Versione

2.5

CMDBuild®

» Technical Manual

Novembre 2017

Author Tecnoteca srl

www.tecnoteca.com

ITA

CMDBuild ® leverages many great technologies from the open source community:
PostgreSQL, Apache, Tomcat, Eclipse, Ext JS, JasperReports, IReport, Enhydra Shark, TWE, OCS Inventory, Liferay, Alfresco, GeoServer, OpenLayers, Prefuse, Quartz, BiMserver.
We are thankful for the great contributions that led to the creation of that products.

CMDBuild ® è un prodotto di Tecnoteca S.r.l. che ne ha curato la progettazione e realizzazione, è maintainer dell'applicazione e ne ha registrato il logo.

Al progetto ha anche partecipato come committente iniziale il Comune di Udine – Servizio Sistemi Informativi e Telematici.

Il sito ufficiale di CMDBuild è http://www.cmdbuild.org

Sommario

Introduzione

CMDBuild è una applicazione Open Source finalizzata a supportare la gestione della configurazione degli oggetti e dei servizi informatici in carico al Dipartimento ICT di una organizzazione e a guidarne i processi di controllo, eventualmente secondo le “best practice” ITIL.

Gestire un Database della Configurazione (CMDB) significa mantenere aggiornata e disponibile per gli altri processi la base dati relativa agli elementi informatici utilizzati, alle loro relazioni ed alle loro modifiche nel tempo.

Con CMDBuild l'amministratore del sistema può costruire ed estendere autonomamente il proprio CMDB (da cui il nome del progetto), modellandolo su misura della propria organizzazione tramite un apposito Modulo di Amministrazione che consente di aggiungere progressivamente nuove classi di oggetti, nuovi attributi e nuove tipologie di relazioni. E' anche possibile definire filtri, “viste” e permessi di accesso ristretti a righe e colonne di ciascuna classe.

CMDBuild è in grado di fornire un completo supporto all'adozione delle “best practice” ITIL, ormai affermatesi come "standard de facto", non proprietario, per la gestione dei servizi informatici secondo criteri orientati ai processi.

Tramite un apposito sistema di gestione dei workflow è possibile definire in modo visuale, con un editor esterno, nuovi processi operanti sulle classi modellate nel database, importarli in CMDBuild ed eseguirli secondo i flussi previsti e con gli automatismi configurati.

E' disponibile un task manager integrato nell'interfaccia utente del Modulo di Amministrazione che consente di gestire in background diverse tipologie di operazioni (avvio di processi, ricezione e invio di mail, esecuzione di connettori) e di controlli sui dati del CMDB (eventi sincroni e asincroni) a fronte delle quali eseguire notifiche, avviare workflow ed eseguire script.

CMDBuild consente la stampa di report tramite il motore open source JasperReports, sia di tipo tabulare prodotti tramite un wizard interno, che di maggiore complessità ottenibili importando template disegnati tramite un apposito editor visuale esterno.

Possono essere poi definite delle dashboard costituite da grafici che mostrino in modo immediato la situazione di alcuni indicatori dello stato corrente del sitema (KPI).

Grazie all'integrazione con il diffuso sistema documentale open source Alfresco è inoltre possibile allegare documenti, immagini, video ed altre tipologie di file alle schede archiviate in CMDBuild.

E' anche possibile utilizzare funzionalità GIS per il georiferimento degli asset e la loro visualizzazione su una mappa geografica (servizi mappe esterni) e / o sulla planimetria di un ufficio (server locale GeoServer) e funzionalità BIM per la visualizzazione di modelli 3D in formato IFC.

Sono poi inclusi nel sistema un webservice SOAP ed un webservice REST, utili per implementare soluzioni di interoperabilità con architettura SOA.

CMDBuild comprende di base due framework denominati Basic Connector e Advanced Connector, che tramite il webservice SOAP sono in grado di sincronizzare le informazioni registrate nel CMDB con fonti dati esterne, ad esempio con sistemi di automatic inventory (quali lo strumento open source OCS Inventory) o con sistemi di virtualizzazione o di monitoraggio.

Un ulteriore strumento, il CMDBuild GUI Framework, consente invece tramite il webservice REST di pubblicare su portali esterni pagine web personalizzate in grado di interagire con il CMDB.

E' infine disponibile una interfaccia utente ottimizzata per strumenti “mobile” (smartphone e tablet), implementata come “app” multipiattaforma (iOS, Android) e anch'essa collegata a CMDBuild tramite il webservice REST.

