostra applicazione ha un costo di gestione elevato rischiamo che alla lunga venga trascurata perchè troppo onerosa.

  Se focalizziamo l'attenzione sugli ultimi due punti ci rendiamo rapidamente conto che uno dei "must" è quello di cercare di rendere la sicurezza il più integrata possibile con il sistema ospite, cercare di renderla il più trasparente possibile a tutti, sia agli amministratori che dovranno gestirla e ancora di più agli utenti finali che dovranno "subirla" come imposizione.

  In questa direzione una risposta e, di conseguenza, una possibile soluzione è il concetto di "Single Sign On"

3.1 Principi Fondamentali

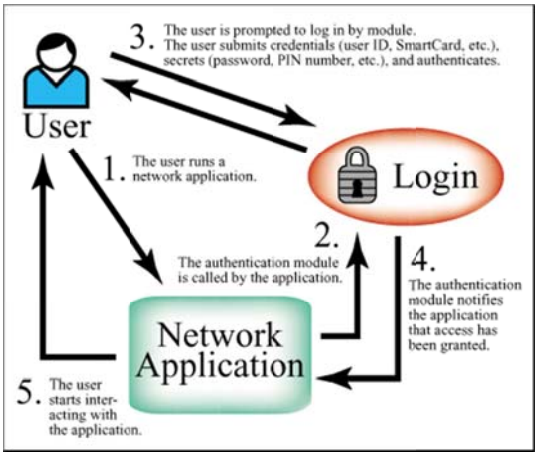
È necessario distinguere nettamente il significato intrinseco presente in tre principi fondamentali su cui si basa parte della sicurezza informatica e in particolar modo per quanto riguarda l'accesso ai dati; ossia l'identificazione, l'autenticazione ed infine l'Autorizzazione.

L'identificazione può essere riassunta, se pur in modo semplicistico con la domanda "Chi è che richiede l'accesso?". Ossia l'utente che vuole accedere ad un

sistema, sia esso un computer o un server, deve "rispondere" a questa domanda. Nella maggior parte dei casi la relativa risposta è il login, username, user-id o la chiave pubblica, gli utenti quindi si possono classificare attraverso l'Identificazione personale o di gruppo e questa informazione può anche essere pubblica, non coperta cioè da vincolo di segretezza.

Il processo di autenticazione invece è riassumibile attraverso la domanda "Ti dimostro che sono io!", di fatto una volta stabilità l'identità di una persona, il sistema deve essere sicuro che l'utente sia quello che dice di essere e si può suddividere in due categorie ben distinte, come "qualcosa che uno sa", ossia un utente è identificato dalla conoscenza di una informazione segreta: una password, una chiave crittografica segreta, un PIN..., oppure "Qualcosa che uno ha", cioè un utente è identificato tramite il possesso di un oggetto fisico: smart card, token card... . Questo tipo di informazione deve essere assolutamente tenuta segreta e conservata con la massima accuratezza.

Per finire il processo di autorizzazione il quale rispecchia "Dove posso andare e cosa posso fare?", cioè una volta che il sistema ha acconsentito all'accesso, ora si tratta di stabilire "cos'è permesso fare", ossia a quali risorse o a quali dati si può accedere. Il tutto viene garantito con un controllo agli accessi, in base alle autorizzazioni precedentemente date al profilo dell'utente. Il sistema è pertanto in grado di stabilire quali operazioni consentire e quali vietare all'utente.



3.2 Filosofia SSO

Si parla di sistema basato su "Single Sign On" (SSO) quando le richieste di autenticazione non vengono direttamente gestite dal sistema stesso ma vengono ridirette verso un'altro sistema di autenticazione che ha precedentemente certificato le credenziali dell'utente connesso, senza quindi avere la necessità di richiedere nuovamente le credenziali per l'accesso.

Il nodo cruciale è l'ultima parte della frase: "*senza avere la necessità di richiedere le credenziali all'utente*", Moltissime volte gli utenti lanciano le varie applicazioni, magari come il gestionale aziendale, e al momento della richiesta delle credenziali, l'utente fornisce come username "Admin" e lascia la password vuota...e se fosse chiesto loro il motivo di questa azione, riceveremmo come risposta, che non è un problema perch'è li fanno tutti così e perché altrimenti il sistema non funziona. Analizzando più a fondo la situazione verremmo a conoscenza che il sistema funziona perfettamente ma la gestione e la manutenzione della sicurezza sono un compito che nessuno si è mai voluto accollare perché troppo oneroso, portando all'inevitabile conseguenza che la sicurezza viene vissuta come una fatica, come un'ulteriore ostacolo verso la realizzazione del proprio lavoro quotidiano.

L'obiettivo principe del Single Sign On è proprio quello di rendere la sicurezza trasparente all'utente finale, facilmente manutenibile e gestibile per gli amministratori; l'utente deve rendersi conto di lavorare in un sistema sicuro, ma non deve assolutamente vivere la sicurezza come un onere aggiuntivo.

A questo punto sorge una ovvia domanda “In che modo il “Single Sign On” ci può aiutare?”

Il Single Sign On è un sistema specializzato che permette ad un utente di autenticarsi una sola volta e di accedere a tutte le risorse informatiche alle quali è abilitato, semplificando la gestione delle password, in quanto maggiore è il numero della password da gestire, maggiore è la possibilità che saranno utilizzate password simili le une alle altre e facili da memorizzare, abbassando così il livello di sicurezza, semplificando la gestione degli accessi ai vari servizi e semplificando la definizione e la gestione delle politiche di sicurezza.

Vi sono tre approcci per la creazione di un sistema di SSO, l’approccio centralizzato, quello federativo ed infine quello cooperativo.

L’Approccio centralizzato è basato sul principio di disporre di un database globale e centralizzato di tutti gli utenti, e di centralizzare allo stesso modo la politica della sicurezza. Questo approccio è destinato principalmente ai servizi dipendenti tutti dalla stessa entità, per esempio all'interno di una azienda.

Utilizzando l’approccio federativo invece, differenti gestori ("federati" tra loro) gestiscono dati di uno stesso utente, quindi l'accesso ad uno dei sistemi federati permette automaticamente l'accesso a tutti gli altri sistemi. Questo approccio è stato sviluppato per rispondere ad un bisogno di gestione decentralizzata degli utenti: ogni gestore federato mantiene il controllo della propria politica di sicurezza.

Per finire, l'approccio cooperativo parte dal principio che ciascun utente dipenda, per ciascun servizio, da uno solo dei gestori cooperanti. In questo modo se si cerca, ad esempio, di accedere alla rete locale, l'autenticazione viene effettuata dal gestore che ha in carico l'utente per l'accesso alla rete. Come per l'approccio federativo, in questa maniera ciascun gestore gestisce in modo indipendente la propria politica di sicurezza. L'approccio cooperativo risponde ai bisogni di strutture istituzionali nelle quali gli utenti sono dipendenti da una entità, come ad esempio in università, laboratori di ricerca, amministrazioni, etc.

3.3 Software Realizzazione SSO

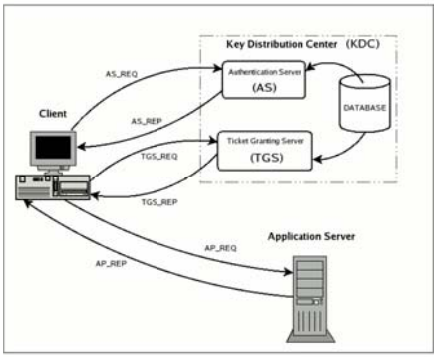
Kerberos è un protocollo per l'autenticazione creato dal Massachusetts Institute of Technology che consente una strong-authentication tra applicazioni client/server utilizzando crittografia a chiave segreta. Kerberos è la soluzione maggiormente adottata tra quelle esistenti (anche la Microsoft ne integra la versione 5 nei suoi prodotti). Kerberos è un protocollo di rete per

l'autenticazione tramite crittografia che permette a diversi terminali di comunicare su una rete informatica insicura provando la propria identità e cifrando i dati. È un meccanismo conosciuto alle applicazioni e serve per estrarre interamente le autenticazioni, gli utenti si registrano sul server Kerberos, il quale rilascia un "biglietto da visita", che il loro client software presenterà ai server a cui loro tenteranno di accedere. È disponibile per Unix, per Windows e per piattaforme di elaborazione dati, ma richiede ampie modifiche al codice dell'applicazione client/server, consente infine l'intercettazione e i replay attack e assicura l'integrità dei dati. I suoi progettisti mirarono soprattutto ad un modello client-server, e fornisce una mutua autenticazione, sia l'utente che il fornitore del servizio possono verificare l'identità dell'altro.

Kerberos si basa sulla crittografia simmetrica e richiede una terza parte affidabile, si basa quindi sul protocollo di Needham-Schroeder, utilizzando una terza parte affidabile per centralizzare la distribuzione delle chiavi detta Key Distribution Center (KDC), che consiste di due parti separate logicamente: l'Authentication Server (AS) e il Ticket Granting Server (TGS). Il suo funzionamento si basa sulla figura dei "biglietti" (detti ticket) che servono per provare l'identità degli utenti. L'AS mantiene un database delle chiavi segrete e ogni entità sulla rete, che sia un client o un server, condivide la chiave segreta solo con l'AS. La conoscenza di questa chiave serve per provare l'identità di un'entità e durante le comunicazioni Kerberos genera una chiave di sessione, che può essere utilizzata dai due terminali per comunicare.

Il protocollo può essere definito come segue utilizzando la notazione per protocolli di sicurezza, dove Alice (A) si autentica presso Bob (B) usando il server S. La sicurezza del protocollo si basa fortemente sui timestamp T e sui tempi di vita L come indicatori affidabili della creazione recente della comunicazione per evitare replay attack. È importante notare come il server S qui stia sia come Authentication Service (AS) che come Ticket Granting Service (TGS).

Di seguito verrà illustrata una descrizione semplificata del protocollo in cui saranno utilizzate le seguenti abbreviazioni:



AS = Authentication Server, **TGS** = Ticket Granting Server, **SS** = Service Server.

In breve.:

Il client si autentica presso AS che gli fornisce un ticket di sessione per accedere a TGS, si autentica presso TGS e riceve il ticket per aprire una sessione di comunicazione con SS.

In dettaglio.:

Autenticazione di base

1. Un utente inserisce username e password sul client.

Client: Autenticazione AS

1. Il client manda un messaggio non crittato all'AS richiedendo i servizi per l'utente. ("L'utente XYZ vorrebbe richiedere dei servizi"). Né la chiave segreta né la password vengono inviate all'AS.
2. L'AS controlla se il client è nel suo database. Se lo è, invia due messaggi al client:
    a. Messaggio A: Chiave di sessione client-TGS criptata usando la chiave segreta dell'utente.
    b. Messaggio B: Ticket-Granting Ticket (che include l'identificativo del client, l'indirizzo di rete, il tempo di validità del ticket e la chiave di sessione client-TGS) criptata utilizzando la chiave segreta di TGS.
3. Quando il client riceve i messaggi A e B, descritta il messaggio A ottenendo la chiave di sessione client-TGS. Questa chiave è utilizzata per le successive comunicazioni con TGS. (Nota: il client non può decriptare

il Messaggio B, che è stato criptato con la chiave segreta di TGS). A questo punto il client possiede i mezzi per autenticarsi presso TGS.

Client: Autenticazione TGS

1. Quando richiede dei servizi, il client invia i seguenti due messaggi a TGS:
    a. Messaggio C: composto dal Ticket-Granting Ticket (mandatogli da AS nel messaggio B) e dall'identificativo del servizio richiesto
    b. Messaggio D: autenticatore (Authenticator) (che è formato da identificativo del client e timestamp), criptato usando la chiave di sessione client—TGS.
2. Ricevendo i messaggi C e D, TGS decripta il messaggio C con la propria chiave e dal messaggio estrae la chiave di sessione client—TGS che utilizza per decriptare il messaggio D (autenticatore). A questo punto invia i seguenti due messaggi al client:
    a. Messaggio E: Ticket client-server (che include l'identificativo del client, l'indirizzo di rete del client, il periodo di validità e la chiave di sessione client-server) criptato utilizzando la chiave segreta del server che offre il servizio.
    b. Messaggio F: Chiave di sessione client-server criptato usando la chiave di sessione client-TGS.

Client: Autenticazione SS

1. Ricevendo i messaggi E e F dal TGS, il client può autenticarsi presso il SS. Il client si connette al SS e invia i seguenti due messaggi:
    a. Messaggio G: Ticket client-server criptato usando la chiave segreta di SS.

b. Messaggio H: un nuovo autenticatore, che include l'identificativo del client, il timestamp e è criptato usando la chiave di sessione client-server.

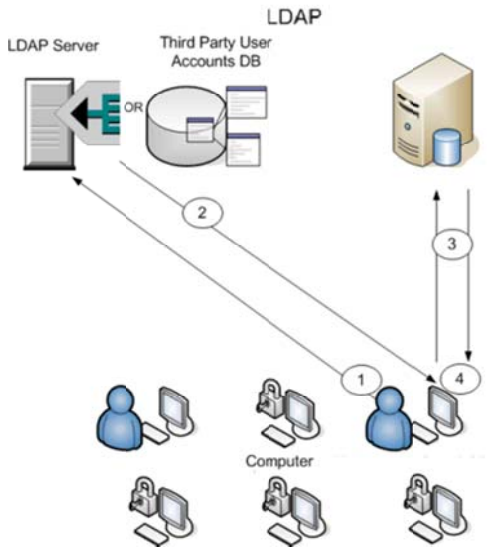
2. Il server decripta il ticket usando la sua chiave segreta e invia il seguente messaggio al client per confermare la propria identità e la volontà di fornire il servizio al client:
    a. Messaggio I: il timestamp trovato nell'autenticatore incrementato di uno, criptato utilizzando la chiave di sessione client-server.
3. Il client decripta la conferma usando la chiave di sessione client-server e controlla che il timestamp sia correttamente aggiornato. Se lo è, il client può considerare affidabile il server e iniziare a effettuare le richieste di servizio.
4. Il server fornisce i servizi al client.

3.3.1 LDAP

Lightweight Directory Access Protocol è un protocollo standard per l'interrogazione e la modifica dei servizi di directory. Una directory è un database specializzato ottimizzato per la lettura e per la ricerca dei dati attraverso filtri sofisticati, in cui il dato è formato da un attributo e la relativa descrizione (ad esempio email=rossi@azienda.it"). Il tipo di informazioni inserite in LDAP è basato sul modello delle Entry. Una entry è l'insieme di una serie di attributi relativi ad una chiave univoca detta Distinguish Name (DN) e ogni entry ha un tipo ed uno o più valori. I tipi sono generalmente stringhe mnemoniche come "cn" per Common Name o "mail" per indirizzi e-mail. La sintassi del valore dipende dal tipo di attributo dichiarato (per esempio cn può contenere il valore Mario Rossi, mentre il tipo jpegPhoto conterrà una fotografia in formato JPEG (binario)). Le informazioni sono organizzate secondo un modello gerarchico, in una maniera simile al DNS. Tipicamente hanno sotto di loro delle alberature di tipo organizational units, ovvero organizzazioni aziendali, People che contengono le persone oppure

Groups che contengono i gruppi. L'organizzazione di un'alberatura LDAP è molto flessibile e può essere gestita secondo esigenze specifiche, ad esempio applicative, ma in ogni caso deve essere ben pianificata. Tutti gli oggetti però devono avere uno o più attributi di tipologia objectClass. I valori di objectClass sono relativi a degli schemi a cui la entry deve obbedire: ad esempio una entry con un objectClass di valore person ha come obbligatorio il tipo sn (Surname), pertanto deve avere un attributo di tipo sn all'interno della entry. Abbiamo accennato precedentemente al Distinguish Name (DN): esso rappresenta il modo per referenziare univocamente una entry nel database LDAP(anche chiamata RDN). Tipicamente il DN è formato dal Common Name (CN), Organizational Unit (OU) e Base DN (la base dell'alberatura LDAP); LDAP definisce operazioni per interrogare e aggiornare il suo database. Le operazioni definite sono aggiunta, cancellazione, modifica e rinomina di una entry. Spesso LDAP viene usato per ricercare: la ricerca viene effettuata su parte dell'alberatura del database con dei criteri chiamati filtri. Ogni entry che corrisponde ai filtri determinati verrà restituita.

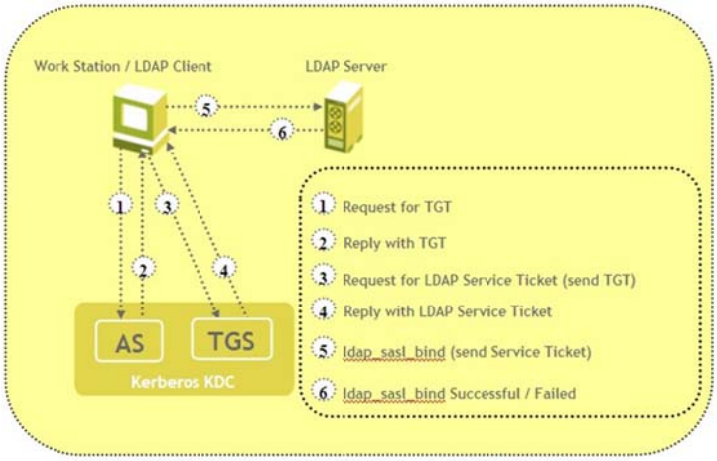
LDAP è un protocollo di tipo client/server. Uno o più LDAP server contengono i dati che formano il Directory Information Tree (DIT). I client si connettono al server e sottopongono una query. Il server risponde con le informazioni e/o con un puntatore da dove il client può richiedere informazioni ulteriori, tipicamente ad un altro LDAP. Quest'ultimo processo è detto di referral. In un modello simile, non importa a quale LDAP server il client sia collegato, in quanto tutti i server vengono collegati tra di loro similmente a quanto avviene nel sistema DNS. Questo è una feature per il concatenamento di più server



LDAP, ad esempio tra organizzazioni della stessa azienda oppure tra aziende diverse che cooperano tra di loro. Per quanto riguarda la sicurezza delle informazioni contenute in una directory, LDAP contempla un meccanismo di autenticazione tra un client e un server LDAP, inoltre permette anche la crittografia attraverso SSL. L'autenticazione di un client non è però l'autorizzazione dello stesso a quali dati può accedere.

3.3.2 Kerberos & LDAP

Un server LDAP, attraverso plugin, è in grado di usare una password back-end differente, ovvero la possibilità di accedere alla password associata ad un utente LDAP tramite Kerberos sfruttando la caratteristica di un LDAP server per poter effettuare il controllo della password sul repository di Kerberos. L'informazione all'interno di una directory è organizzata in elementi chiamati entry.