I moduli di CMDBuild

Il sistema CMDBuild comprende due moduli principali:

- il modulo di Amministrazione, dedicato alla definizione iniziale ed alle successive modifiche del modello dati e delle configurazioni di base (classi e tipologie di relazioni, utenti e permessi, upload report e workflow, opzioni e parametri)
- il modulo di Gestione dati, dedicato alla consultazione ed aggiornamento delle schede e delle relazioni nel sistema, alla gestione di documenti allegati, all'avanzamento dei processi, alla visualizzazione di dashboard e produzione di report

Il modulo di Amministrazione è riservato agli utenti abilitati al ruolo di amministratore, il modulo di Gestione è utilizzato dagli operatori addetti alla consultazione ed aggiornamento dei dati.

Documentazione disponibile

Il presente manuale è dedicato a tecnici informatici interessati ad installare il sistema e a conoscere le modalità tecniche di implementazione di alcune sue componenti.

Sono disponibili sul sito di CMDBuild (http://www.cmdbuild.org) ulteriori specifici manuali dedicati a:

- overview concettuale del sistema ("Overview Manual")
- utilizzatore del sistema ("User Manual")
- amministrazione del sistema ("Administrator Manual")
- configurazione del workflow ("Workflow Manual")
- utilizzo del webservice per l'interoperabilità con sistemi esterni ("Webservice Manual")
- utilizzo di connettori per la sincronizzazione di dati con sistemi esterni (“ConnectorsManual”)

Composizione del sistema

L'installazione di CMDBuild richiede l'utilizzo di uno o più server su cui suddividere le componenti logiche costitutive del sistema:

- server web
- componenti di elaborazione
- database
- webservice
- archivio documentale

Vengono descritti di seguito i requisiti software richiesti da CMDBuild e le modalità di installazione e configurazione.

Nella progettazione dell'infrastruttura sistemistica va considerato che l'attivazione di applicazioni web come quella in oggetto richiede la disponibilità di componenti hardware e di rete dotate di adeguati livelli di sicurezza, sia rispetto accessi esterni indesiderati (firewall, DMZ) che rispetto le esigenze di disponibilità continuativa del sistema e di adeguate prestazioni di accesso.

Requisiti hardware

Per l'installazione di CMDBuild è richiesto un server fisico o virtuale, avente le seguenti caratteristiche:

- CPU di recente generazione
- memoria RAM minimo 4 GB, meglio 8 GB per istanze dedicate ad un numero elevato di utilizzatori
- spazio disco minimo 100 GB, molto superiore se si prevede di gestire un esteso archivio documentale (alimentato tramite la funzione di gestione allegati)

Sono altresì consigliati:

- la presenza di un sistema di dischi in configurazione RAID
- un sistema di backup giornaliero dei dati
- un sistema di continuità elettrica per preservare il server da interruzioni anomale dell'alimentazione

Requisiti software

L'installazione di CMDBuild richiede la presenza dei componenti software di seguito elencati.

1) Sistema operativo

Qualunque sistema operativo supporti gli applicativi sotto elencati (consigliato il sistema operativo Linux sul quale CMDBuild è soggetto a test molto più estesi).

2) Database

PostgreSQL 9.0 o superiore (consigliato PostgreSQL 9.3).

Accertarsi che sia attivato il supporto al linguaggio "plpgsql" e che il database sia impostato con codifica UTF-8.

CMDBuild utilizza per la connessione al database la libreria “tomcat-dbcp”, che viene distribuita con Tomcat ma che non è inclusa in alcune distribuzioni Linux. In tal caso può essere recuperata nella distribuzione ufficiale di Tomcat o nella cartella extras/tomcat-libs/{Tomcat version dir} dello zip di CMDBuild e va posizionata in /usr/share/{Tomcat version dir}/lib.

CMDBuild supporta solamente il database PostgreSQL essendo l'unico che implementa le funzionalità di "derivazione" di tabelle in senso "object oriented", utilizzata sia per la gestione delle sottoclassi che per la gestione della storicizzazione delle schede dati.

Nel caso si utilizzino le estensioni GIS di CMDBuild è necessario installare anche l'estensione spaziale PostGIS nella versione 1.5.2 o 2.0. Nel caso si crei un nuovo database è necessario eseguire queste operazioni (per esempio utilizzando il tool “psql”):

```
$ psql ... cmdbuild
cmdbuild=# CREATE SCHEMA gis;
cmdbuild=# SET SEARCH_PATH TO gis, public;
cmdbuild=# \i ${POSTGIS_DIR}/postgis.sql
cmdbuild=# \i ${POSTGIS_DIR}/spatial_ref_sys.sql
cmdbuild=# \i ${POSTGIS_DIR}/legacy.sql (nel caso si utilizzi PostGIS 2.0)
cmdbuild=# ALTER DATABASE your_database_name_here SET
search_path="$user", public, gis;
```

Nel caso invece si voglia ripristinare un database esistente (per esempio utilizzando il tool “psql”):

```
$ psql ... cmdbuild
cmdbuild=# CREATE SCHEMA gis;
cmdbuild=# SET SEARCH_PATH TO gis, public;
cmdbuild=# \i ${POSTGIS_DIR}/postgis.sql
cmdbuild=# \i ${POSTGIS_DIR}/legacy.sql (nel caso si utilizzi PostGIS 2.0)
cmdbuild=# ALTER DATABASE ${DB_NAME} SET search_path="$user", public, gis;
cmdbuild=# DROP TABLE gis.geometry_columns;
cmdbuild=# DROP TABLE gis.spatial_ref_sys;
```

Sito di riferimento: http://www.postgresql.org/

3) Servlet Container / Web Server

CMDBuild richiede l'installazione di Apache Tomcat 7.0.32 o superiori (consigliato Tomcat 8.0). Attualmente, per problemi derivati dall'utilizzo dell'applicazione Shark, va verificato che Tomcat sia eseguito come utente “root” e non come utente non privilegiato (es. “tomcat”).

Al fine di supportare caratteri UTF-8 nell'utilizzo degli allegati (si veda Installazione del sistema DMS Alfresco), editare il file di configurazione “server.xml” e specificare per l'elemento “Connector” principale l'attributo URIEncoding="UTF-8".

Può essere utilizzato il web server Apache 2.2 per accedere a più istanze di CMDBuild tramite virtual host che gestiscano domini distinti.

Sito di riferimento per entrambi: http://www.apache.org/

4) Sistema DMS

Può essere interfacciato tramite protocollo CMIS (Alfresco nelle ultime versioni o altri sistemi DMS

che supportino quel protocollo) o tramite protocollo FTP e webservice proprietario (Alfresco 3.4).

Nel primo caso, per poter utilizzare la funzione di gestione di documenti allegati alle schede di CMDBuild è richiesta l'installazione del sistema di gestione documentale che supporti il protocollo CMIS, preferibilmente versione 1.1. Nel caso di Alfresco, per quanto riguarda CMIS, sono supportate le versioni 4.2 o superiori.

Nel secondo caso è possibile utilizzare utilizzare la gestione documentale offerta da Alfresco 3.4.

Sito di riferimento: http://www.alfresco.com/

5) Librerie Java

Le librerie Java sono necessarie per il funzionamento di Apache Tomcat.

CMDBuild richiede JDK 1.8 Oracle.

Sito di riferimento: http://www.oracle.com/

6) Librerie incluse nel rilascio

Il file di CMDBuild scaricabile dal sito del progetto contiene alcune librerie già all'interno del pacchetto di installazione, ed in particolare:

- la libreria per il collegamento JDBC al database PostgreSQL
- le librerie JasperReports versione 5.1.0 per la produzione di report (http://www.jasperforge.org/)
- le librerie TWS Together Workflow Server 4.4 che implementano il motore di workflow utilizzato da CMDBuild (http://www.together.at/prod/workflow/tws)
- il webservice reso disponibile dal sistema DMS Alfresco per l'utilizzo del relativo repository (http://www.alfresco.com/)
- le librerie ExtJS versione 4.1 per la generazione dell'interfaccia utente Ajax (http://extjs.com/)
- i componenti server e client per la pubblicazione di cartografia georiferita (http://geoserver.org/ e http://openlayers.org/)

Per l'eventuale disegno di report custom è utilizzabile l'editor visuale JasperStudio (versione ??) o IReport (versione 5.1.0), che producono il relativo descrittore in formato compatibile con il motore JaspertReports (http://jasperforge.org/projects/ireport).

Per l'eventuale disegno di workflow personalizzati è suggerito l'editor visuale TWE Together Workflow Editor 4.4 (http://www.together.at/prod/workflow/twe). L'editor produce come output un file XPDL 2.0 compatibile con il motore Together Workflow Server 4.4.

Per l'integrazione di funzionalità di automatic inventory si suggerisce l'utilizzo di OCS Inventory versione 1.3.3 (http://www.ocsinventory-ng.org/).

Alcune funzionalità di CMDBuild possono essere integrate sotto forma di portlet all'interno di sistemi Portal JSR compatibili, fra cui Liferay versione 6.0.6 o successive (http://www.liferay.com/).

Tutti i software sopra elencati sono rilasciati con licenza Open Source (ad eccezione eventualmente del Sistema Operativo se si optasse per una soluzione diversa da Linux).

Requisiti client

CMDBuild è una applicazione completamente funzionante in ambiente web, sia per le funzionalità di aggiornamento e consultazione delle schede dati, che per quelle di amministrazione e

strutturazione del database.