Gli elementi di una directory LDAP presentano una struttura gerarchica che riflette confini politici, geografici o organizzativi. Nella struttura ad albero, ad ogni livello esiste un Relative distinguished name (RDN) che lo identifica (ad esempio ou=people). L'unione di tutti gli RDN, presi in successione dal nodo foglia fino alla radice, costituisce il distinguished name (DN), una stringa che rappresenta univocamente una entry nella directory.

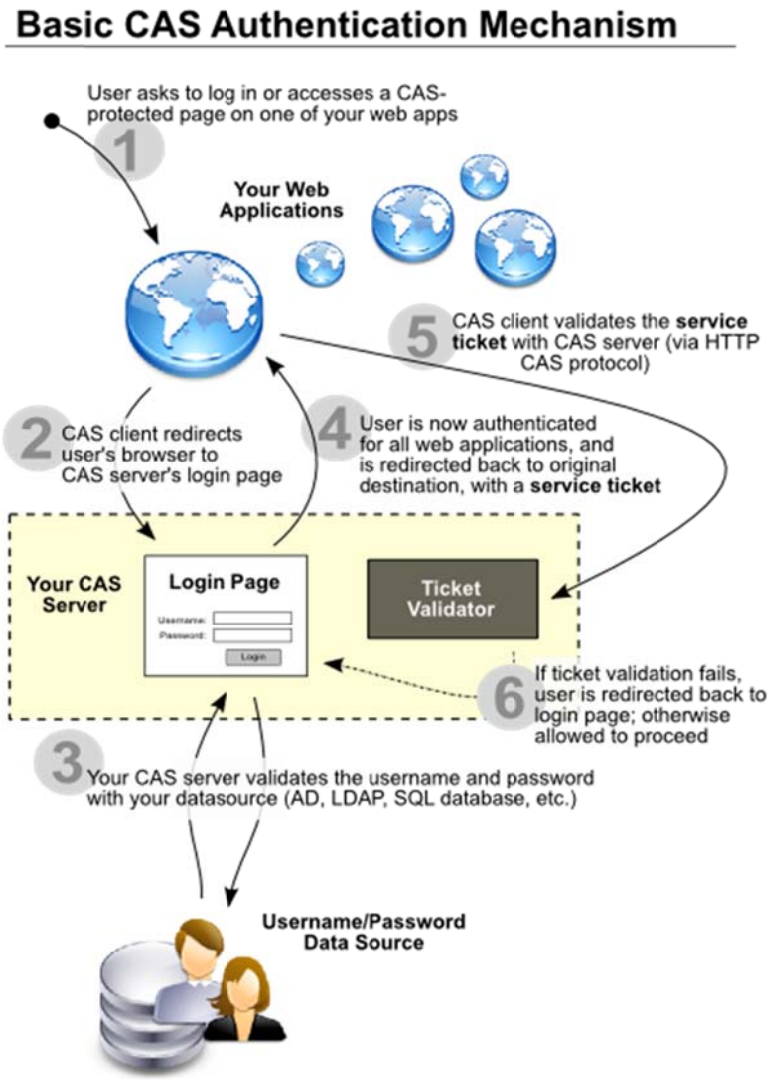
Un dn può essere ad esempio: "cn=John Doe , ou=people , dc=wikipedia , dc=org".

Ciascuna entry ha una serie di attributi, costituiti dall'associazione attributo-valore e per ogni attributo possono esserci più valori. Ognuno degli attributi dell'elemento è definito come membro di una classe di oggetti, raggruppati in uno schema. Ogni elemento nella directory è associato a una o più classi di oggetti, che definiscono se un attributo sia opzionale o meno, e che tipo di informazioni questo contenga. I nomi degli attributi solitamente sono scelti per essere facilmente memorizzabili, per esempio "cn" per common name, o "mail" per un indirizzo e-mail. I valori degli attributi dipendono dal tipo, e la maggioranza dei valori non binari sono memorizzati in LDAPv3 (LDAP versione 3) come stringhe UTF-8. Per esempio, un attributo di tipo mail potrebbe contenere il valore "user@example.com", mentre un attributo "jpegPhoto" potrebbe contenere una fotografia nel formato (binario) JPEG.

3.3.3 Ja Sig Cas

Il JA-SIG Central Authentication Service (CAS) è un servizio Single Sign On libero che permette alle applicazioni web la possibilità di rinviare tutte le autenticazioni a un server centrale o a più server di fiducia. Numerosi client sono disponibili gratuitamente, inclusi client per Java, Microsoft.Net, PHP, Perl, Apache, uPortal, Liferay e altri. Il suo funzionamento di base prevede un server ed un client come illustrato nell'immagine seguente,

1. L'utente tenta di accedere ad una pagina protetta.
2. Il CAS client reindirizza il browser alla pagina di login sul CAS server.
3. Il CAS server valida username e password su un data source.
4. Il browser viene rimandato alla pagina richiesta inizialmente con un service ticket.
5. Il service ticket garantisce la persistenza delle credenziali su tutte le altre applicazioni.

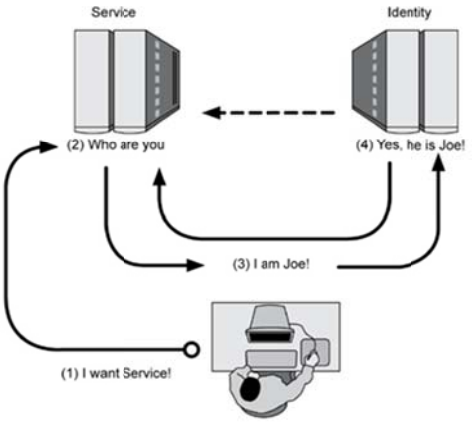


Datasource

La forza di questo protocollo risiede nella possibilità di utilizzare qualsiasi datasource per la gestione degli utenti. Può essere usato un DBMS, Active Directory, un servizio LDAP, un fileXML etc. Gli utenti possono essere addirittura autenticati con i propri account Gmail**.** Il controllo può essere anche effettuato su più datasource ordinati per priorità; ad esempio possiamo controllare le credenziali dell'utente sul un nostro database e, in caso di autenticazione fallita, proseguire il controllo su Gmail.

3.3.4 Saml

L'OASIS Security Services Technical Committee (SSTC), ha lo scopo di definire un framework XML finalizzato allo scambio di informazioni di autenticazione e di autorizzazione. Saml (Security assertion markup language) è uno standard chiave per l'utilizzo dei servizi Web nelle applicazioni commerciali. Sfruttando tecnologie Xml, Saml fornisce infatti un framework per le operazioni di autenticazione e autorizzazione dei servizi Web. Permette, inoltre, di implementare funzionalità di single-sign-on ed è quindi una tecnologia decisiva nella gestione delle identità digitali online.



Inserire il supporto per Saml all'interno dei propri servizi Web è abbastanza facile in quanto le specifiche sono infatti molto semplici e chiare e chiunque sappia scrivere un servizio Web Xml potrà facilmente usare Saml per le funzioni di autenticazione.

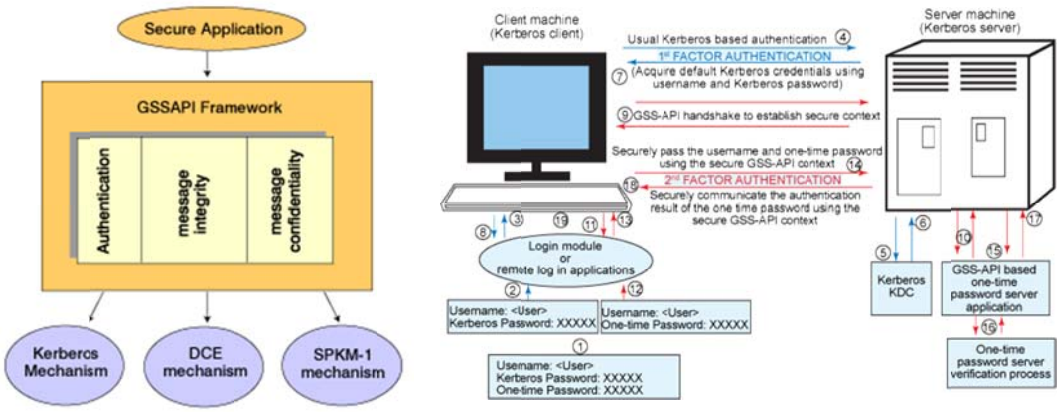
Nella sua forma più elementare, Saml associa un'identità digitale (come un indirizzo di posta elettronica o la voce di una directory) a un soggetto (un utente o un sistema) e definisce i suoi diritti di accesso all'interno di uno specifico dominio. Uno dei maggiori punti di forza di Saml è l'ottima capacità d'interazione con tutti i tipi di sistemi disponibili. Nelle procedure di autenticazione, per esempio, supporta password tradizionali, token hardware, chiavi pubbliche e certificati digitali. In più, presenta un supporto integrato per le firme Xml, che permette non solo di autenticare un messaggio, ma anche di garantirne l'autenticità e impedire che il mittente lo respinga.

Il meccanismo di trasporto predefinito per Saml è il protocollo Soap (Simple object access protocol) su connessioni Http. È la scelta più logica, perché lo standard si concentra soprattutto sui servizi Web, ma la tecnologia di base Xml permette a Saml di utilizzare facilmente anche tutti gli altri meccanismi di

trasporto disponibili. Un'autorità di autenticazione Saml non si limita a emettere dichiarazioni di autenticazione, ma ne può anche ricevere. Questo permette di sfruttare Saml anche per sistemi di single-sign-on perché, quando un utente si autentica ed esegue qualche azione all'interno di un dominio, l'autorità Saml conosce anche tutte le sue autenticazioni e autorizzazioni precedenti. Tuttavia, Saml non è supportato da Microsoft, che finora si è concentrata su altri meccanismi di single-sign-on per i servizi Web, per esempio Passport.

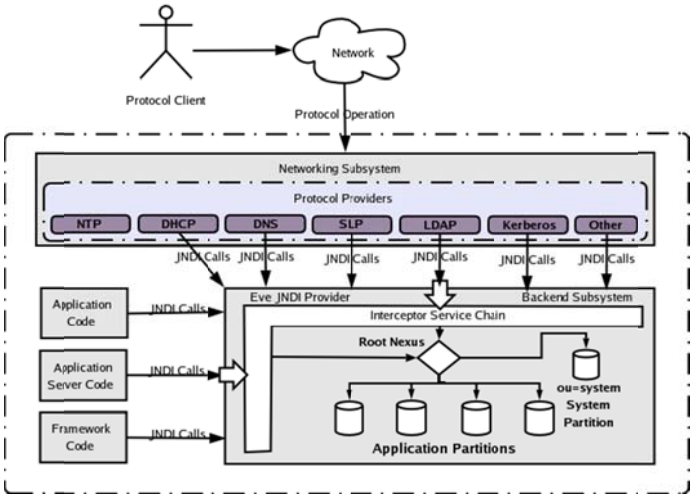
3.3.5 GSSAPI

Generic Security Service Application Programming Interface (GSSAPI) è un'interfaccia standardizzata astratta che fornisce una autenticazione generica e un'interfaccia di messaggistica sicura in base alle quali possono essere collegati altri meccanismi di sicurezza. Un meccanismo GSSAPI comune è il meccanismo di Kerberos che si basa sulla crittografia a chiave segreta. GSSAPI offre una interfaccia standard generica attraverso la quale una domanda di sicurezza può utilizzare molteplici meccanismi di sicurezza di base.

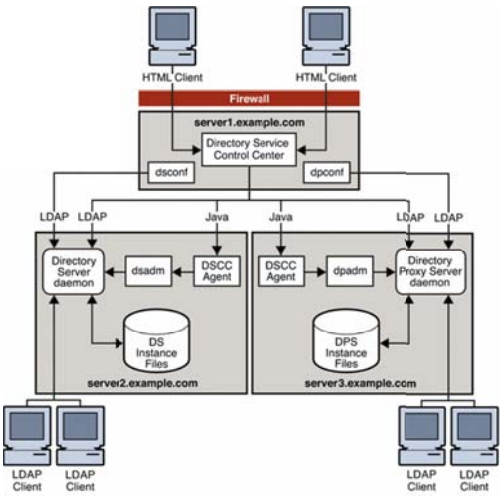


3.3.6 Apache Directory Server

Apache Directory Server è un server che fornisce directory interamente scritte in Java, che è stata certificata LDAPv3 compatibili con Open Group. Oltre LDAP supporta Kerberos 5 e il Protocollo di Cambio Password. È stato progettato per introdurre trigger, procedure stored, le code e le viste al mondo di LDAP che era privo di questi costrutti.



3.3.7 RedHat Directory Server



Red Hat Directory Server è un server basato su LDAP che centralizza le impostazioni delle applicazioni, profili utente, i dati del gruppo, le politiche e le informazioni di controllo di accesso in un sistema operativo indipendente. Semplifica la gestione degli utenti, eliminando la ridondanza dei dati e l'automazione dei dati di manutenzione. Migliora anche la sicurezza, memorizzando le politiche e le informazioni di controllo di accesso, Red Hat Directory Server crea un'unica fonte di autenticazione in tutta l'intera impresa sia per applicazioni Intranet ed Extranet. Centralizza la gestione delle persone e dei

loro profili, riducendo i costi amministrativi. Funge da repository centrale per i profili utente e delle preferenze, che permette la personalizzazione. Consente single sign-on di accesso con una soluzione di un partner. Fornisce supporto completo per i 64 bit di HP-UX, Solaris e Red Hat Enterprise Linux. Fornisce le basi per un forte certificato di autenticazione basato su quando usato in unione a Red Hat Certificate System.

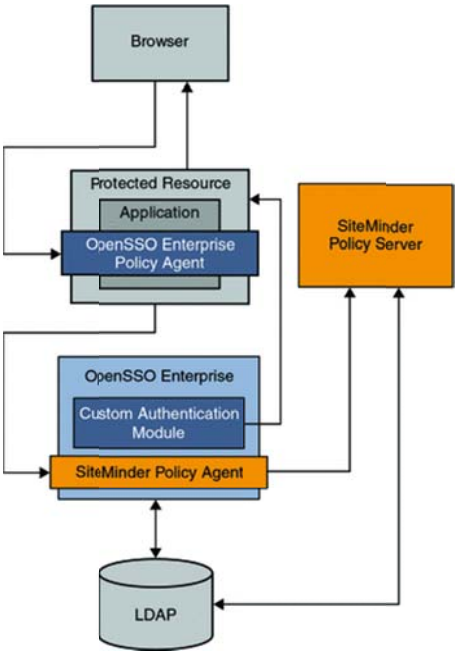
3.3.8 SiteMinder

SiteMinder fornisce l'autenticazione basata come single sign-on su tutte le applicazioni web-based.

Consente di gestire e implementare applicazioni Web protette per .:

- Aumentare le nuove opportunità di business
- Gestire i costi
- Migliorare la sicurezza per ridurre il rischio
- Semplificare la conformità

SiteMinder assicura livelli ineguagliati di affidabilità, disponibilità, scalabilità e



gestibilità ed inoltre dispone di uno standard per la gestione degli accessi Web di classe enterprise, è anche un componente fondamentale della soluzione CA Identity & Access Management, che automatizza l'amministrazione delle identità degli utenti e assicura che solo gli utenti autorizzati possano accedere alle risorse IT critiche, dal Web al mainframe. Assicura la gestione centralizzata della sicurezza di cui l'organizzazione necessita per autenticare gli utenti e controlla l'accesso alle applicazioni Web e ai portali. In ambito Internet e Intranet, consente una

fornitura protetta di informazioni e applicazioni essenziali per dipendenti, partner, fornitori e clienti. SiteMinder è usato in congiunzione con IdentityMinder, che gestisce profili utente dettagliati, e TransactionMinder, che fornisce l'accesso ai servizi web.

Funzionalità principali .:

1. SINGLE SIGN-ON (SSO)

   SiteMinder elimina il problema dei login utente multipli consentendo l'utilizzo di un unico set di credenziali per un accesso immediato a tutta una serie di applicazioni Web, portali e domini di sicurezza.

   La funzionalità SSO di SiteMinder consente inoltre l'accesso diretto alle applicazioni aziendali, quali SAP, Siebel, PeopleSoft e Oracle.

   Il componente SSO aziendale della soluzione Identity & Access Management (IAM), permette agli utenti di eseguire un unico processo di autenticazione per accedere sia alle applicazioni Web protette da SiteMinder, sia alle applicazioni non Web con accesso controllato da SSO.

2. GESTIONE DELLE AUTENTICAZIONI COMPLESSE

   SiteMinder realizza una strategia di autenticazione unificata per assicurare alle applicazioni Internet e Intranet il giusto livello di protezione. In questo modo è possibile proteggere le applicazioni di valore più elevato mediante metodi di autenticazione più complessi e le applicazioni di valore inferiore con approcci più semplici basati su nome utente e password. SiteMinder fornisce supporto per la gestione degli accessi per molti sistemi di autenticazione (tra cui password, token, certificati X.509, smartcard, modelli personalizzati, biometrica) e diverse combinazioni di metodi di autenticazione. Inoltre, SiteMinder abilita l'integrazione del contesto di autenticazione nelle policy di autorizzazione. Ad esempio, quando una particolare organizzazione utilizza due metodi di autenticazione (ad esempio

nome utente/password e token per la generazione di password monouso) e l'utente esegue l'autenticazione utilizzando la password monouso, a tale utente possono essere concessi privilegi aggiuntivi per le applicazioni.

3. AMMINISTRAZIONE E VERIFICA CENTRALIZZATA BASATA SU POLICY

   SiteMinder centralizza la gestione degli accessi di clienti, partner e dipendenti alle applicazioni Web dell'azienda. In questo modo si elimina la necessità di logiche di protezione ridondanti e specifiche per singole applicazioni. Le policy di SiteMinder sono regole o gruppi di regole che autorizzano o negano l'accesso a una risorsa. L'accesso può essere limitato da attributi utente, ruoli, gruppi e gruppi dinamici e determinato su base geografica e temporale. L'autorizzazione può essere attribuita a livello di file, pagina o oggetto. Sempre tramite le policy è possibile definire una "assunzione di privilegi" controllata, in cui un utente autorizzato, ad esempio un rappresentante dell'assistenza clienti, può accedere alle risorse cui un altro utente non è autorizzato ad accedere. L'autorizzazione dinamica richiama policy di sicurezza che valutano in tempo reale i dati provenienti da una serie di fonti locali o esterne, compresi i servizi Web e i database, per stabilire se autorizzare o rifiutare l'accesso. La valutazione contestuale permette di conferire maggiore granularità al processo di autorizzazione. È possibile, ad esempio,limitare l'accesso a un'applicazione Web (un determinato servizio di banking) ai soli clienti che soddisfano specifici criteri (un saldo minimo). Le policy di autenticazione possono essere applicate anche parallelamente a sistemi esterni, quali i fornitori di servizi per la gestione del rischio.