L'utilizzatore del sistema deve disporre sul proprio elaboratore esclusivamente di un browser web di recente generazione (Mozilla Firefox 3.6 o superiori fino alla versione 45 inclusa, Chrome 9 o superiori fino alla versione 49 inclusa, Microsoft Explorer 8 o superiori fino alla versione 10 inclusa).

La completa utilizzabilità web del sistema consente di supportare eventuali organizzazioni IT operanti in più sedi, consentendo l'accesso ai dati anche ad eventuali strutture esterne cui dovessero essere state affidati servizi in outsourcing (in particolare nell'ambito della partecipazione a workflow collaborativi).

Installazione di CMDBuild da interfaccia grafica

Per partire

L'installazione di CMDBuild richiede che siano già stati installati i prodotti di base necessari al suo funzionamento, e cioè:

- il database PostgreSQL (deve essere avviato e raggiungibile)
- l'application server Tomcat (non deve essere avviato)
- il DMS Alfresco (o altri che supportino il protocollo CMIS, se si intende utilizzare la funzione di gestione documenti allegati)
- l'ambiente Java

Come prima operazione è quindi necessario provvedere a scaricare ed installare tali prodotti, recuperandoli dai link indicati al capitolo precedente.

Come unica avvertenza bisognerà fare attenzione ad utilizzare directory che non contengano spazi all'interno del percorso complessivo.

Si dovrà poi procedere ad avviare il servizio PostgreSQL ed eventualmente (non è obbligatorio) il servizio DMS.

Installazione di CMDBuild

Avendo provveduto ad eseguire le operazioni indicate al punto precedente, l'installazione e la configurazione standard di CMDBuild sono molto semplici.

Per installare CMDBuild è infatti sufficiente:

- scaricare dal sito del progetto (http://www.cmdbuild.org/download) il file compresso (ZIP) corrispondente all'ultima versione rilasciata
- copiare la directory cmdbuild-{versione}.war (si trova nella "root" del file ZIP) nella directory "webapps" di Tomcat, rinominandolo in cmdbuild.war
- copiare la directory cmdbuild-shark (si trova nel file ZIP presente nella directory "extras" del file ZIP) nella directory "webapps" di Tomcat
- copiare le librerie addizionali per la versione di Tomcat scelta (si trovano nella directory "extras/tomcat-libs") nella directory "lib" di Tomcat

Una volta effettuate queste operazioni, e solamente a questo punto, si dovrà procedere ad avviare anche l'application server Tomcat.

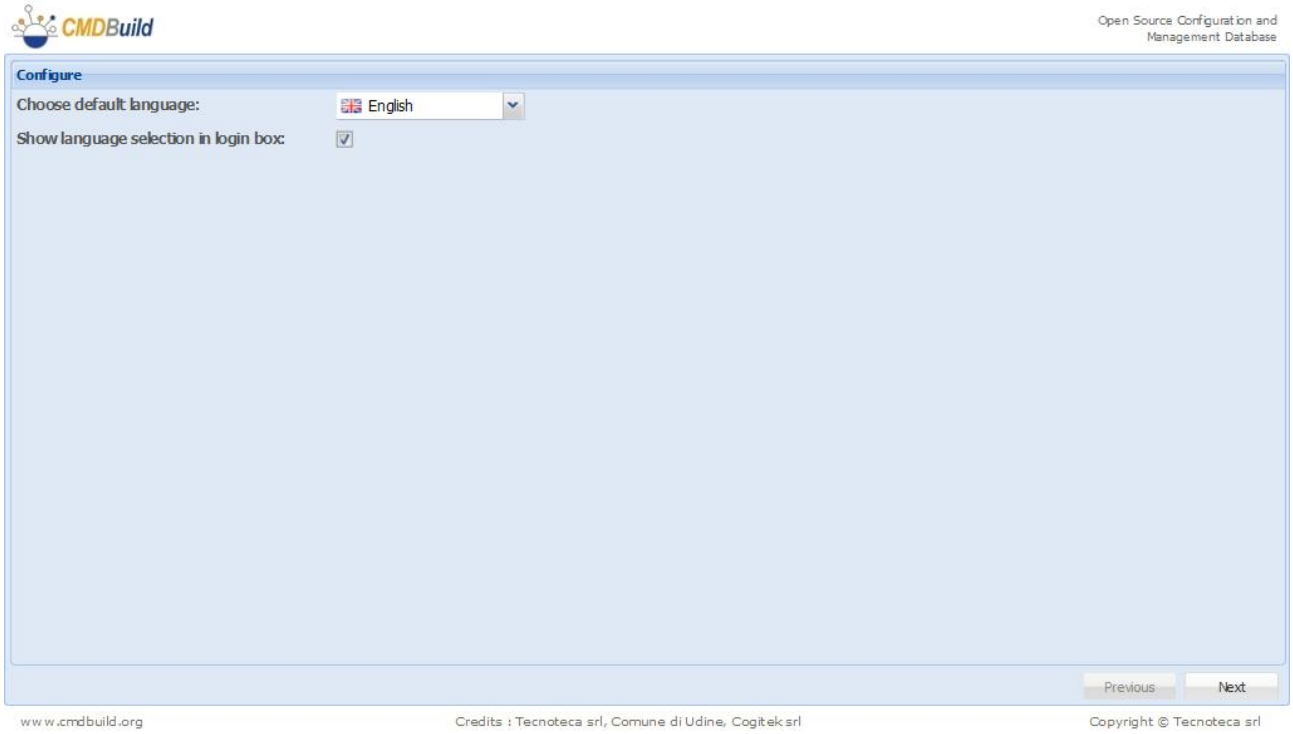
Configurazione di CMDBuild

La configurazione base di CMDBuild viene effettuata tramite alcune pagine di setup presentate automaticamente da CMDBuild al primo collegamento.

Per accedere alle pagine di setup è sufficiente collegarsi tramite il proprio browser all'indirizzo http://localhost:8080/cmdbuild (l'indirizzo potrebbe variare in base alla configurazione di Tomcat).

1) Configurazione della lingua

Se le operazioni descritte ai punti precedenti sono stati effettuati correttamente comparirà il seguente screenshot:



Da questa finestra è possibile impostare la lingua del sistema.

Spuntando la voce "Mostra la scelta della lingua nella finestra di login" sarà presentato nella interfaccia di login di CMDBuild un menù di selezione della lingua.

Una volta effettuata la scelta, cliccare su Avanti.

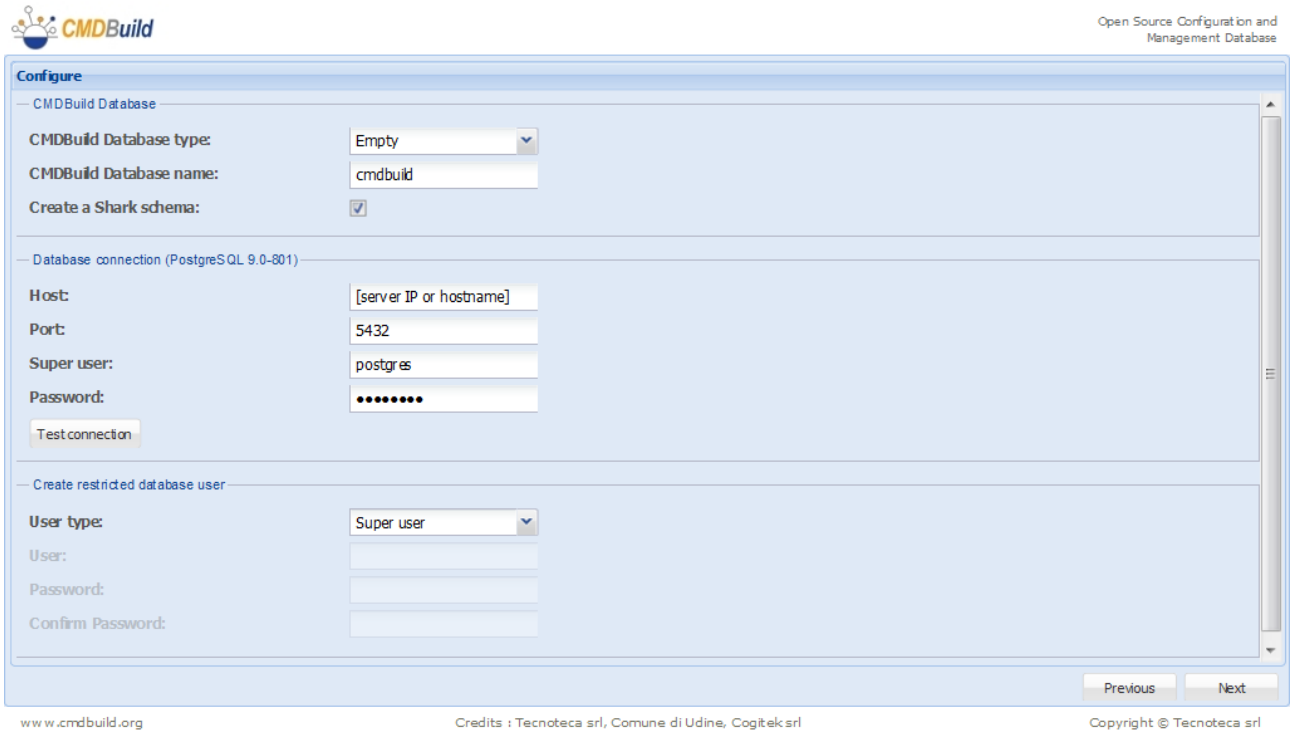
2) Configurazione del database

Nella sezione "Database connection" si deve indicare:

- l'host (nome host o indirizzo IP) su cui risiede il database PostgreSQL
- la porta su cui è raggiungibile il database PostgreSQL (la porta di default è la 5432)
- lo username con cui accedere alla base dati PostgreSQL (per attività DBA)
- la password con cui accedere alla base dati PostgreSQL (per attività DBA)

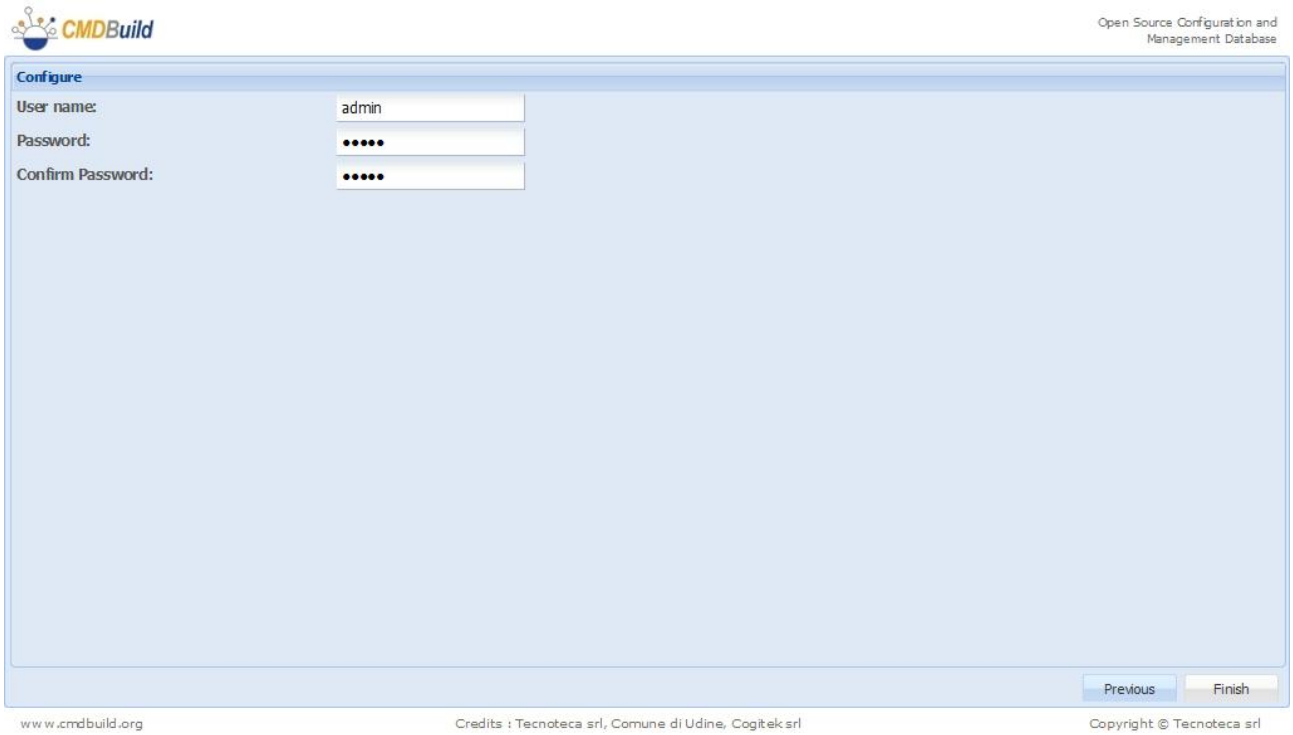
Nella sezione Database CMDBuild viene richiesto di indicare:

- il tipo di database CMDBuild da configurare, scegliendo fra:
  - creazione di un database vuoto
  - selezione di un database preesistente compatibile con CMDBuild 1.0
  - creazione di un database con dati di test
- il nome da assegnare al database



Spuntando la successiva voce "Crea utente database con privilegi limitati", viene creato in PostgreSQL un nuovo utente con tutti i privilegi di DBA, ma solo sul database che sarà utilizzato/creato nell'istanza di CMDBuild in corso di configurazione.