4. GESTIBILITÀ AZIENDALE

SiteMinder fornisce strumenti di gestione dei sistemi di classe enterprise, che consentono al personale addetto di monitorare, gestire e mantenere ambienti multipli, ad esempio ambienti di sviluppo, collaudo e produzione, in modo più efficace. Questi strumenti di gestione sono .:

- Installazioni non sorvegliate
- Monitoraggio operativo
- Aggiornamenti continui
- Gestione centralizzata degli agenti Web
- Migrazione delle policy di sicurezza
- Interfaccia di scripting

5. SCALABILITÀ E AFFIDABILITÀ

SiteMinder offre la scalabilità necessaria per soddisfare i requisiti di sicurezza di classe enterprise, sia in termini di numero di utenti che in termini di numero di risorse protette, assicurando la possibilità di gestire la crescita, anche quella derivante da acquisizioni o partnership. Con SiteMinder è possibile distribuire applicazioni aziendali critiche a una base di diversi milioni di utenti, con la certezza che le sue prestazioni sono state verificate mediante test indipendenti per garantire velocità di transazione, affidabilità e gestibilità nettamente superiori rispetto alle soluzioni della concorrenza. Affidabilità, disponibilità e scalabilità sono supportate da diverse funzionalità, tra cui:

- Bilanciamento dinamico del carico elaborativo
- Caching a due livelli
- Clustering di server di policy e failover cluster-to-cluster
- Replica di archivi di policy e di utenti
- Supporto di server SMP a 4 e 8 vie

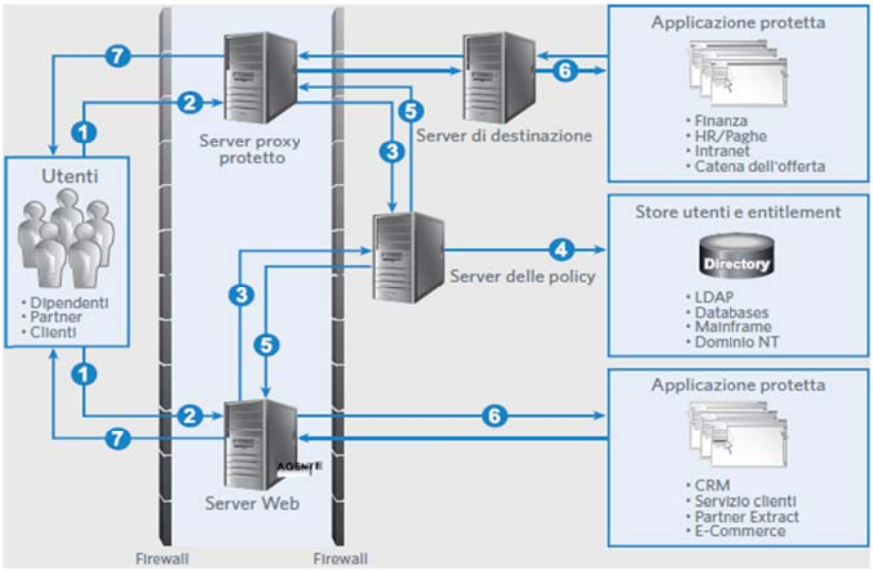
6. SERVIZI DI SICUREZZA FEDERATI

SiteMinder fornisce anche una funzionalità integrata dal punto di vista architettonico e concessa separatamente in licenza, che assicura la federazione basata su browser, in cui gli utenti possono passare in modo protetto da siti interni a siti Web ospitati da partner e clienti. Gli utenti di SiteMinder possono svolgere entrambi i ruoli (ospitante oppure ospitato) in qualunque sito Web nell'ambito di una federazione. Questa capacità di federazione, denominata SiteMinder Federation Security Services (FSS), fornisce una piattaforma flessibile e robusta per la federazione delle identità, consentendo alle organizzazioni di sfruttare i vantaggi derivanti dal collegamento di applicazioni di business distribuite in diversi domini, senza compromettere la sicurezza. FSS supporta un insieme completo di standard di federazione, tra cui Security Assertion Markup Language (SAML) eWS-Federation/Microsoft ADFS. Il prodotto abilita una federazione completa e bidirezionale con la massima interoperabilità tra le entità che ne fanno parte.

Di seguito verranno descritte le varie fari che portano ad una procedura di accesso sicuro alle applicazioni Web:

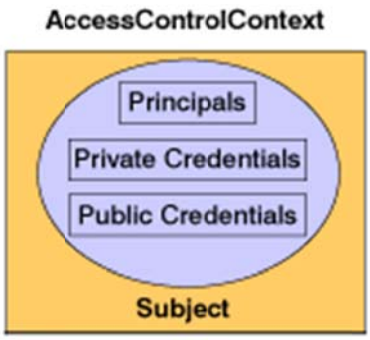
1. L'utente tenta di accedere a una risorsa protetta.
2. All'utente viene richiesto di fornire le credenziali. Questi le presenta all'agente Web CA SiteMinder o al server proxy protetto.
3. Le credenziali dell'utente vengono passate al server che gestisce i criteri di sicurezza.
4. L'utente viene autenticato mediante un controllo nell'archivio utenti appropriato.
5. Il server di policy valuta i privilegi dell'utente e concede l'accesso.
6. Le informazioni sul profilo e i privilegi dell'utente vengono passati all'applicazione.

7. L'utente ottiene l'accesso all'applicazione protetta, che fornisce un contenuto personalizzato



3.3.9 Jaas

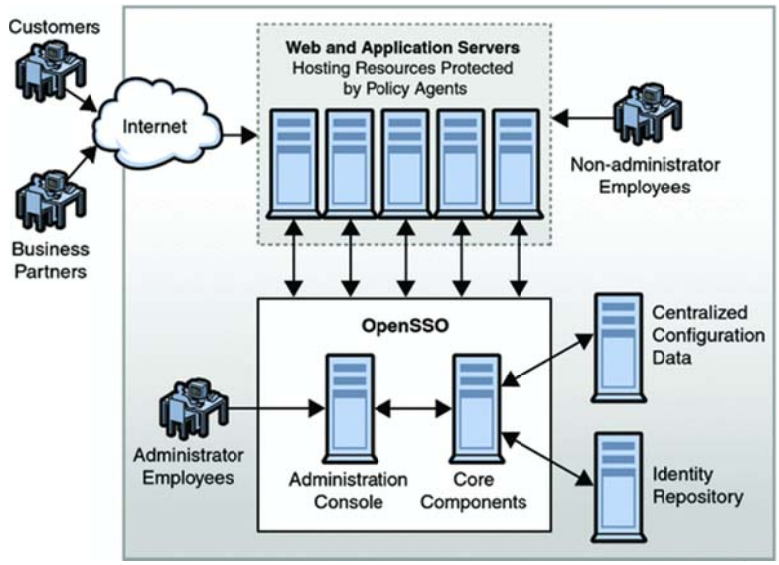
Java Authentication and Authorization Service (JAAS) è un insieme di librerie



e funzionalità standard per implementare processi di autenticazione e di autorizzazione in modo indipendente dalla piattaforma. L'architettura basata su provider consente di aggiungere o aggiornare meccanismi di autenticazione e autorizzazione senza modificare l'applicazione che ne fruisce. JAAS può essere utilizzato in applicazioni stand-alone, in Applet Java, Enterprise JavaBean (EJB) o Servlet ed è indipendente dall'application server. Consente di implementare meccanismi di Single Sign-On in ambienti eterogenei e supporta il controllo di accesso basato su utenti, gruppi e ruoli. L'utilizzo di JAAS è ora considerato in Java 5 come la tecnologia standard per implementare e utilizzare processi di autenticazione e autorizzazione.

3.3.10 OpenSSO

Il progetto Open Web SSO (OpenSSO) fornisce servizi di identità di base per semplificare l'implementazione di single sign-on (SSO) come una componente di sicurezza in una infrastruttura di rete. OpenSSO fornisce le basi per l'integrazione di applicazioni web diverse che potrebbero operare in genere su diverse piattaforme, come web server e application server. Questo progetto si basa sul codice di base di Sun Java ™ System Access Manager, un prodotto di base dell'infrastruttura identità offerti da Sun Microsystems.



3.3.11 Microsoft SSO

Enterprise Single Sign-On (SSO). C**i** sono diversi tipi di Single Sign-On oggi disponibili .:

1. Integrated Windows Single Sign-On

   Questi servizi consentono di connettersi a più applicazioni all'interno della vostra rete ed utilizzano un meccanismo di autenticazione comune e di verifica di credenziali dopo di che utilizzano le credenziali per determinare le azioni che si possono eseguire, sulla base di diritti utente. Ad esempio, se l'integrazione delle applicazioni utilizzando Kerberos, dopo che il sistema autentica le credenziali utente, è possibile accedere a qualsiasi risorsa di rete che è integrata con Kerberos.

2. Extranet Single Sign-On (SSO Web)

   Questi servizi consentono di accedere alle risorse su Internet utilizzando un unico set di credenziali utente. L'utente fornisce un insieme di credenziali per accedere a diversi siti Web che appartengono a organizzazioni diverse. Un esempio di questo tipo di Single Sign-On è il Microsoft Passport Network per il mercato consumer-based. Per gli scenari federate, Active Directory Federation Services consente di Web SSO.

3. Server-Based Intranet Single Sign-On. Questi servizi permettono di integrare molteplici applicazioni eterogenee e sistemi in ambiente enterprise. Queste applicazioni e sistemi non possono utilizzare l'autenticazione comune. Ogni applicazione ha il suo gruppo di directory utente. Ad esempio, un'organizzazione Windows utilizza il servizio Active Directory per autenticare gli utenti, e usa il mainframe IBM Resource Access Control Facility (RACF) per autenticare gli utenti stessi.

   La sincronizzazione delle password semplifica la gestione del database SSO, e mantiene le password in sincronia tra le directory degli utenti. È possibile effettuare questa operazione utilizzando schede di sincronizzazione di password, che è possibile configurare e gestire utilizzando gli strumenti di sincronizzazione di password.

4. Enterprise Single Sign-On fornisce servizi per immagazzinare e trasmettere le credenziali utente criptati attraverso i confini locali e di rete, compresi i confini del dominio. Memorizza le credenziali SSO nel database di credenziali e fornisce un segnale generico unico sulla soluzione.

Il sistema Single Sign-On è costituito da un database di credenziali, un master server segreto, e uno o più Single Sign-On server. Il sistema di SSO contiene domande di affiliazione che un amministratore definisce. Una domanda di affiliazione è un'entità logica che rappresenta un sistema o sub-sistema, come un ospite, sistema di back-end, o line-of-business a cui ci si connette utilizzando Enterprise Single Sign-On. Ogni domanda di affiliazione è mappatura per utenti multipli, ad esempio, ha il mapping tra le credenziali di un utente in Active Directory e le loro credenziali RACF corrispondenti. Il database di credenziali è il database di SQL Server che memorizza le informazioni circa le domande di affiliazione, così come tutte le credenziali crittografate utente a tutte le domande di affiliazione.

Cap. 4

Implementazione SSO

4.1 Introduzione

Per effettuare i seguenti progetti di sviluppo con lo scopo di integrare software diversi e dalle funzionalità molto diversificate tra loro, è necessario preparare una una macchina virtuale su cui installare un sistema operativo, che verrà utilizzato come base per gli applicativi che saranno successuvante presi in esame. Negli sviluppi che verranno dettagliatamente descritti successivamente, è stato utilizzato il prodotto VMware Workstation per la crezione della Virtual Machine e Linux CentOS 32 Bit in lingua inglese come sistema operativo di base. Un'importante considerazione riguarda il nome che viene associato alla macchina virtuale in quanto durante la fase di configurazione è sconsigliato l'utilizzo dell'indirizzo "localhost" ma l'utilizzo invece dell'effettivo nome assegnato. Risulta necessario quindi assegnare al server un nome che userò per i certificati e per le configurazioni, e in questo caso sarà *"hostname" (hostname.tv.smc).*

4.2 Ja Sig Cas

Il CAS (Central Authentication Service) è sviluppato dalla comunità di JA-SIG, molti prodotti di Single Sign On per applicazioni web, e tra questi, CAS, sfruttano in modo intelligente due meccanismi del protocollo HTTP: i *cookie* e i *redirect*.

HTTP è un protocollo *connectionless*, ovvero ogni singola richiesta a un URL è a sé stante, senza relazioni con le richieste precedenti o future,proprio per questo motivo sono stati inventati i cookie, degli identificativi univoci e difficilmente indovinabili da parte di un attaccante, che vengono rilasciati da un web server al browser dell'utente finale.

Una volta che un utente ha ottenuto un cookie, lo ripresenterà nelle richieste successive allo stesso web server rilasciante, in modo tale che questi possa riconoscere l'utente e mantenere quindi una sessione di navigazione. Tipicamente, ogni applicazione Web ha il suo modo di gestire le sessioni e ha il proprio archivio di nomi utenti e password. Per motivi di sicurezza i cookie non possono essere condivisi tra domini e/o webserver diversi ed è proprio in questo contesto che CAS si inserisce, secondo il diagramma illustrato in Figura 1, diventando una sorta di collante per cookie tra domini e webserver diversi.

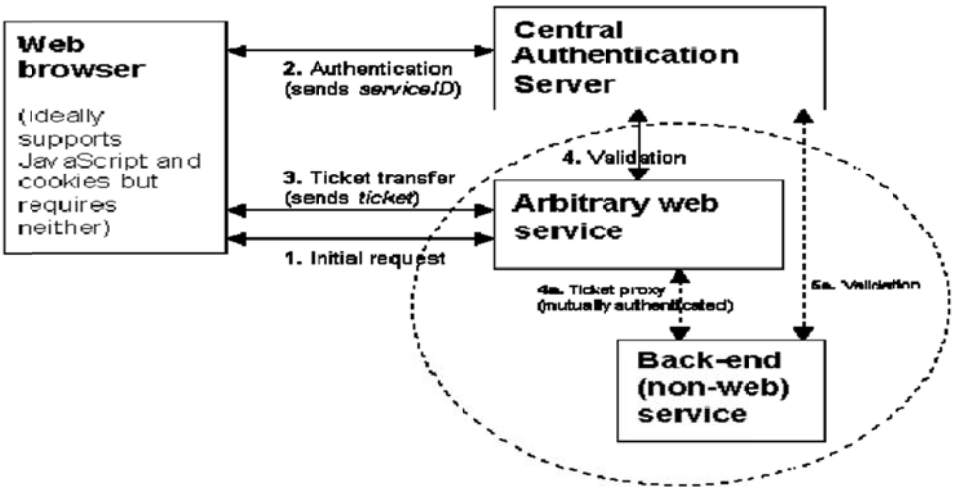


Fig. 1 - Schema di funzionamento di CAS

L'utente che si collega con un'applicazione Web e non ha un cookie valido viene reindirizzato automaticamente verso il server CAS. Si noti che al server CAS viene passato, tramite il parametro *service*, l'indirizzo cui rimandare l'utente in caso si autentichi con successo. L'utente dovrà quindi inserire le proprie credenziali, username e password, che il server CAS valida appoggiandosi a LDAP (ma è possibile utilizzare backend diversi).

Se l'autenticazione ha successo, il server CAS intraprende due azioni:

- Rilascia al Web browser dell'utente un cookie valido in modo tale che l'utente risulti già autenticato,
- Fa un HTTP redirect verso l'url specificata dal parametro service, aggiungendo un parametro *ticket* valido una-tantum.

A questo punto l'applicativo iniziale legge questo ticket e va a chiedere al CAS server se è valido (con una richiesta https) e in caso affermativo, il CAS server restituisce al servizio Web lo username dell'utente che si è autenticato. Da questo momento in poi, l'applicazione può proseguire come per ogni normale servizio non CASsizato, ovvero rilasciare all'utente un proprio cookie, in modo tale da riconoscerlo in successive richieste. Si noti che non è strettamente necessario implementare meccanismi di crittografia su transito in rete dei ticket, in quanto questi ultimi hanno validità per un solo tentativo e comunque dei limiti temporali di validità. È invece importante che il servizio Web validi il ticket presentandolo al CAS server utilizzando il protocollo cifrato https: se qualcuno fosse in grado di fare un attacco di tipo *Man In The Middle*, potrebbe banalmente fingere risposte valide indipendentemente dal ticket presentato dal servizio Web (o per contro fornire risposte non positive a fronte di ticket validi, creando di fatto un disservizio). Notiamo infine che, se l'utente avesse necessità di autenticarsi su un altro applicativo Web, quello che succede è ancora un redirect verso il CAS server che, in virtù del proprio cookie, lo riconosce immediatamente e non gli chiede username e password ; di fatto, l'utente viene rediretto al secondo applicativo Web con un nuovo ticket temporaneo, che verrà validato come nello schema descritto.

È possibile customizzare il server CAS:

- Aggiungendo nuovi backend di autenticazione, basati per esempio su database;
- Aggiungendo metodi di autenticazione per esempio basati su IP;
- Personalizzando le informazioni ritornate dal CAS server all'applicativo Web.

L'implementazione originale restituisce soltanto una risposta positiva e il nome utente ma è possibile far restituire altre informazioni, quali i gruppi di appartenenza, i livelli di autorizzazione e così via; funzionalità non previste dal protocollo CAS che, per definizione, si occupa solo di autenticazione, e non di autorizzazione. Tramite CAS e un backend LDAP è possibile avere un sistema

altamente affidabile in grado di accompagnare un'intera giornata tipo di un utente. Un utente può autenticarsi al mattino su CAS fornendo una volta sola le proprie credenziali (username e password) e per tutta la giornata può muoversi da un'applicazione all'altra senza dover fornire nuovamente la propria password. Le possibilità sono quindi molteplici, anche se richiedono un certo sforzo implementativo e di progettazione, soprattutto per quanto riguarda gli aspetti di autorizzazione (utenti diversi hanno ruoli e possibilità diverse a seconda delle applicazioni cui si collegano).