3) Configurazione delle credenziali di accesso



Tramite la precedente form è possibile specificare le credenziali con cui l'utente amministratore (superuser) accederà a CMDBuild (sia al modulo di Gestione che a quello di Amministrazione). Cliccando su "Fine" si verrà redirezionati all'interfaccia di login del sistema.

Installazione di CMDBuild in modalità manuale

Per partire

Prima di iniziare l'installazione di CMDBuild è necessario decomprimere la directory cmdbuild.war.

Per farlo si dovrà copiarla nella directory webapps di Tomcat ed attendere che l'application server crei la directory cmdbuild.

Verificare che in Tomcat sia presente il driver JDBC per PostgreSQL[1].

Nota: nel seguito indicheremo con:

- **{CMDBUILD}**, la directory cmdbuild in webapps
- **{ALFRESCO}**, la directory alfresco in webapps
- **{SHARK}**, la directory di Together Workflow Server in webapps

Attenzione: la seguente installazione prevede la creazione di un nuovo database vuoto. Se avete a disposizione un database preesistente passate alla sezione "Configurazione dell'applicazione".

Installazione del database

Per procedere all'installazione manuale è necessario eseguire le seguenti operazioni:

- tramite un tool con interfaccia grafica (ad esempio pgAdmin3, nativo di PostgreSQL) o da linea di comando, creare il database specificandone un nome, ad esempio cmdbuild:

```
CREATE DATABASE cmdbuild
    WITH OWNER cmdbuilduser
    ENCODING = 'UTF8';
```

- accedere al nuovo database cmdbuild e creare il linguaggio plpgsql:

```
CREATE LANGUAGE plpgsql;
```

- per creare un database con dati demo, eseguire nell'ordine alfabetico gli script che troverete nella directory directory **{CMDBUILD}/WEB-INF/sql/base_schema**, o in alternativa il file

```
{CMDBUILD}/WEB-INF/sql/sample_schemas/demo_schema.sql
```

- nel caso si scelga il database vuoto, è necessario verificare l'ultima patch disponibile analizzando la directory

  {CMDBUILD}/WEB-INF/patches

  prendendo nota del nome dell'ultima patch (in ordine alfabetico e senza estensione, per esempio “1.2.3”) ed eseguire questi comandi:

```
SELECT cm_create_class('Patch', 'Class', 'DESCR: |MODE: reserved|STATUS: active|SUPERCLASS: false|TYPE: class');
INSERT INTO "Patch" ("Code") VALUES ('1.2.3');
```

- eseguire i seguenti comandi SQL per creare l'utente di CMDBuild di tipo "Superuser" (nell'esempio con username **admin** e password **admin**):

```
INSERT INTO "User" ("Status", "Username", "IdClass", "Password", "Description")
```

1 Il driver JDBC può essere scaricato all'indirizzo http://jdbc.postgresql.org/download.html

```
        VALUES ('A', 'admin', '"User"', 'DQdKW32Mlms=', 'Administrator');
    INSERT INTO "Role" ("Status", "IdClass", "Administrator", "Code", "Description")
        VALUES ('A', '"Role"', true, 'SuperUser', 'SuperUser');
    INSERT INTO "Map_UserRole" ("Status", "IdClass1", "IdClass2", "IdObj1", "IdObj2", "IdDomain")
        VALUES ('A', '"User"'::regclass,'"Role"'::regclass, currval('class_seq')-2, currval('class_seq'),
'"Map_UserRole"'::regclass);
```

A questo punto è disponibile un database vuoto compatibile con il sistema CMDBuild.

Configurazione dell'applicazione

La configurazione manuale dell'applicazione prevede la modifica di alcuni file. Se Tomcat fosse già avviato si dovrà provvedere ad arrestarlo e a posizionarsi nella directory **{CMDBUILD}/WEB-INF/conf**.

E' necessario poi eseguire le seguenti operazioni:

- file **cmdbuild.conf**
  - aprirlo con un editor di testo
  - selezionare la lingua di default del sistema modificando la voce "language" (i valori accettati sono IT e EN, lasciando "true" la voce "languageprompt" dall'interfaccia di login sarà possibile selezionare di volta in volta la lingua)
  - salvare e chiudere il file
- file **database.conf**
  - decommentare le tre righe presenti
  - indicare nella voce "**db.url**" il nome del database da utilizzare
  - nelle voci "**db.username**" e "**db.password**" indicare le credenziali per accedere al database
  - salvare e chiudere il file

L'installazione è a questo punto terminata e riavviando Tomcat si potrà accedere a CMDBuild.