4.2.1 Predisposizione del software necessario

Risulta necessario il download di alcuni pacchetti software che utilizzeremo durante la procedura di integrazione, quali:

- Apache-tomcat-6.xx
- Xerces-J-bin..9.1
- Cas-Server-3.3.5-release
- Cas-Client-3.1.10-release
- Un Bundle-With-Tomcat di Liferay Portal
- Java Development Kit (JDK)

Procedo quindi con la fase iniziale di creazione degli utenti, creo l'utente *jasigcas* con password *"password"* che verrà utilizzato per far lavorare il Server Cas, successivamente creo l'utente liferay con password *"password"* che verrà utilizzato per avviare il portale Liferay dopo di che installo il Java Development Kit 6 (jdk.6.0_16) e imposto la variabile JAVA_HOME in *$BASE_DIR/apps/jdk1.6.0_16* e aggiungo *$JAVA_HOME/bin* alla variabile *$PATH* terminando quindi con la modifica del file *$HOME/.bash_profile* per impostare le variabili d'ambiente.

Come utente jasigcas installo il tomcat su cui lavorerà il CAS

```
cd  /home/jasigcas/
unzip apache-tomcat-6.0.24.zip
```

Devo modificare il file server.xml del tomcat che trovo in */home/jasigcas/apache-tomcat-6.0.24/conf/server.xml* per abilitare il canale ssl, rimuovo i commenti ed aggiungo i riferimenti ai certificati.

```
<! - Definire un SSL HTTP/1.1 Connector sulla porta 8443 ->
<Connector port = "8443" maxHttpHeaderSize = "8192"
MaxThreads = "150" MinSpareThreads = "25" MaxSpareThreads = "75"
enableLookups = "false" disableUploadTimeout = "true"
acceptCount = "100" scheme = "https" secure = "true"
= "false clientAuth" sslProtocol = "TLS" />
```

Modifico le porte .: *8080 – 8443 – 8009 in 8081-8444-8010* dopo di che avvio il Server Tomcat :

```
cd /home/jasigcas/apache-tomcat-6.0.24/bin
./startup.sh
```

Verifico che sia andato tutto bene dal file di log

```
cd $HOME
tail -f /home/jasigcas/apache-tmcat-6.0.24/logs/catalina.out
```

e dal fatto che aprendo un browser e puntando http://hostname.tv.smc:8081 mi appare la maschera base di tomcat. Spengo a questo punto il Server Tomcat

```
cd  /home/jasigcas/apache-tomcat-6.0.24/bin
./shutdown.sh
```

4.2.2 Attivazione di un Server Cas

4.3 Attivazione di un server CAS

Inizieremo con la creazione di JA-SIG CAS su server Tomcat 6.xx ossia copio dal Zip *"cas-server-3.5.5-release. zip"*, dalla cartella *"cas-server-3.3.5/modules"* il file *"cas-server-webapp-3.5.5.war"* in */home/jasigcas/apache-tomcat-6.0.24/webapps/.* Rinomino il file *cas-server-webapp-3.3.5.war* in *cas-web.war* ed elimino il contenuto delle cartelle logs e work (per avere uno scenario pulito).

Avviare il server Tomcat e dopo averlo spento si può notare che che si è creata in automatico una nuova cartella cas-web all'interno della Directory

Webbapps. Aggiungo nella cartella */home/jasigcas/apache-tomcat-6.0.24/webapps/examples/WEB-INF/lib*, i .jar che mi servono e sono presenti nei seguenti file .:

- All'interno del file *"cas-client-3.1.10-release.zip"*, nella cartella *"cas-client-3.1.10/modules"*, recupero:
  - *cas-client-core-3.1.10.jar*
  - *commons-logging-1.1.jar*
  - *xmlsec-1.3.0.jar*
- All'interno del file *"Xerces-J-bin.2.9.1.zip"*, nella cartella *"xerces-2_9_1"*, recupero:
  - *xercesImpl.jar*
  - *xml-apis.jar*

Modifico il file */home/jasigcas/apache-tomcat-6.0.24/webapps/examples/WEB-INF/web.xml* per aggiungere i filtri CAS, e praticamente è necessario inserire alla fine del file:

Il filtro "AuthenticationFilter" che ha il compito di verificare se l'utente deve essere autenticato o meno per accedere alla risorsa richiesta. Se deve essere autenticato l'utente viene mandato alla pagina di login.

```
<filter>
<filter-name>CAS Authentication Filter</filter-name>
<filter-
class>org.jasig.cas.client.authentication.AuthenticationFilter</fil
ter-class>
<init-param>
<param-name>casServerLoginUrl</param-name>
<param-value>https://hostname.tv.smc:8444/cas/login </param-value>
</init-param>
</filter>

<filter>
<filter-name>CAS Authentication Filter</filter-name>
<filter-
class>org.jasig.cas.client.authentication.AuthenticationFilter</fil
ter-class>
<init-param>
<param-name>casServerLoginUrl</param-name>
<param-value>https:// hostname.tv.smc:8444/cas/login </param-value>
</init-param>
```

```
<init-param>
<param-name>service</param-name>
<paramvalue>http://hostname.tv.smc:8081/examples/servlets/servlet/H
elloWorldExample </param-value>
</init-param>
<init-param>
<param-name>serverName</param-name>
<param-value> hostname.tv.smc:8081</param-value>
</init-param>
</filter>
```

Il filtro "HttpServletRequestWrapperFilter" che ha il compito di restituire i dati CAS come risultato delle richieste getRemoteUser e getPrincipalsss

```
<filter>
<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
<filter-
class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</fi
lter-class>
</filter>
```

Il filtro "AssertionThreadLocalFilter" che ha il compito di fornire i dati CAS all'applicazione "CASify" anche attraverso il ThreadLocal nel caso l'applicazione non riesca ad usare la HttpServletRequests

```
<filter>
<filter-name>CAS Assertion Thread Local Filter</filter-name>
<filter-
class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-
class>
</filter>
```

Ed infine devo aggiungere il mapping, ovvero indicare per quali risorse applicare i filtri indicati.

```
<filter-mapping>
    <filter-name>CAS Authentication Filter</filter-name>
    <url-pattern>/servlets/servlet/HelloWorldExample</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>CAS Validation Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>CAS HttpServletRequest Wrapper Filter</filter-
name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>CAS Assertion Thread Local Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

4.3.1 Generare il CERT SSL con Keytool Java

Ora bisogna generare un Certificato SSL per il server CAS come Utente jasigcas, creo quindi un certificato con il seguente comando

```
Keytool -genkey -alias tomcat -keypass changeit
```

Rispondo quindi alle domande : (NOTE: Il Nome e il Cognome DEVE essere hostname del server e non può essere un indirizzo IP, questo è molto importante, come un indirizzo IP, mancherà di verifica client hostname, anche se è corretto). E' importante indicare *(hostname.tv.smc)*, ovvero il nome completo del server, come nome e cognome.

```
Inserisci password del keystore: changeit
Immettere nuovamente la nuova password: changeit
Qual è il vostro nome e cognome?
[Unknown]: hostname.tv.smc
Qual è il nome della vostra unità organizzativa?
[Unknown]: Ricerca e Sviluppo
Qual è il nome della vostra organizzazione?
[Unknown]: SMC Group
Qual è il nome della vostra città o località?
[Unknown]: Villorba
Qual è il nome del vostro Stato o Provincia?
[Unknown]: Treviso
Qual è il codice a due lettere del paese per questa unità?
[Unknown]: IT
Il dato CN = hostname.tv.smc, OU = Ricerca e Sviluppo, O = SMC
Group, L = Villorba, ST = Treviso, C = IT è corretto?
 [no]: si
```

Esporto il certificate (nel file .crt)

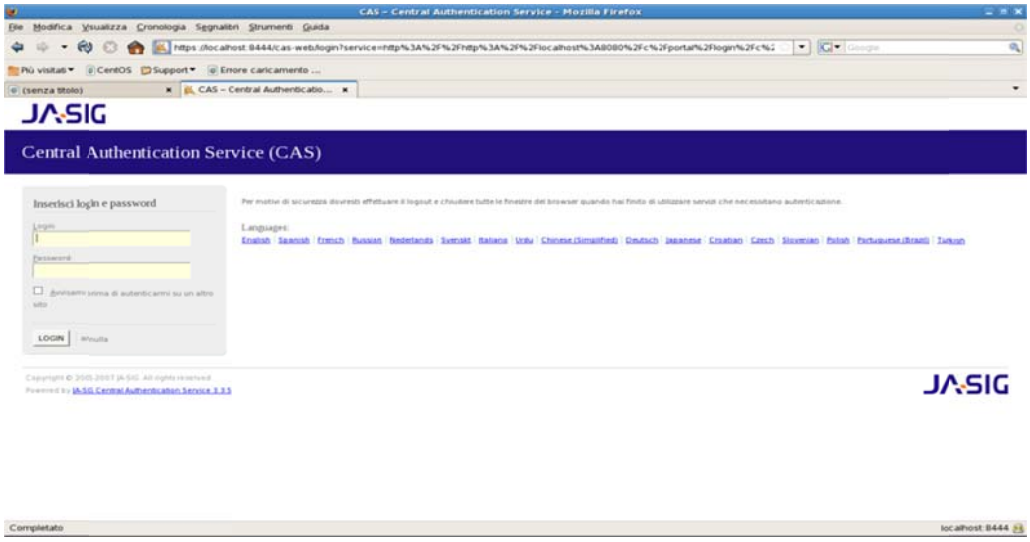
```
keytool –export –alias tomcat –keypass changeit –file server.cert
      Immettere la password del keystore : changeit
Il certificato è memorizzato nel file <server.crt>
```

Infine importo il certificato all'interno del portafoglio usato da java

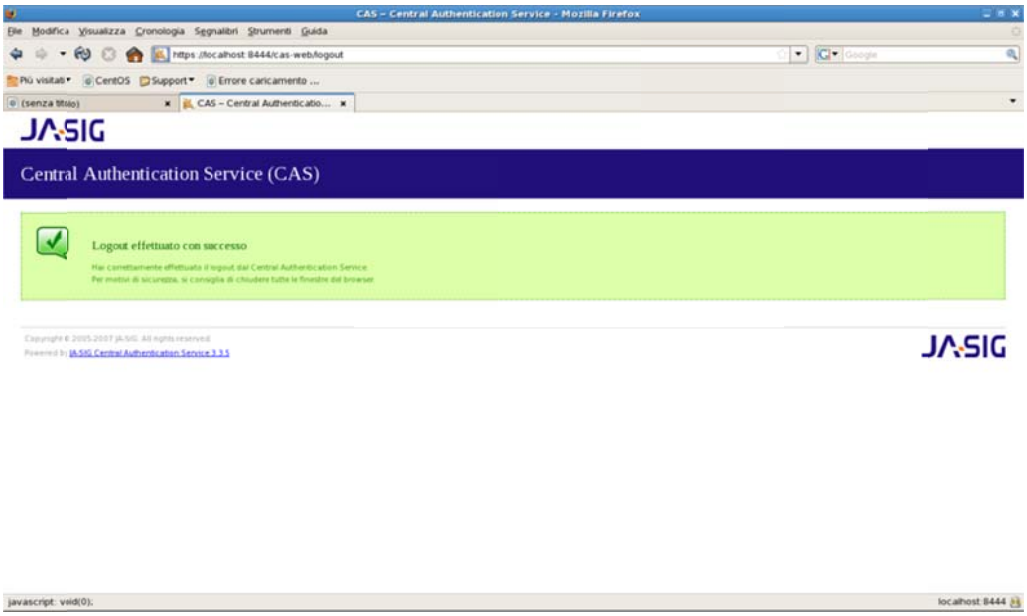
```
keytool -import -file server.crt -keypass changeit -keystore
/w3/java/apps/jdk1.6.0_16/jre/lib/security/cacerts

Immettere la password del keystore:  changeit
Proprietario: CN=cassrv01, OU=Ricerca e Sviluppo, O=SMC Group,
L=Treviso, ST=Treviso, C=IT
AutoritÃ  emittente: CN=cassrv01, OU=Ricerca e Sviluppo, O=SMC
Group, L=Treviso, ST=Treviso, C=IT
Numero di serie: 4b7ed729
Valido da: Fri Feb 19 19:23:37 CET 2010 a: Thu May 20 20:23:37 CEST
2010
Impronte digitali certificato:
         MD5:  CD:68:1D:8E:94:EC:19:58:D5:2C:0F:F1:8E:B5:3F:5A
```

```
         SHA1:
D8:8B:27:4D:66:AC:E2:5D:F8:ED:75:C4:16:92:D7:3E:E3:1F:D2:75
         Nome algoritmo firma: SHA1withDSA
         Versione: 3
Considerare attendibile questo certificato? [no]:  si
Il certificato è stato aggiunto al keystore
```

Trovo quindi il *".keystore"* nella homedir dell'utente jasigcas con i dati del certificato, e lo stesso (la sua parte pubblica) importata all'interno del jdk. In questo momento è possibile avviare il server CAS. Avviare quindi il Tomcat ed effettuare l'accesso al CAS con https://hostname.tv.smc:8444/cas-web/login (NOTE: La porta 8444 deve essere ugale a quella modificata nel file server.xml). Dovrebbe apparire quindi la schermata di loin CAS e nessun errore nei log Catalina.



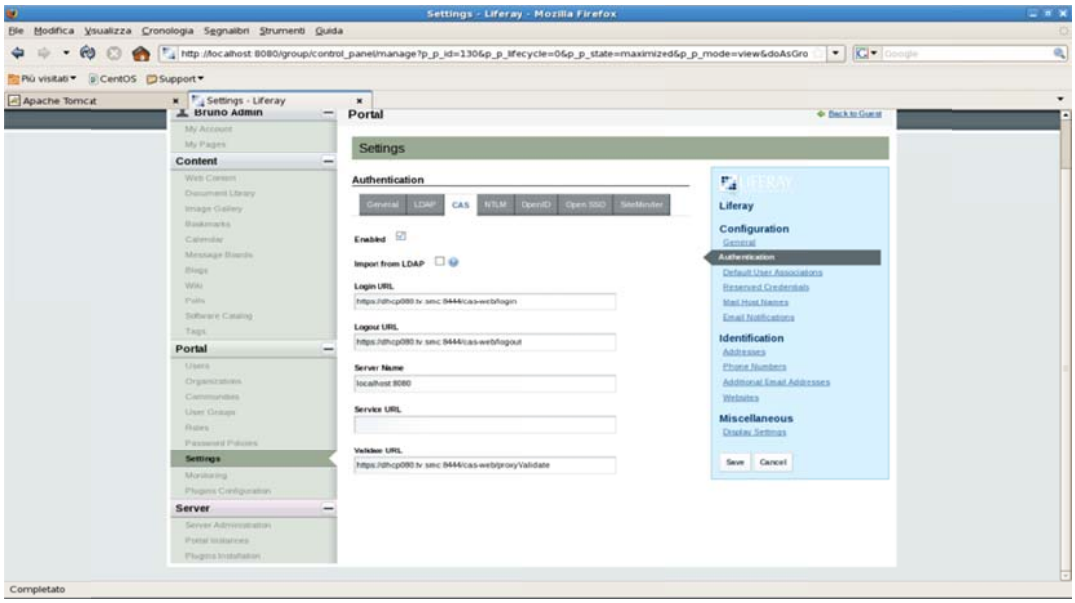
Dovrebbe apparire quindi la schermata di logOut nel momento in cui si esce dalla sessione.



4.3.2 Parametri Liferay

È necessario quindi impostare i seguenti valori richiesti da Cas in Liferay *ControlPanel/Settings/Authentication/Cas* :

- cas.login.url=[https://hostname.tv.smc:8444/cas-web/login]
- cas.logout.url=[https://hostname.tv.smc:8444/cas-web/logout]
- cas.server.name=[localhost:8080]
- cas.service.url=[null]
- cas.validate.url=[https://hostname.tv.smc:8444/cas-web/proxyValidate]



4.3.3 Avvio e Conclusioni

Dopo aver avviato il server tomcat che gestisce il Cas, avvio il bundle di Liferay che a sua volta avvia il portale, dopo di che dalla homepage è necessario cliccare il link di accesso. Se è stato configurato correttamente si dovrebbe essere reindirizzati alla schermata di login del server CAS. Apparirà un avviso in cui avvisa che il certificato di *hostname.tv.smc* non è valido in quanto auto-firmato. Eseguo la procedura per ignorare l'avviso ed acquisire comunque il certificato, vengo redirezionato su *"https://hostname.tv.smc:8444/cas/loging"* ovvero la pagina di login del cas server. Inserisco "bruno" nei campi login e password (l'installazione demo accetta qualsiasi login basta che sia uguale nei due campi) e dalla pressione del pulsante "login" vengo rimandato alla pagina di Liferay già autenticato come “bruno”, quindi senza inserire nuovamente le credenziali sul portale

Se il test ha funzionato, si dispone già di un server CAS installato e integrato con Liferay, a questo punto il portale non è più responsabile per l'autenticazione degli utenti, confida che il server CAS autentichi gli utenti in modo corretto. Il server CAS ha strategie configurabili per l'autenticazione degli utenti e finora è stato usato quello predefinito, che autentica l'utente solo se l'utente e la password sono gli stessi.

4.4 OpenSSO

4.4.1 Predisposizione del software necessario

Risulta necessario il download di alcuni pacchetti software che utilizzeremo durante la procedura di integrazione, quali:

- Apache-tomcat-6.xx
- opensso_express_20090901
- Bundle-With-Tomcat di Liferay
- Java Development Kit (JDK)

Procedo quindi con la fase iniziale di creazione degli utenti, creo l'utente *opensso* con password *"password"* che verrà utilizzato per l'esecuzione di OpenSSO, successivamente creo l'utente liferay con password *"password"* che verrà utilizzato per avviare Liferay. Installo quindi il Java Development Kit 6 (jdk.6.0_16) e imposto la variabile JAVA_HOME in *$BASE_DIR/apps/jdk1.6.0_16* e aggiungendo *$JAVA_HOME/bin* alla variabile *$PATH* terminando quindi con la modifica del file *$HOME/.bash_profile* per impostare le variabili d'ambiente.

A questo punto devo modificare il file di gestione del server tomcat adibito all'OpenSSO per abilitare il canale ssl, e per farlo modifico il file */home/opensso/apache-tomcat-6.0.24/conf/server.xml*, inoltre devo adattare in ogni tomcat il file di configurazione in quanto devo fare coesistere le varie istanze in porte diverse altrimenti entrambi i server tomcat lavorerebbero sullo stesso numero di porta generando dei conflitti. Modifico quindi le porte .: *8080 – 8443 – 8009* in *8081-8444-8010* . Successivamente avvio il server Tomcat di OpenSSO e per farlo eseguo i seguenti comandi :

```
cd /home/opensso/apache-tomcat-6.0.24/bin
./startup.sh
```

Verifico che sia andato tutto bene dal file di log

```
cd $HOME
tail -f /home/opensso/apache-tmcat-6.0.24/logs/catalina.out
```

e dal fatto che aprendo un browser e puntando http://hostname.tv.smc:8081 mi appaia la maschera base di tomcat e a questo punto spengo il server.

```
cd  /home/opensso/apache-tomcat-6.0.24/bin
./shutdown.sh
```

4.4.2 Attivazione del server OpenSSO

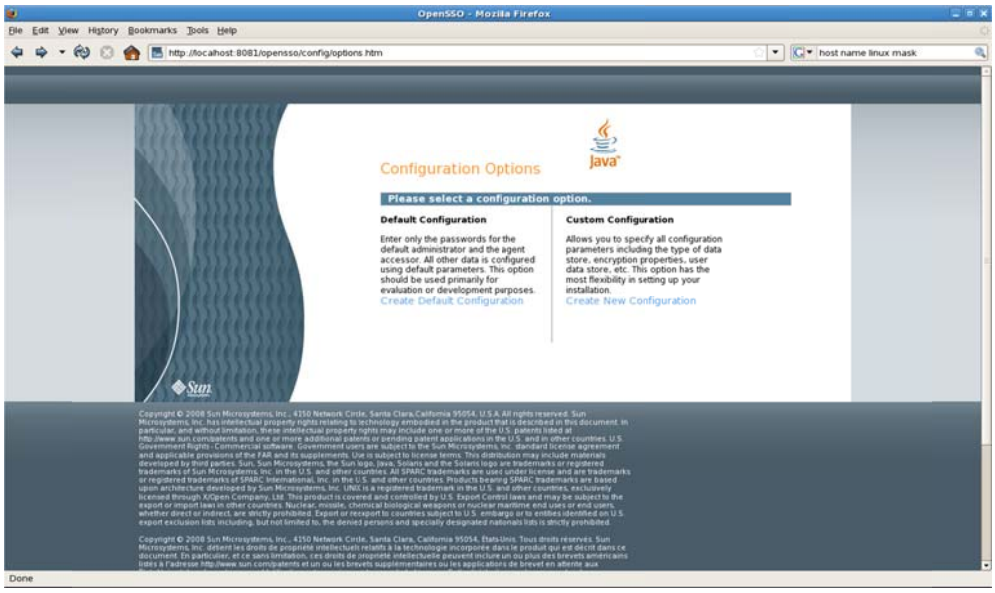
Inizieremo con la creazione di OpenSSO sul server Tomcat 6.xx ossia copio dal Zip *"opensso_express_20090901.zip"*, dalla cartella *"opensso/deployable-war"* il file *"opensso.war"* in */home/opensso/apache-tomcat-6.0.24/webapps/*

Elimino il contenuto delle cartelle logs e work (per avere uno scenario pulito) e avvio il server Tomcat. Successivamente allo spegnimento del server si può notare

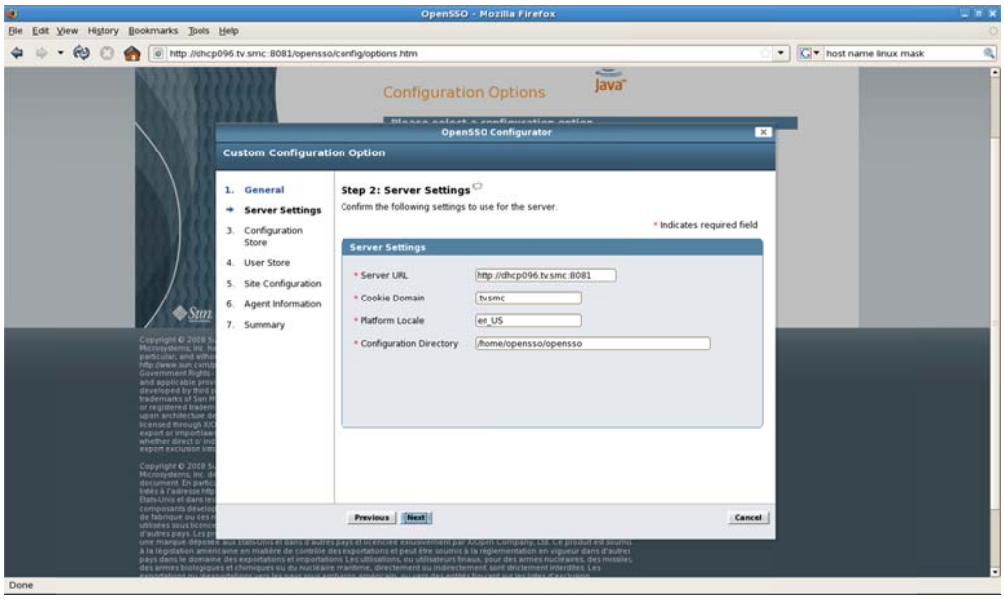
la presenza di una nuova cartella creata automaticamente dal sistema all'interno della Directory Webbapps.

4.4.3 Configurazione OpenSSO con AD e Generazione del CERT

Dopo aver avviato il server tomcat è necessario configurare OpenSSO, e per farlo apriamo un browser e andiamo all'indirizzo http://hostname.tv.smc:8081/opensso dove *8081* è la porta che abbiamo modificato nel file *server.xml* del server tomcat. E' necessario prestare quindi attenzione al fatto che se il sistema operativo di base è in una lingua diversa dall'inglese durante la configurazione del servizio di OpenSSO apparirà un errore che impedisce alla configurazione di terminare con successo in quanto la Generazione del CERT fallisce.



Si avvierà in automatico una pagina di configurazione dove è possibile configurare il tomcat in modalità standard o personalizzata e quindi scegliamo il metodo personalizzato *"Custom Configuration"*. Dopo aver cliccato su *"Create New Configuration"* apprarirà la seguente schermata in cui bisogna inserire la password per accedere come amministratore e successivamente configurare o modificare le impostazioni di OpenSSO.

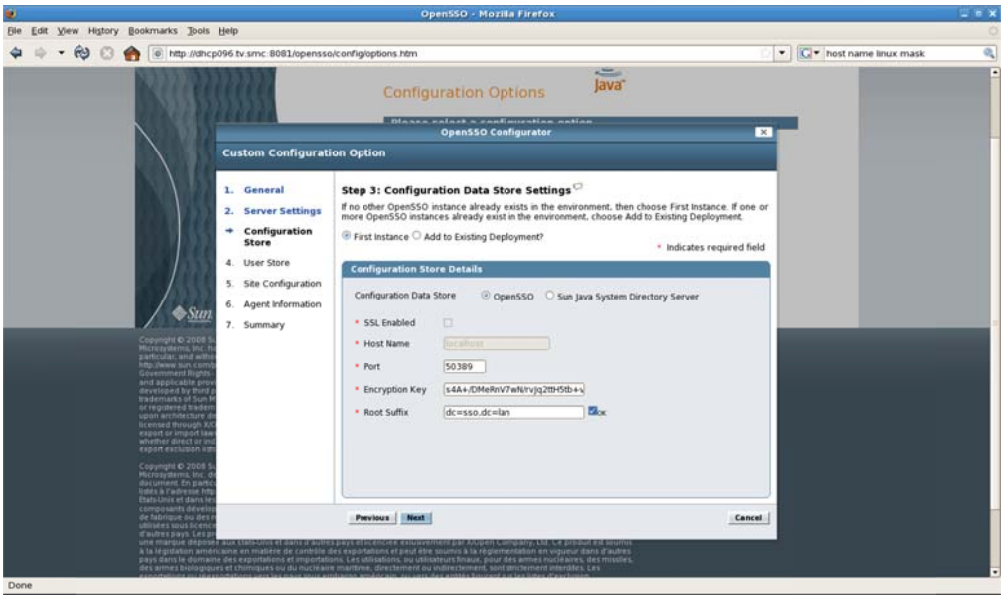


È richiesto di impostare i parametri riguardanti il server utilizzato .:

- Server URL : http://hostname.tv.smc:8081
- Cookie Domain : .tv.smc
- Platform Locale : en_US
- Configuration Directory : /home/opensso/opensso

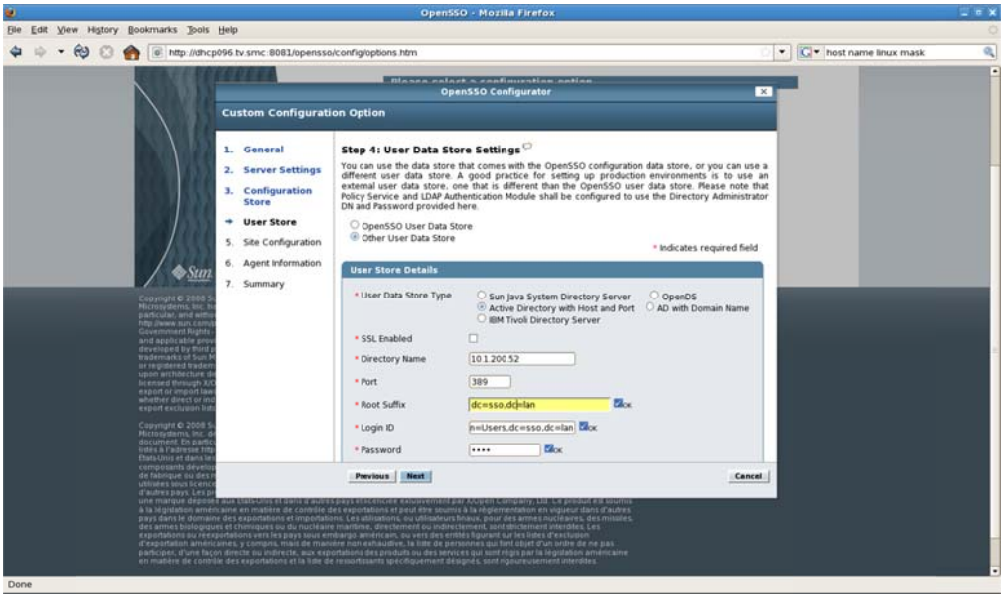
Bisogna avere a disposizione i dati di configurazione riferiti al Data Store che si intende utilizzare per l'autenticazione e nel nostro caso facciamo uso di OpenSSO associato ad Active Directory quindi impostiamo i parametri nel seguente modo . :

- Configuration Data Store : OpenSSO
- SSL Enable : Null
- Host Name : Null
- Port : Default
- Encryption Key : Default
- Root Suffix : dc=sso , dc=lan

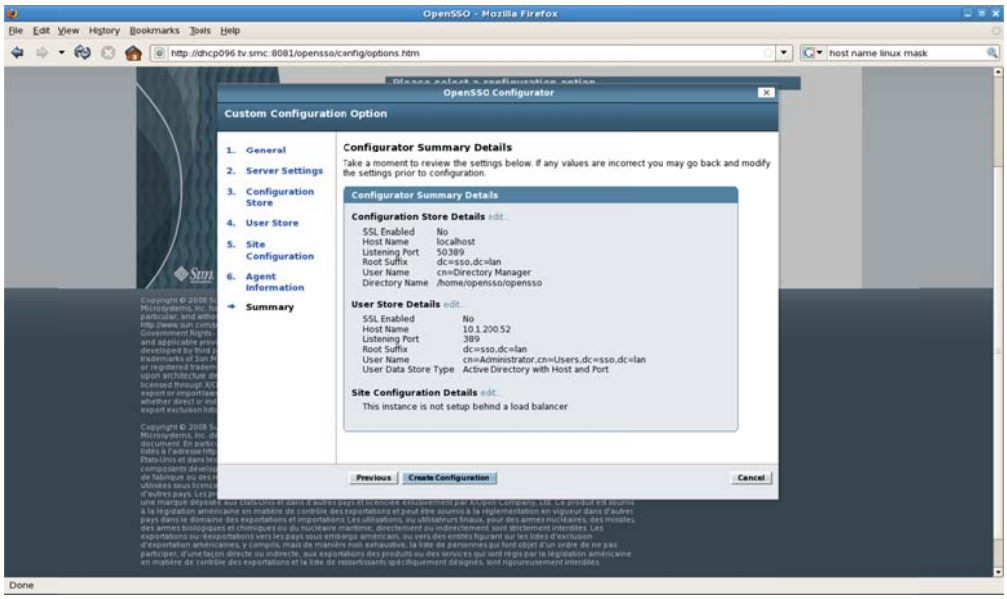


Impostazione dei parametri per l'utilizzo del Data Store .:

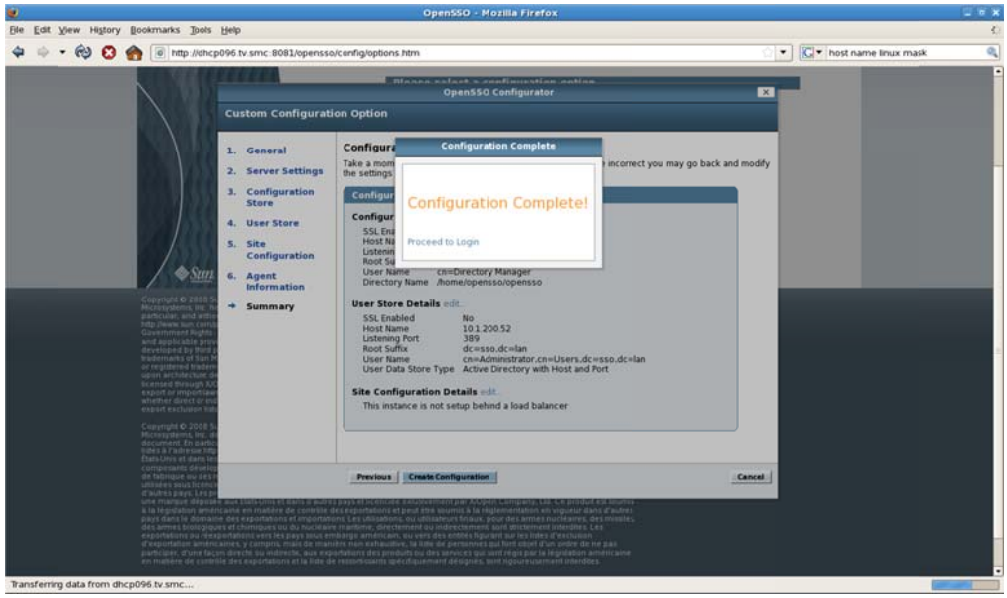
- User Data Store Type : Active Directory with Host and Port
- SSL Enable : Null
- Directory Name : 10.1.200.52
- Port : 389
- Root Suffix : dc=sso , dc=lan
- Login ID : n=Users , dc=sso, dc=lan
- Password = *********



Appare una finestra riassuntiva dei parametri inseriti per un controllo generale dopo di che cliccando su “Create Configuration” si da il via alla configurazione.



Al termine della configurazione appare una finestra che ne conferma l’effettivo successo della procedura.



Trovo quindi il *".keystore"* nella homedir dell'utente opensso con i dati del certificato, e lo stesso (la sua parte pubblica) importanta all'interno del jdk. In questo momento è possibile avviare OpenSSO.

4.4.4 Parametri Liferay

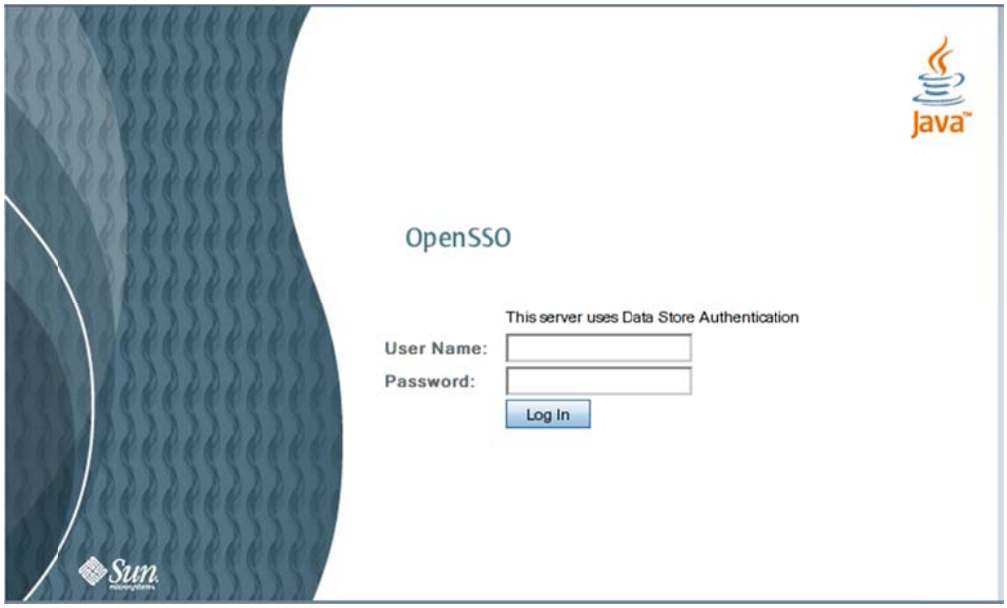
Risulta necessario impostare i seguenti valori richiesti da OpenSSO in Liferay nella pagina *ControlPanel/Settings/Authentication/OpenSSO* e per farlo spunto la casellina Enable di inizio pagina e inserisco i seguenti parametri :

- Login URL e Logout URL
  - http://hostname.tv.smc:8081/opensso/UI/Login?goto=http://hostname.tv.smc:8080/c/portal/login
  - http://hostname.tv.smc:8081/opensso/UI/Logout?goto=http://hostname.tv.smc:8080/portal/logout
- Service URL : http://hostname.tv.smc:8081/opensso
- Screen Name Attribute : sAMAccountName
- Email Address Attribute : mail
- First Name Attribute : cn
- Last Name Attribute : sn

4.4.5 Avvio OpenSSO e Test

Avviare il server tomcat che gestisce OpenSSO dopo di che avvio il bundle di Liferay che a sua volta avvia il portale, dopo di che dalla homepage bisogna cliccare il link di accesso. Se è stato configurato correttamente si dovrebbe essere reindirizzati alla schermata di login di OpenSSO che risulta essere la seguente : http://hostname.tv.smc:8081/opensso/UI/Login?goto=http://hostname.tv.smc:8080/c/portal/login ovvero la pagina di login del server OpenSSO. Prestare attenzione al fatto che l'utente che utilizzerò per eseguire il login deve essere presente anche all'interno del database del Portale ossia di Liferay. Inserisco "alorenzon" nel campo login e "password" nel campo password dopo di che premento il pulsante del "login" vengo reindirizzato alla home di Liferay, già autenticato come "alorenzon", quindi senza inserire nuovamente le credenziali nel portale.

Se il test ha funzionato, si dispone di un sistema di Single Sign On basato su Opensso e integrato con Liferay, e di conseguenza il portale non è più responsabile per l'autenticazione degli utenti.



4.5 Apache Directory Server

4.5.1 Predisposizione del software necessario

Risulta necessario il download di alcuni pacchetti software che utilizzeremo durante la procedura di integrazione, quali:

- *Java Development Kit (jdk.6.0_16)*
- *Apache Directory Server*
- *Apacheds-1.5.5-i386.rpm*
- *Aggiungicontextentry.ldif*

Procedo quindi con la fase iniziale di creazione degli utenti, creo l'utente *apacheds* con password *"password"* che verrà utilizzato per far lavorare ApacheDS dopo di che installo il Java Development Kit 6 (jdk.6.0_16) e imposto la variabile JAVA_HOME in *$BASE_DIR/apps/jdk1.6.0_16* e aggiungo *$JAVA_HOME/bin* alla variabile *$PATH* terminando quindi con la modifica del file *$HOME/.bash_profile* per impostare le variabili d'ambiente.

4.5.2 Installazione e configurazione

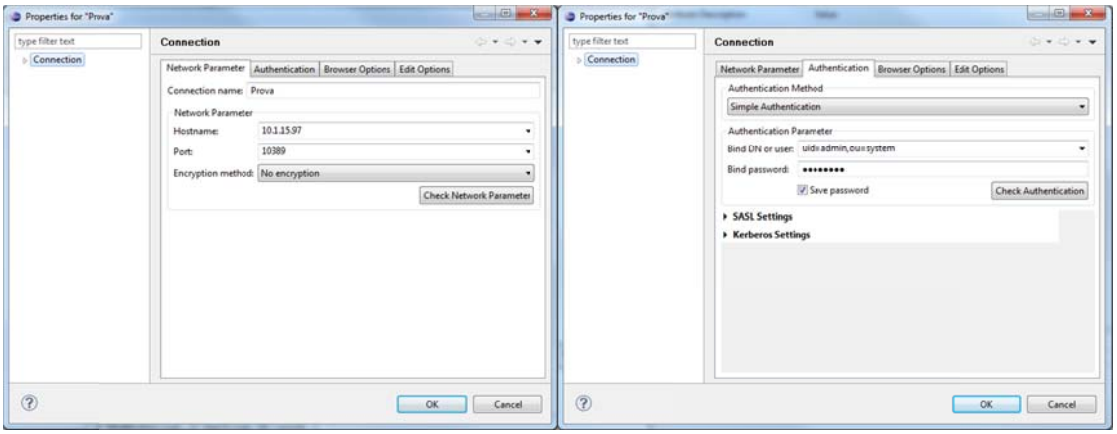
Si procede quindi con l'installazione del file Apache Directory Server precedentemento scaricato dal sito ufficiale *" Apacheds-1.5.5-i386.rpm "*, e una volta terminata la procedura verifico la presenza della cartella contenete il file *server.xml* che si trova in */var/lib/apacheds/default/conf/.* Da questo file è possibile modificare la porta predefinita su cui è in ascolto il server LDAP, ossia la numero "10389" se la trasmissione avviene in chiaro altrimenti la numero "10636" se la connessione viene cifrata "SSL".

```
<ldapserver id = "ldapserver"
            ...>
  <transports>
  <tcpTransport address= "0.0.0.0" port= "10389"
       nbThreads= "8"
       backLog= "50" enableSSL= "false" />
  <tcpTransport address= "localhost" port= "10636"
  enableSSL= "true"/>
  </ trasporti>
  ...
</ ldapserver>
```

È necessario riavviare il server per rendere effettive le modifiche, nel nostro caso però, le porte dovranno rimanere quelle di default.

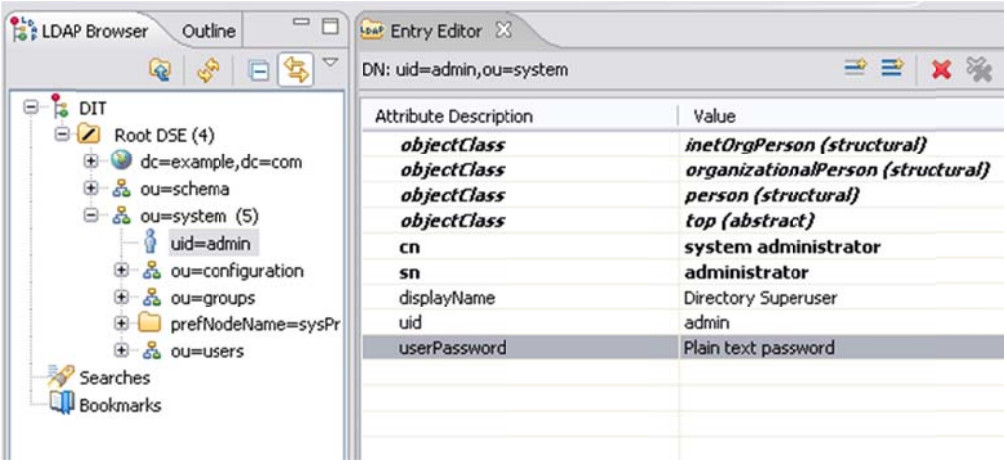
4.5.3 Avvio e Modifica dei parametri

Avvio il Server portandomi su *"/etc/init.d"* ed eseguendo come root il seguente comando *"./apacheds start default"*. Se è stato installato correttamente, tramite il comando *"./apacheds status default"* dovrebbe apparire il seguente segnale *"Apache Directory Server – default is running (#processo)"*.



Mentre il server è installato e funzionante, modifico il valore dell'attributo userPassword dell'amministratore *(uid = admin, ou = System)* tramite LDAP, e per eseguire questa procedura devo utilizzare l' Apache Directory Studio basato su Eclipse. Avvio quindi Eclipse e aggiungo LDAP Browser cliccando su *"help/Install new software.."* e inserendo nel menù *"work with"* il link riguardante *"LDAP_Browser"* http://directory.apache.org/studio/1.x. Utilizzo quindi *"LDAP Connection"* per eseguire il collegamento all'Apache Directory Server creando una nuova connessione e inserendo i seguenti dati necessari per il collegamento ed infine premo il tasto *"FINISH"* per stabilire la connessione.

- *ConnectionName* *Dato presente di default*
- *Hostname* *hostname.tv.smc*
- *Port* *10389*
- *Admin DN* *" uid=admin, ou=system "*
- *Password* *" secret "*

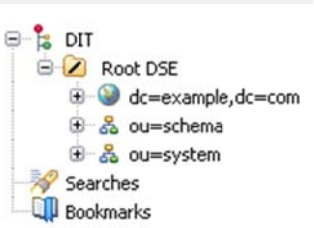


Modifico quindi il valore dell'attributo userPassword della voce uid=admin, ou=sistema, passo quindi alla voce del DIT *(LDAP Browser)* e faccio doppio click nell'Entry Editor, dove è sufficiente poi immettere la nuova password.

4.5.4 Aggiunta di una Partizione

In Apache Directory Server gli accessi sono memorizzati in partizioni ed ogni partizione contiene un albero completo di accessi denominato DIT . La configurazione di default di ApacheDS contiene la partizione contenente i dati con il suffisso *"dc=example,dc=com"*, di conseguenza aggiungo una partizione con suffisso *"o=SevenSeas"* ma ciò richiede la modifica del file *"server.xml "*. Apro quindi il file *"server.xml"* che trovo in *"/var/lib/apacheds/default/config/server.xml"*.

```
...  <partitions>
  ...
   <jdbmPartition id="example" cacheSize="100"
    suffix="dc=example,dc=com"
    ioptimizerEnabled="true" syncOnWrite="true">
    <indexedAttributes>
    ...
    </indexedAttributes>
  </jdbmPartition>
 </partitions>
...
```



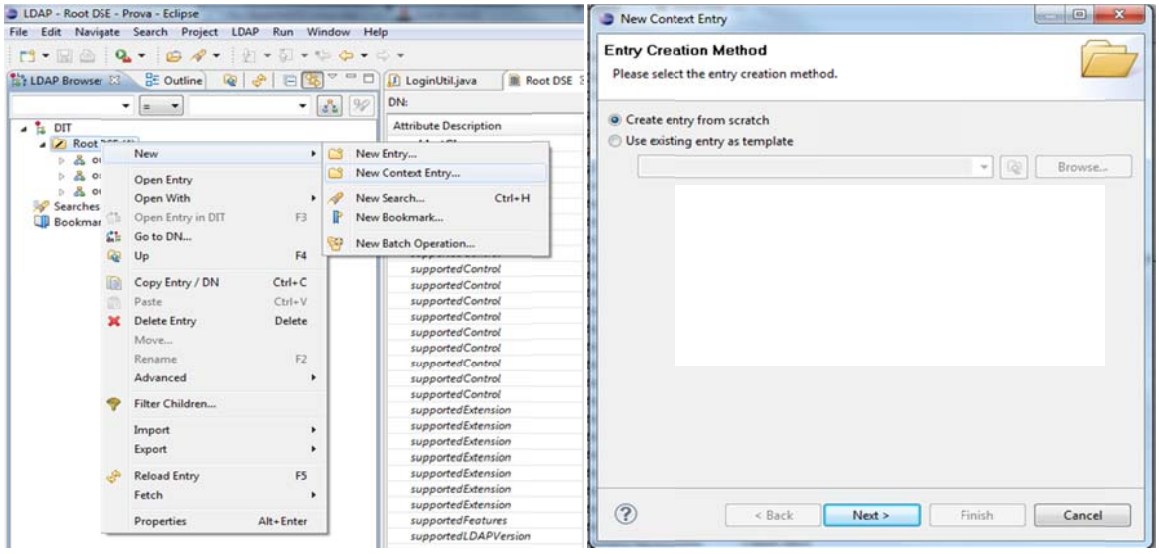
Aggiungo un altro elemento jdbmPartition per la partizione SevenSeas, appena sotto la partizione di esempio.

```
...
<partitions>
 <jdbmPartition ...>
    ...
 </jdbmPartition>
 <jdbmPartition id="sevenSeas" suffix="o=sevenSeas" />
</partitions>
...
```

Salvo il file "*server.xml*" e riavvio il server per rendere effettivi gli aggiornamenti, cosi il server possiede ora un nuovo suffisso ma ancora nessuna Entry d'accesso.

4.5.5 Creazione del Context Entry

A questo punto bisogna creare il Context Entry, da eclipse clicco su *"RootDSE"* con il tasto destro, dopo di che *"New"* ed infine *"New Context Entry"* .

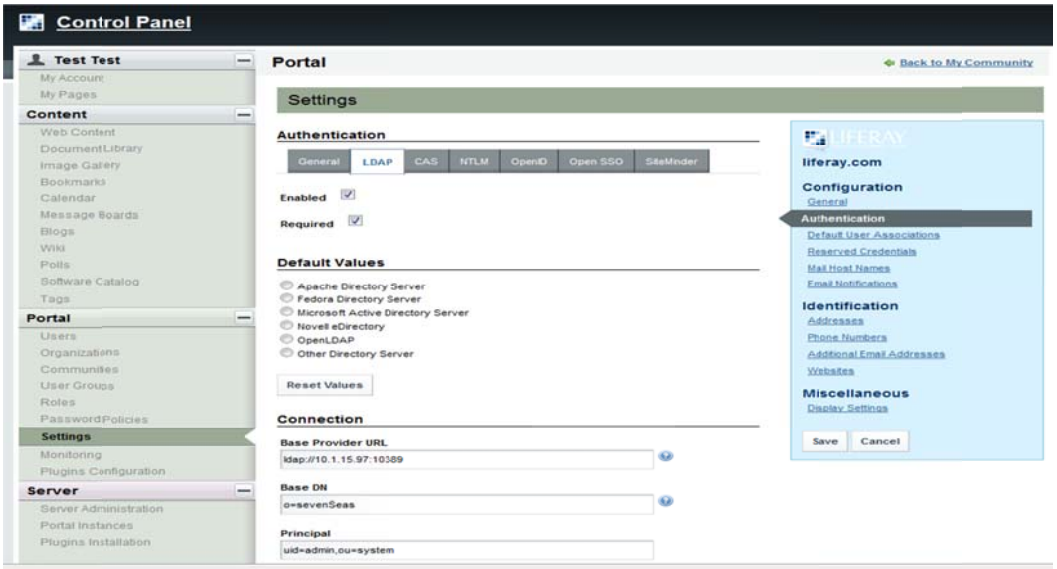


Clicco quidi su *"Create entry from scratch"* , *"organization"* , *"o=sevenSeas"* e inserisco manualmente i valori, oppure importo un file *"aggiungicontextentry.ldif"* contenente le varie entry.

Seguendo la seconda alternativa, dopo aver scaricato il file.ldif clicco su *"import"* , *"LDIFImport"* e inserisco la traccia di dove si trova il file nel computer, in questo modo sono state inserite le entry e quidi seleziono uno a scelta, ad esempio *"Horatio Nelson"* e modifico la password in *"test"*. Questa entry sarà utilizzata per eseguire l'accesso una volta configurata la sincronizzazione con Liferay.

4.5.6 Parametri Liferay

Avvio Eclipse e di conseguenza Liferay, accedo come amministratore ed imposto dal pannello di controllo i valori richiesti per l'integrazione con LDAP *ControlPanel/Settings/Authentication/LDAP*



- Authenticate by : *By Screen Name In "Authentication/General".*
- Enable : *Spunto la voce*
- Required : *Spunto la voce*
- Connection
  - Base Pmovider URL : *ldap://hostname.tv.smc:10389*
  - Base DN : *o=sevenSeas*

- Users
  - Authentication Search Filter : *(objectClass=inetOrgPerson)*
  - Import Search Filter : *(objectClass=inetOrgPerson)*
  - User Mapping Screen Name : *uid*
  - User Mapping Password : *userPassword*
  - User Mapping Email Address *: mail*
  - User Mapping First Name : *givenName*

4.5.7 Avvio e Conclusioni

Avvio eclipse che a sua volta avvia il portale, dopo di che dalla homepage bisogna cliccare il link di accesso e se è stato configurato correttamente inserendo le credenziali contenute in LDAP, accedo alla pagina personale dell'utente inserito. Inserisco quindi "hnelson" nel campo login e "test" nel campo password e confermando le credenziali vengo reindirizzato nella home.
Se il test ha funzionato, Liferay è integrato correttamente con ApacheDirectoryServer, il portale quindi non è più responsabile per l'autenticazione degli utenti ma delega questo compito ad ApacheDS.

4.6 Active Directory

4.6.1 Scelta utilizzo Active Directory

La scelta dell'utilizzo di Active Directory in Liferay come sistema di identificazione e autenticazione degli utenti, è certamente giustificato dal fatto che in questo momento è sicuramente il più diffuso (la percentuale di utilizzatori del servizio di directory di Microsoft dovrebbe aggirarsi intorno al 80%). Tuttavia, limitarsi a supportare Active Directory porta con sé due significativi svantaggi. Nel caso in cui si decida di supportare la Integrated Windows Authentication (IWA), l'utilizzo di Liferay sarà sostanzialmente limitato a Internet Explorer, dal momento che il browser dovrà essere in grado di comunicare al web server le credenziali dell'utente autenticato in Windows. Google Chrome, ad esempio, non

supporta tale funzionalità, sebbene per Mozilla Firefox e Apple Safari siano disponibili le estensioni SPNEGO, che comunque non consentono di effettuare l'impersonificazione degli utenti.

La filosofia portante di Liferay è basata sulla flessibilità e l'adattabilità ai molteplici scenari che si possono incontrare nelle varie realtà aziendali, relativamente alle diverse soluzioni software in uso (siano esse Microsoft o di altre aziende o provenienti dal mondo open source), alle varie architetture hardware, ecc. In questo ambito si inserisce uno dei requisiti fondamentali del portale, ovvero il supporto al multi-browser. È chiaro che, limitando il supporto al solo Active Directory, browser particolari (ad esempio, in ambito mobile) e diversi sistemi operativi client e server potrebbero infatti non essere in grado di gestire un sistema di autenticazione basato su Active Directory.

Per quanto riguarda il primo punto, ovvero il supporto a IWA, nel caso in cui si decida di non implementare tale funzionalità, gli utenti dovranno reinserire le proprie credenziali al momento dell'accesso al portale e di conseguenza l'integrazione con il meccanismo di autenticazione di Windows potrà al massimo rappresentare un accessorio, un "qualcosa in più" per gli utenti aziendali in LAN o in VPN, ma non un vero e proprio servizio di Single Sign On.

La consegunte scelta di non limitarsi a supportare unicamente Active Directory, porta lo sviluppatore a valutare le varie alternative come ad esempio, se integrarsi con il directory service presente in azienda o se utilizzarne uno stabilito a priori per tutte le installazioni di Liferay. Il primo caso risponde in pieno alla filosofia di Liferay ma tuttavia, dovranno essere tenute in conto eventuali azioni di adattamento alla piattaforma software presente in azienda, nel caso in cui non vi sia un supporto "out of the box" di una particolare tecnologia. Liferay è naturalmente allineato a una serie di standard internazionali, sia "de facto" che "de iure", in special modo per quanto riguarda i protocolli di sicurezza. Prodotti conformi a tali standard potranno così essere integrati senza problemi al suo

interno. In caso contrario, è comunque possibile raggiungere la piena compatibilità fra Liferay e gli altri sistemi attraverso opportune attività di sviluppo.

Nel secondo caso, invece, l'adozione di un unico Identity Manager per tutte le installazioni di Open Square, se da un lato porta a una certa riduzione dei tempi di sviluppo dovuti all'integrazione con i sistemi preesistenti, dall'altro andrà a collidere con la filosofia "open" di Liferay, rappresentando peraltro una limitazione non indifferente, soprattutto nel caso in cui in un'azienda sia già presente un servizio di directory popolato con i dati di tutti gli utenti.

Per inciso, quanto si sta cercando di implementare non rappresenta un vero e proprio meccanismo di Single sign-on, dal momento che l'accesso al database avverrà sempre e comunque con un unico utente, e di conseguena il servizio di SSO di Liferay sarà unicamente a livello di applicazione.

4.6.2 Liferay & Active Directory in WindowServer2000

4.6.2.1 Autenticazione (con export=false)

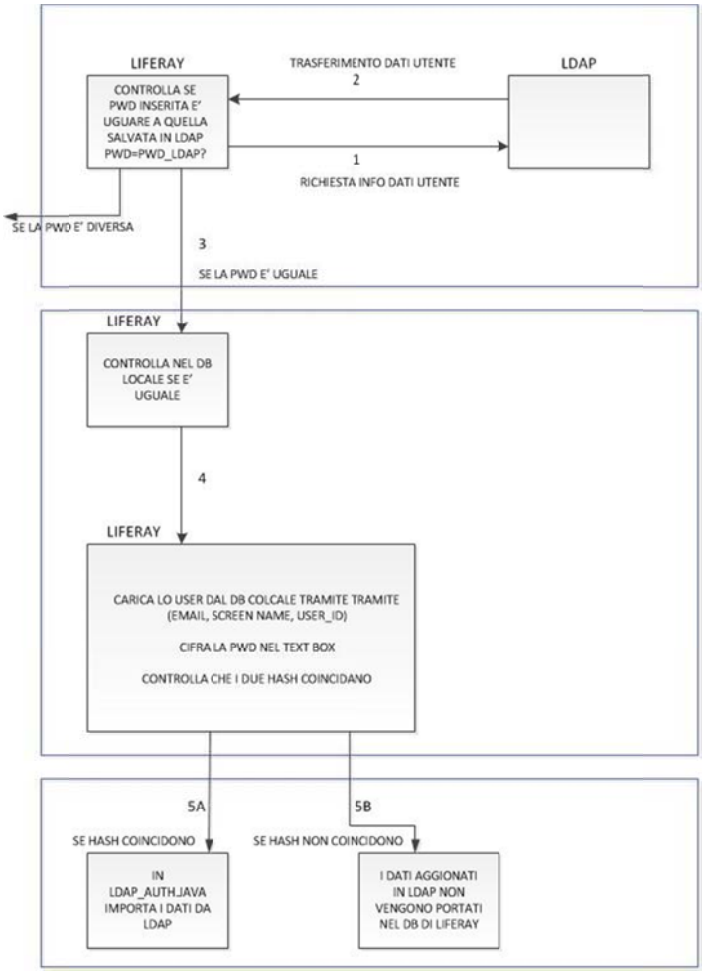
Il processo di autenticazione ha inizio dal LoginUtil.java, ossia quando un utente si autentica al portale, avvengono tre forme di autenticazione, una dopo l'altra:

La "pre-authentication pipeline" che normalmente esegue l'autenticazione in LDAP (in realtà è una lista di classi "autenticatrici" chiamate probabilmente tramite un meccanismo di IoC). Liferay chiede a LDAP di farsi dare le informazioni dell'utente, esegue la ricerca in LDAP in base a quanto specificato nelle impostazioni dopo di che controlla la pwd inserita inviandola a LDAP. Il processo di autenticazione con LDAP potrebbe aver esito positivo o negativo, nel caso ci sia un successo (ma anche in caso di insuccesso se "l'autenticazione LDAP non è required"), restituisce il valore "success".

Authentication con Database di Liferay. Se la " pre-authentication pipeline " è terminata con successo e la proprietà "auth.pipeline.enable.liferay.check" è true, controlla anche nel db locale ossia carica lo User dal db locale di Liferay tramite la chiave (e-mail, screen name o userID) dopo di che cifra la pwd nella textbox

"ispirandosi" a quella dell'utente (da questa prende il file per fare l'hash) e controlla se i 2 hash coincidono.

Post-Authentication Pipeline. Se l'autenticazione in locale è avvenuta con successo (e questo può avvenire a condizione che la "pre-authentication pipelin" abbia dato success), allora esegue la "post-authentication pipeline". L'utente è autenticato, ossia tutte e tre le autenticazioni hanno avuto esito positivo.



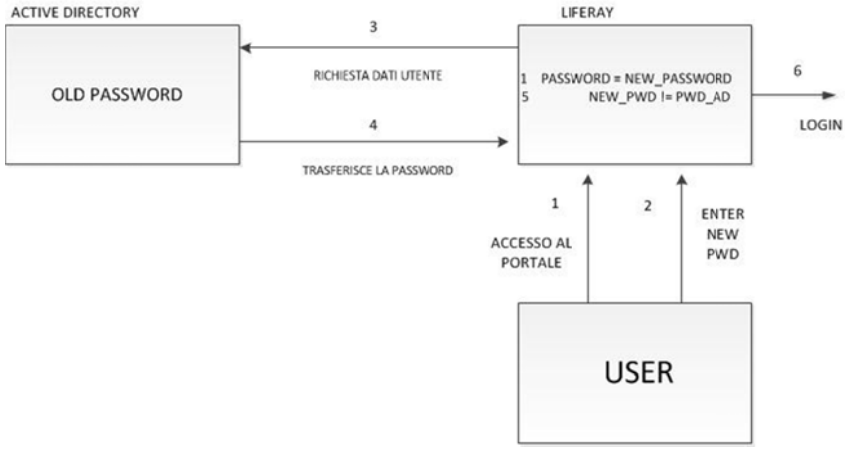
Se tutte e tre le fasi di autenticazione hanno successo allora il login viene effettuato altrimenti appare una segnale d'errore. Se l'autenticazione LDAP avviene con successo (non se restituisce "success", perché questo lo fa anche se required=false, ossia se la password inserita coincide con quella presente in LDAP), allora LDAPAuth.java:213 importa i dati da LDAP; altrimenti, se l'autenticazione fallisce (ma ripeto, può anche restituire success se required=false), allora i dati

aggiornati in LDAP non vengono portati indietro nel db di Liferay. Se viene eseguito un test di prova, cambiando il job title di “test”, pur autenticandosi con successo con la password locale, la pre-auth con LDAP è fallita e dato che non era required, aveva dato comunque successo ma il job title in Liferay non è cambiato.

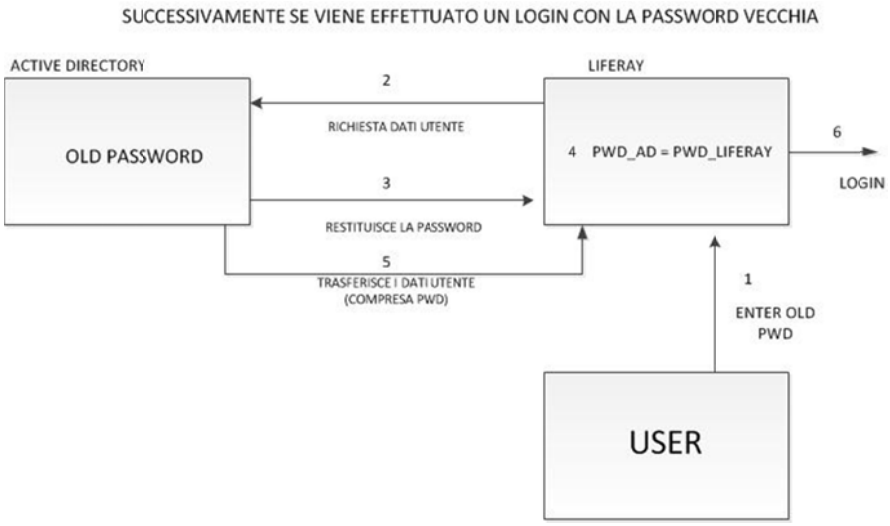
La procedura di importazione dei dati da LDAP si trova in PortalLDAPUtil.java:674 e avviene in varie fasi. Preleva i dati da LDAP (Non tutti, solo quelli per i quali c'è il campo del mapping), carica lo User dal database di Liferay e aggiorna lo User invocando prima UserLocalServiceImpl.updatePassword(…), che aggiorna il database di Liferay con la nuova password, e poi UserLocalServiceUtil.updateUser(...) passandogli i dati che ha ottenuto da LDAP, la vecchia password (che è in realtà quella specificata dall'utente durante il login) e BLANK come la password nuova, in modo che non si preoccupi di cambiare la password (visto che era stato fatto prima). Quindi, se l'utente cambia la propria password con export=false, questa non sarà esportata in AD, ma rimarrà solo nel database locale.

A questo punto la situazione cambia in base al comportamento dell’utente, ossia con quale password effettua i successivi login.

Se l’utente utilizza la password nuova (cioè quella nel db locale), la "pre-authentication pipeline" fallisce, perché le due password non coincidono e di conseguenza, i dati da Active Directory non saranno importati dentro il database locale ma il metodo dà comunque esito “success” perché required=false. L'autenticazione locale dà “success”, visto che la password inserita è quella che c'è nel database locale e l'utente può accedere con la password nuova. Nel database locale resterà la nuova password, e in Active Directory quella vecchia ma al successivo login, sia la password nuova che quella contenuta in Active Directory continueranno a funzionare.



Se l'utente utilizza invece la password vecchia (cioè quella in Active Directory, che non è stata sostituita al cambio password visto che export=false). La "pre-authentication pipeline" dà "success", visto che la password inserita è quella vecchia in Active Directory e quindi i dati da Active Directory (compresa la password) vengono importati nel database locale e in seguito all'importazione, sembra che il codice voglia riesportare indietro i dati in Active Directory (forse per aggiornare le date di modifica), ma, visto che export=false, questo non avviene e l'autenticazione locale dà success, visto che la password nel database locale è stata appena aggiornata importando i dati da Active Directory. A questo punto se al successivo login, metto la password che avevo cambiato, la "pre-" fallisce, ma restituisce comunque "success", l'importazione quindi non avviene e l'autenticazione locale fallisce (visto che nel database è stata importata la password di Active Directory) e il login viene bloccato. Se invece utilizzo la password di Active Directory, sia la "pre-authentication pipeline" che l'autenticazione locale danno esito positivo e il login viene consentito.



4.6.2.2 AUTENTICAZIONE (con export=true)

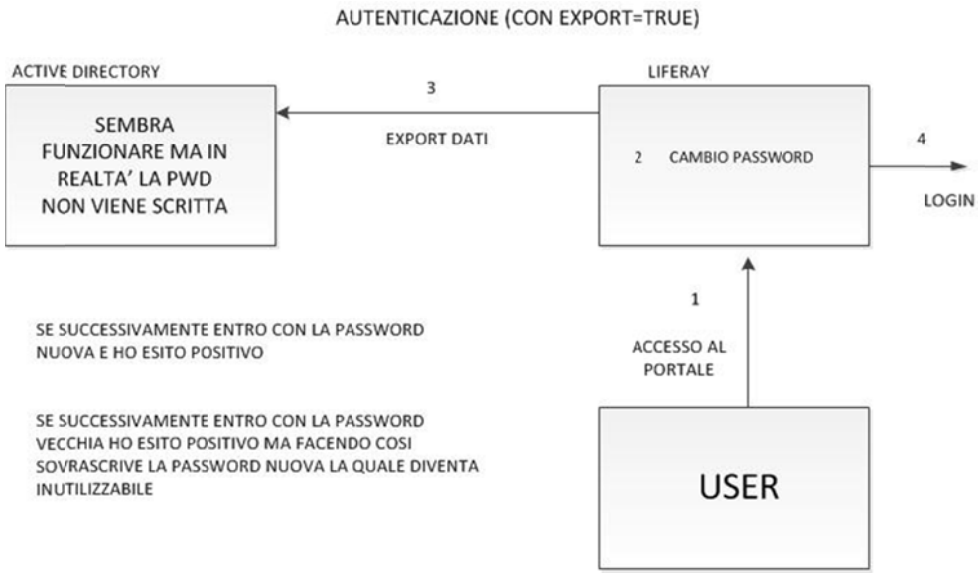
La prima fase richiede la modifica di alcuni parametri all'interno del portale Liferay, è necessario attivare export=true dalle impostazioni di Liferay (questa operazione viene eseguida dall'OmniAdmin, test@liferay.com), con parametri (UsersDN = "cn=Users,dc=sso,dc=lan" ; User Default Object Classes = "top,person,organizationalPerson,user" ; Groups DN = "cn=Users,dc=sso,dc=lan")

Dopo aver effetuato un salvataggio delle impostazioni viene avviato il CompanyLocalServiceImpl.updateDisplay, che cerca di aggiornare il language e la time zone per l'utente di default. L'aggiornamento richiede il salvataggio dei dati dell'utente di default nel db locale, il che scatena il listener registrato per l'esportazione su LDAP. Ricordiamo però che i parametri sono impostati in export=true, e dunque viene eseguito PortalLDAPUtil.exportToLDAP(User user).

Questo metodo, cerca l'utente in Active Directory e non controlla se esiste o meno e per questo motivo viene lanciata una NullPointerException (come riportato anche in http://issues.liferay.com/browse/LPS-3636).

L'eccezione comunque viene semplicemente registrata nel log, mentre il resto del codice continua a funzionare. Si procede quindi ad autenticarsi con export=true, all'inizio dei test, l'utente è maumar@sso.lan e la pwd è "test" sia in AD che nel db

e quindi la "pre-authentication pipeline" va a buon fine, (le password sono le stesse) e dà “success” e dato che la prima fase ha esito positivo Liferay cerca di importare i dati di Active Directory nel db locale nella fase di importazione dei dati dell'utente e viene importata anche la password, il che invoca Hibernate per salvare lo User nel db locale di Liferay.



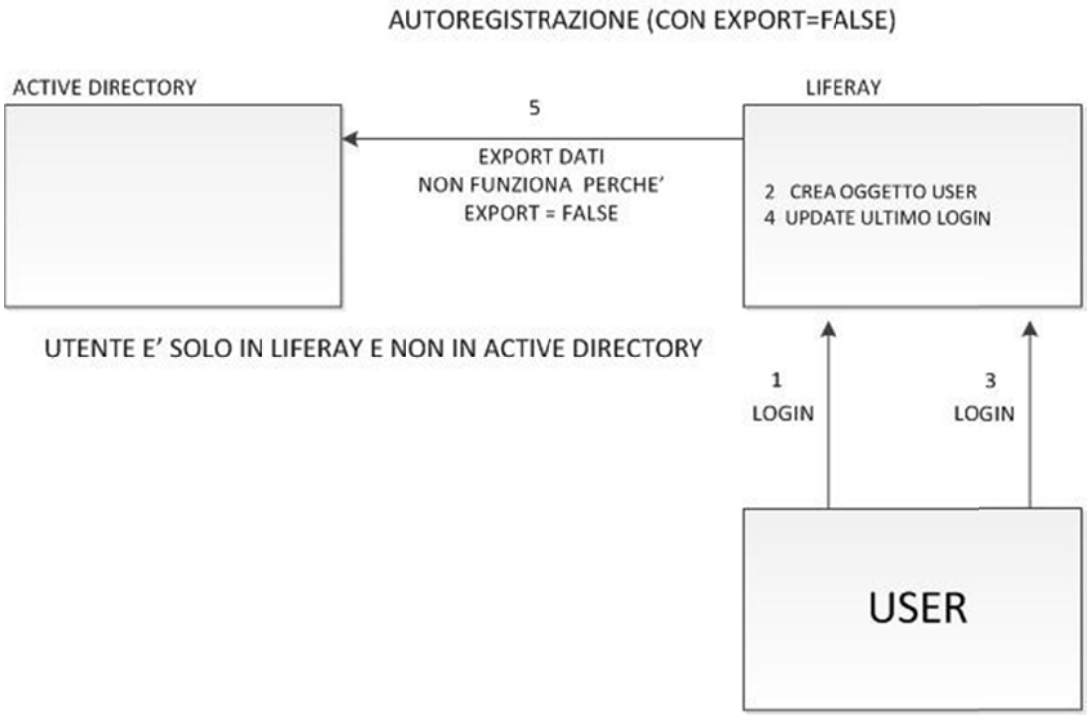
L'autenticazione è andata a buon fine e l'utente si è loggato, possiamo quindi provare a cambiare la password tramite il portale per vedere se effettivamente la password viene esportata. Nel momento in cui viene modificata la password, si ha di conseguenza l'invocazione dei metodi per salvare il nuovo User nel database locale di Liferay(cioè Hibernate). Il listener di esportazione su LDAPviene avviato, ossia viene eseguito il metodo PortalLDAPUtil.exportToLDAP(User user), questo metodo esporta i dati citati sopra in LDAP, fra cui la password, utilizzando le classi definite in javax.naming (cioè la specifica JNDI). La password dunque, dovrebbe essere cambiata sia in locale che in Active Directory, ma in realtà, questo cambiamento, non viene rilevato in quanto, se provo a loggarmi con la vecchia password, (quella originale), l'autenticazione LDAP va a buon fine, come se la password in Active Directory fosse effettivamente quella vecchia e non quella nuova. Di conseguenza, se l'utente si autentica con la nuova password, che

avrebbe dovuto essere uguale ad Active Directory, la "pre-authentication pipeline" fallisce, ma restituisce comunque esito “success”, e dato che required=false l'autenticazione locale da anch’esssa esito “success”. Si può quindi intuire che nel db locale ci sia la nuova password e sembrerebbe quindi che il codice di esportazione dei dati in Active Directory sia sbagliato, e non sia così in grado di esportare la password.

4.6.2.3 AUTO-REGISTRAZIONE (con export=false)

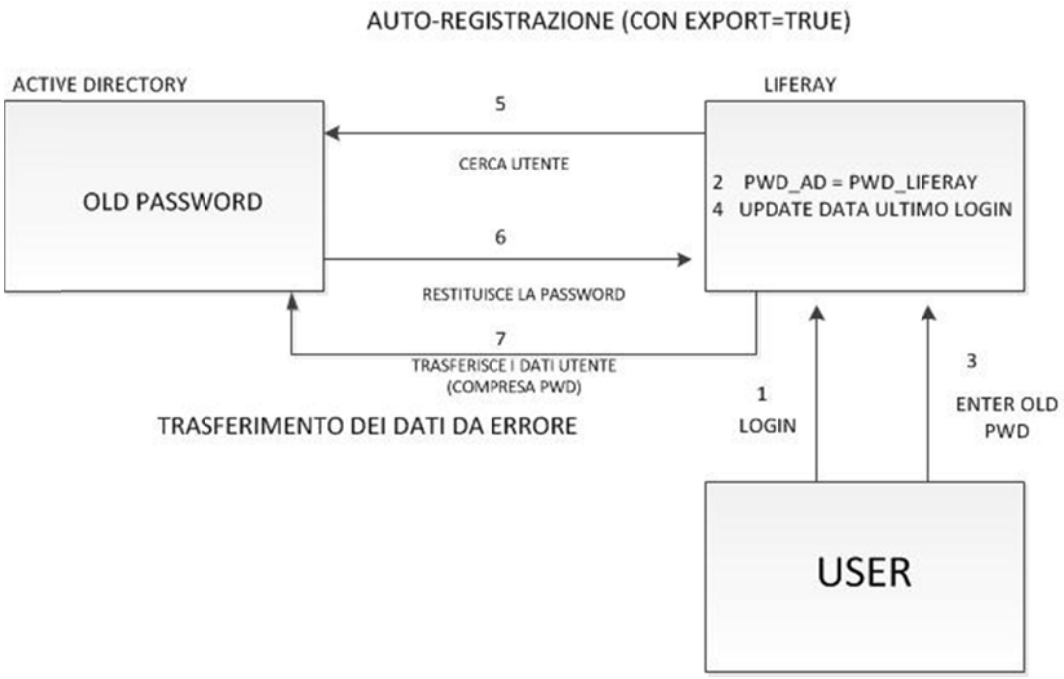
La procedura di auteregistrazione ha il via da CreateAccountAction.addUser(...), la proprietà login.create.account.allow.custom.password" definisce se il nuovo utente che si auto-registra può specificarsi una pwd e in caso contrario, gli sarà assegnata una password d'ufficio. Il punto esatto in cui si crea questa password è UserLocalServiceImpl.addUser(...) e questo metodo si preoccupa anche di creare un nuovo oggetto User e di passarlo al layer di persistenza affinché questo possa salvarlo nel database. Se l'utente, dopo essersi registrato esegue il login con la password che gli ha assegnato Liferay, il codice in MainServlet (riga 774) invoca UserLocalServiceUtil.updateLastLogin(...), se la proprietà "users.update.last.login" vale true, cerca quindi di aggiornare nel database locale la data dell'ultimo login. Questo metodo è implementato in UserLocalServiceImpl e chiama il layer di persistenza per aggiornare la data dell'ultimo login, questo chiaramente scatena il listener di esportazione su LDAP ma dato che export=false, non esegue nessuna operazione. A questo punto il nuovo utente User2 non è ancora in Active Directory e quindi dopo essersi loggato per la prima volta, l'utente riceve la pagina con i termini della licenza. Premendo “I Agree” si richiama un'altra volta il layer di persistenza e di conseguenza il listener di esportazione su Active Directory. Viene richiesta quindi la domanda segreta e anche in questo caso il layer di persistenza e il listener di esportazione su Active Directory. Con export=false, l'utente non viene quindi inserito in Active Directory, ma rimane solamente all'interno del

database locale di Liferay. Provando ad attivare export=true e ad effettuare nuovamente il login, compare un errore, NullPointerException in PortalLDAPUtil (riga 242),dato che il codice cerca l'utente in Active Directory per aggiornare la data dell'ultimo login e non lo trova.



4.6.2.4 AUTO-REGISTRAZIONE (con export=true)

Un nuovo utente, si auto-registra con export=true, quindi in UserLocalServiceImpl:addUser(...), c'è una chiamata al layer di persistenza per il Contact. All'evento After Update di un contatto è sottoscritto un ContactListener che vuole esportare il contatto in Active Directory. Per farlo usa il metodo portalLDAPUtil.exportToLDAP(Contact contact), che cerca l'utente in Active Directory, se non lo trova, crea un nuovo User e lo invia a LDAP (fa il "bind" fra l'oggetto LdapUser e il DN), ma anche questa procedura purtroppo non funziona.



4.6.3 Requisiti necessari per ottenere un'efficace integrazione

Verranno ora descritti i requisiti fondamentali che dovranno essere necessariamente soddisfatti al fine di ottenere un'efficace integrazione fra i due sistemi nell'ambito del servizio di SSO. Una volta raggiunti tali obiettivi, Open Square potrà appoggiarsi completamente e in modo affidabile al server Active Directory già presente in una realtà già avviata, (il quale contiene di conseguenza tutte le credenziali degli utenti del sistema informatico), utilizzando quest'ultimo come Identity Manager.

Il processo di integrazione fra Liferay e Active Directory risulta favorevole separarlo in due fasi distinte, per ognuna delle quali definire un set di requisiti da soddisfare in modo da velocizzare il lavoro di adattamento del codice di Liferay, separando le funzionalità di primaria importanza da quelle accessorie, implementabili dunque in un secondo momento.

Fase I: in questa fase, l'integrazione fra Liferay e Active Directory è limitata a pochi aspetti di primaria importanza, ovvero consentire l'accesso in Liferay a utenti definiti in Active Directory. Tutte le operazioni di amministrazione dell'archivio di utenti (creazione di nuovi utenti, modifica dei dati, reimpostazione password, ecc.)

devono essere obbligatoriamente effettuate mediante la console di Active Directory, ad opera dell'amministratore di sistema. In tale fase, dunque, il lavoro di sviluppo e di correzione del codice di Liferay si limita a bloccare tali operazioni di modifica nell'interfaccia grafica del portale.

Fase II: in questa fase, l'integrazione fra Liferay e Active Directory è completa. Una volta raggiunti gli obiettivi definiti, sarà possibile effettuare le operazioni di amministrazione dell'archivio utenti (fra cui la modifica della password) indifferentemente dall'interfaccia grafica di Liferay o dalla console di Active Directory. Sarà inoltre consentita l'auto-registrazione di nuovi utenti al portale, il che potrà risultare utile nel caso in cui Liferay venga utilizzato per la creazione del portale istituzionale dell'azienda.

4.6.4 Modifica codice sorgente di Liferay

4.6.4.1 Modifiche relative Prima Fase

Le modifiche relative alla prima fase sono brevemente illustrate successivamente, in base alla funzionalità che rappresentano :

La creazione di nuovi utenti dovrà avvenire unicamente tramite l'interfaccia di configurazione di Active Directory: dovrà dunque essere disabilitata la funzione di auto-registrazione, nonché la possibilità di creare nuovi utenti tramite il pannello di controllo del portale.

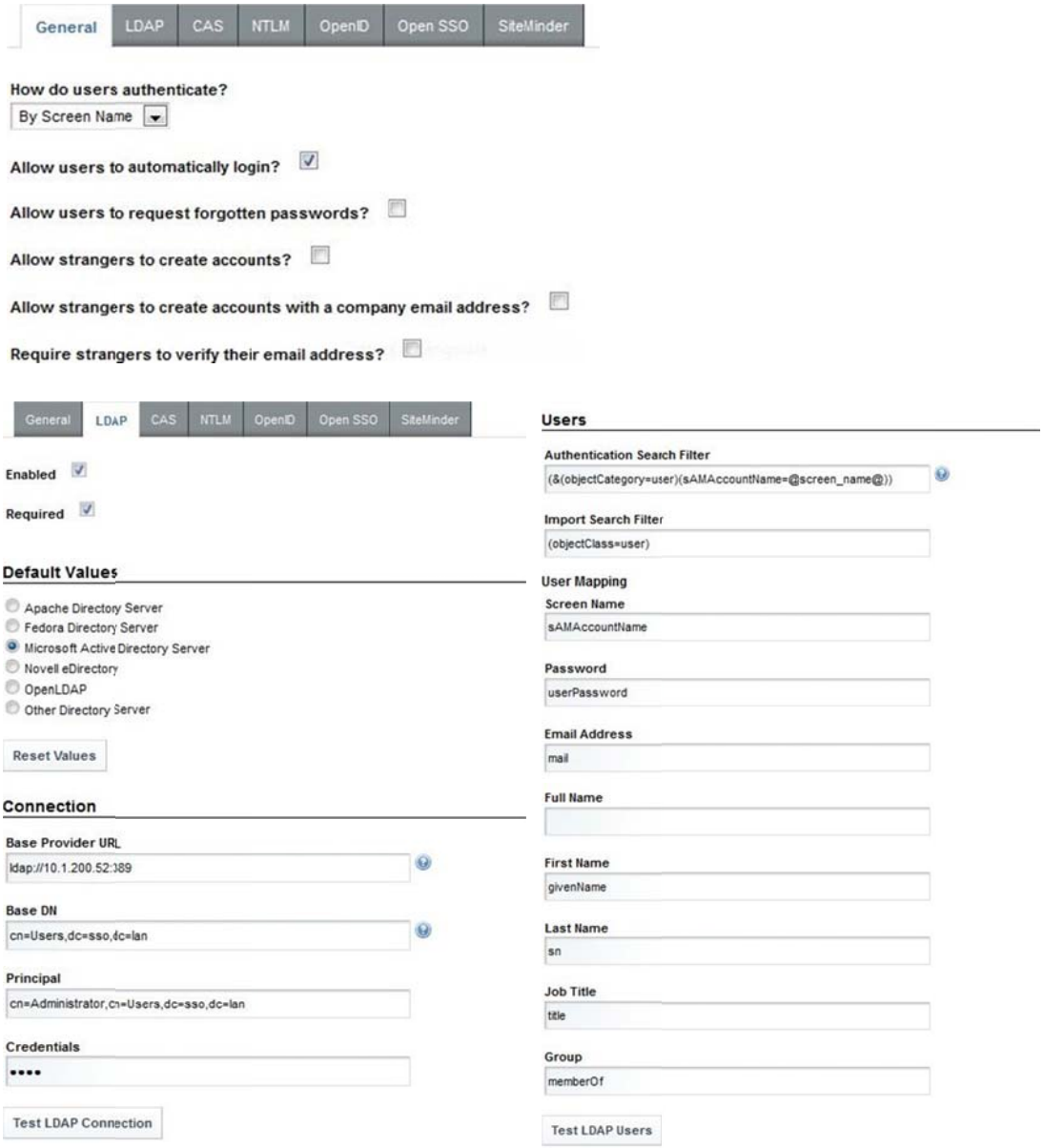
Gli utenti saranno identificati univocamente dal loro Screen Name, mappato con l'attributo SamAccountName in Active Directory.

Non è necessario prevedere la possibilità di modificare lo Screen Name di un utente, né dal portale né dalla console di Active Directory: una volta aggiunto l'utente al servizio di directory e al portale, lo Screen Name sarà un attributo immutabile.

La modifica della password di un utente potrà avvenire unicamente tramite l'interfaccia di configurazione di Active Directory; dovrà dunque esserne impedita la modifica dal portale.

La modifica degli altri campi di un utente (indirizzo e-mail, nome, cognome, nome completo, job title e appartenenza ai gruppi) sarà consentita unicamente tramite l'interfaccia di configurazione di Active Directory: in tal modo, sarà possibile mantenere l'impostazione export di Liferay a false.

La maggior parte dei requisiti previsti per la prima fase, sono già correttamente soddisfatti dalle versioni "vanilla" di Liferay. Al fine di rispondere ai requisiti di questa prima forma di integrazione, è necessario configurare opportunamente



Liferay, agendo in particolare sulle impostazioni in Control Panel → Portal Settings → Authentication. Gli screenshots che seguono illustrano una configurazione di esempio di Liferay, relativamente all'interfacciamento con un server Active Directory di Windows 2003 Server:

L'attributo `Required` in Control Panel → Portal Settings → Authentication → LDAP impone che, alla fase di login, le credenziali specificate dall'utente siano validate mediante la comunicazione al server di Active Directory; nel caso in cui questo, per qualche motivo, non sia disponibile, l'accesso al portale sarà negato. Sarà dunque necessario valutare opportunamente se mantenere o no questo vincolo. Imponendo `Required=true`, gli utenti eventualmente creati dall'amministratore del portale mediante le funzioni presenti nel pannello di controllo di Liferay e non ancora presenti in Active Directory non saranno utilizzabili, dal momento che, con `Export=false`, questi non saranno esportati in Active Directory all'atto della creazione. Infatti, durante la fase di login, Liferay valida le credenziali inserite mediante una comunicazione con il servizio di directory. I dati degli utenti inseriti mediante il pannello di controllo del portale non saranno trovati e dunque, visto che `Required=true`, l'accesso sarà negato.

Con i parametri sopra elencati è possibile fare in modo che Liferay soddisfi sostanzialmente tutti i requisiti previsti per la fase I. L'impostazione `Export=false` limita la sincronizzazione dei dati degli utenti in una sola direzione, ovvero da Active Directory a Liferay e qualunque modifica effettuata nel database locale degli utenti non sarà riportata nel servizio di directory centrale. Di conseguenza, potrebbe essere conveniente impedire agli utenti di Liferay di modificare i propri dati tramite la sezione My Account del Control Panel: infatti, essendo disabilitata l'esportazione su Active Directory, le modifiche effettuate ai dati di un utente sarebbero salvate unicamente nel database locale di Liferay.

Ad esempio, modificando lo Screen Name di utente (ovvero il campo specificato nel parametro `How do users authenticate?`), i dati nel database di Liferay e nel servizio di directory non sarebbero più sincronizzabili fra loro, dal momento che la chiave di ricerca avrebbe valori diversi nei due archivi. In particolare, è consigliabile evitare del tutto la modifica dello Screen Name di un utente, sia tramite la funzione My Account che mediante la console di Active Directory.

Per fare ciò, può essere ad esempio possibile eliminare la voce di menu nel Control Panel, cosicché gli utenti non possano accedere alla sezione My Account; oppure modificare il portlet in modo che non possa più essere in grado di salvare i dati; oppure, infine, disattivare completamente il portlet mediante una modifica al file liferay-portlet.xml. Chiaramente, l'efficacia delle varie soluzioni deve essere valutata attentamente (ad esempio, eliminare semplicemente la voce di menu non può impedire con assoluta sicurezza che un utente possa accedere al portlet My Account). Presumibilmente, inoltre, più efficace sarà la soluzione che si deciderà di implementare, più tempo sarà necessario per apportare le necessarie modifiche al codice.

Tuttavia, il portlet My Account consente la modifica di una moltitudine di campi, di cui soltanto un ristretto sottoinsieme è esportabile in Active Directory. Bloccare l'intero portlet impedirebbe l'utilizzo e la modifica di tutti i campi di un utente, il che potrebbe non essere ammissibile. In tal caso, sarà necessario modificare il portlet My Account in modo da rendere in sola lettura solamente i campi in comune fra il database locale di Liferay e il servizio di directory.

Per quanto riguarda invece il requisito "Impedire la creazione di nuovi utenti mediante l'interfaccia del portale", l'impostazione di `Required=true` dovrebbe essere sufficiente allo scopo. Tuttavia, pur se, così facendo, gli utenti creati all'interno del portale non saranno mai validati nella fase di login, una soluzione più elegante potrebbe consistere nel disabilitare completamente gli elementi dell'interfaccia grafica deputati all'inserimento di nuovi utenti.

Cap. 5

Conclusoni & Sviluppi Futuri

5.1 Conclusioni e Sviluppi

In questo periodo di tirocinio sono state analizzate diverse tecnologie utili per la creazione di un servizio di Single Sign On e create alcune basi di integrazione tra alcune di esse. Un successivo sviluppo di questo progetto, potrebbe essere la scelta di alcuni dei software precedentemente descritti e la realizzazione di una soluzione unica e completa in base alle esigenze di un'azienda specifica.

Durante lo studio di queste tecnologie è risultato molto difficile trovare un punto in cui questi software risultino completamente interoperabili tra loro e di conseguenza, lo sviluppo di un progetto completo, funzionante e stabile, richiede molti sforzi sia a livello di personale impiegato che di costi economici.

Bibliografia

Apache Software Foundation. 1999-2010. http://tomcat.apache.org/download-60.cgi.

Comunity, Spring Source. s.d. http://www.springsource.org/documentation.

Foundation, The Apache Software. 2010. http://directory.apache.org/apacheds/1.5/.

Hibernate. 2009. http://www.hibernate.org/docs.html.

Home, Oracle Wikis. 2010. http://wikis.sun.com/display/OpenSSO/getstarted.

IBM. s.d. https://www.ibm.com/developerworks/web/library/wa-singlesign/#1.

Intesys. 2009. http://www.intesys.it/Liferay-Portal-Server/?ti=522.

Jasig. 2009. http://www.jasig.org/cas/community.

Java.net, OpenSSO. s.d. https://opensso.dev.java.net/.

Liferay 2010. 2010. http://www.liferay.com/community/forums.

Liferay Inc. 2010. http://www.liferay.com/community/wiki.

Novell. 2010. http://www.novell.com/products/securelogin/.

Richard L. Sezov, Jr. *Liferay Administrator's Guide.* 2009.

TechNet, Microsoft. s.d. http://technet.microsoft.com/it-it/library/cc678193.aspx.

Terracotta. 2010. http://www.terracotta.org/documentation/?src=/index.html.

Wikipedia, The Free Encyclopedia. 2010. http://it.wikipedia.org/wiki/Enterprise_2.0.

—. 2010. http://it.wikipedia.org/wiki/Web_2.0.

—. 2010. http://en.wikipedia.org/wiki/SAML_2.0.

—. 20 03 2009. http://it.wikipedia.org/wiki/Single_sign-on.

—. s.d. http://it.wikipedia.org/wiki/Portlet.

Yuan, Jonas X. *Liferay Portal 5.2 Systems Development.* Birmingham-Mumbai: Packt Publishing Ltd., 2009.

Ringraziamenti

Giunto al termine della laurea triennale, ritengo importante rivolgere un sentito ringraziamento a tutte le persone che mi hanno aiutato e sostenuto, in vari modi, per raggiungere questo mio importante obiettivo.

Le persone che meritano la mia più grande riconoscenza sono i miei genitori che in questi anni di studio, oltre ad avermi aiutato dal punto di vista economico sono stati un valido sostegno nei momenti in cui il mio unico pensiero era "mollare tutto" e quindi un grazie di cuore per tutto quello che avete fatto per me.

Non posso inoltre non menzionare il mio relatore, Professore Sergio Congiu che mi ha seguito nei mesi di tirocinio e nella stesura della tesi, i miei più sinceri ringraziamenti quindi per la sua disponibilità, cortesia e interesse che ha dimostrato per il mio lavoro.

Ringrazio inoltre tutti i professori della facoltà di ingegneria dell'Università di Padova e i tutors della sede stacccata di Feltre che negli anni di studio mi hanno aiutato nella preparazione agli esami e sostenuto nei momenti in cui il percorso universitario si è rivelato una strada piuttosto difficile e in salita.

Un pensiero speciale a Francesca e alla sua famiglia che proprio come una seconda famiglia si sono sempre interessati alla mia carriera universitaria, mi hanno veramente aiutato nei momenti più difficili e condiviso con me la gioia dei successi.

Un ringraziamento speciale a Michele che mi ha dedicato parte del suo tempo libero per aiutarmi sia con il tirocinio che con l'università, e ad Andrea per avermi seguito e aiutato durante il tirocinio svolto presso l'azienda SMC di Lancenigo di Villorba, la quale mi ha permesso di svolgere lo stage e questa tesi .

Grazie quindi a tutti i “colleghi della teledidattica di Feltre”, abbiamo condiviso ogni giorno momenti impegnativi ma anche di svago e spensieratezza, oltre alla strada per arrivare all’università sempre molto lunga e piena di sorprese .

Infine un ringraziamento speciale al mio amico Dario che mi ha sempre aiutato in questi anni di studio, posso quasi dire che ti sei laureato due volte, una da studente della sede di Padova e una da studente “privilegiato” in teledidattica a Feltre, che è realmente quella a cui presti maggiore attenzione.

Alberto Lorenzon

29 Marzo 2011