Configurazione dell'interfaccia fra CMDBuild e Together Workflow Server

La configurazione di Together Workflow Server prevede la modifica di alcuni file di configurazione. Se Tomcat fosse già avviato si dovrà provvedere ad arrestarlo ed eseguire poi le seguenti operazioni:

1) Indirizzo del database

Nel file **{SHARK}/META-INF/context.xml** della webapp di shark va configurato l'indirizzo del database modificando la riga:

```
url="jdbc:postgresql://localhost/${cmdbuild}"
```

Ad esempio nel caso in cui il database abbia il nome di default "**cmdbuild**" ed il server postgres sia in esecuzione sulla stessa macchina ed utilizzi la posta standard, sarà:

```
url="jdbc:postgresql://localhost/cmdbuild"
```

2) URL ed utente di servizio

Nel file **{SHARK}/conf/Shark.conf** impostare correttamente i seguenti parametri:

```
# CMDBUild connection settings
org.cmdbuild.ws.url=http://localhost:8080/cmdbuild/
org.cmdbuild.ws.username=workflow
org.cmdbuild.ws.password=changeme
```

inserendo rispettivamente l'url a cui risponde l'applicazione CMDBuild, lo username e la password del ServiceUser utilizzato da Together Workflow Server per utilizzare i servizi di CMDbuild.

3) Impostazione del ServiceUser per Together Workflow Server

Aprire con un editor di testo il file **{CMDBUILD}/WEB-INF/conf/auth.conf** e decommentare la riga

```
serviceusers.prigileged=workflow
```

Sostituento eventualmente a `workflow` il valore inserito per il parametro `org.cmdbuild.ws.username` al punto precedente.

4) Avvio dei servizi

Il servizio Tomcat di Shark deve essere già attivo nel momento in cui si inizia ad operare sui workflow tramite l'interfaccia utente di CMDBuild.

In ambiente Windows il path va espresso con la barra invertita:

```
DatabaseManager.ConfigurationDir=C:/srv/tomcat/webapps/shark/conf/dods
```

oppure doppia

```
DatabaseManager.ConfigurationDir=C:\\srv\\tomcat\\webapps\\shark\\conf\\dods
```

Un modo per verificare il corretto funzionamento dell'istanza Shark è quello di consultare il relativo file di log.

Riavviare quindi anche l'istanza tomcat di CMDBuild prima di proseguire con la configurazione.

5) Abilitazioni

Dal modulo di Amministrazione di CMDBuild bisogna entrare nel menu di Configurazione e:

- abilitare il workflow (con il relativo check)
- impostare l'URL a cui risponde il servizio Shark

6) Creazione dell'utente in CMDBuild ed assegnazione dei privilegi

Dal modulo di amministrazione di CMDBuild bisogna entrare nel menu “Utenti e Gruppi” e:

- alla voce Utenti creare un nuovo utente avente Nome utente e Password uguali ai valori definiti per i parametri org.cmdbuild.ws.username e org.cmdbuild.ws.password al punto 2)
- aggiungere l'utente appena creato ad un Gruppo avente privilegi di Amministratore (ad esempio SuperUsers oppure un gruppo ad-hoc avente il flag Amministratore abilitato)

Aggiornamento CMDBuild e Together Workflow Server

Aggiornamento CMDBuild

Di seguito si trova una lista di passaggi necessari per aggiornare CMDBuild:

1. spegnere Tomcat ed effettuare un backup dell'applicazione e del relativo database
2. salvare le configurazioni presenti in ${tomcat_home_cmdbuild}/webapps/${cmdbuild_instance}/WEB-INF/conf
3. se si usa il GIS, salvare le icone gis presenti in: ${tomcat_home_cmdbuild}/webapps/cmdbuild_instance}/upload/images/gis
4. cancellare la directory ${tomcat_home_cmdbuild}/webapps/${cmdbuild_instance}/
5. cancellare il contenuto della cartella work di Tomcat
6. spostare il war fornito nella directory webapps di Tomcat e avviare Tomcat
7. collegarsi a CMDBuild ed eseguire la configurazione specificando che il database esiste già e quale sia il suo nome
8. se necessario, quando la configurazione finisce, il sistema avviserà che bisogna aggiornare il database e, premendo "conferma", eseguità autonomamente le patch necessarie.
9. al momento della connessione al CMDBuild aggiornato si consiglia di pulire sempre la cache del browser

Dopo aver eseguito le operazioni descritte precedentemente, il sistema risulterà aggiornato. Bisognerà riattivare eventuali altre configurazioni (ad esempio alfresco, workflow). Suggeriamo di compiare i valori dai file di configurazuione salvati precedentemente.

Non è possibile sovrascrivere i file tra una versione e l'altra dato che questi potrebbero contenere parametri in più o in meno.

Ogni modifica manuale (da filesystem) alla configurazione richiede un riavvio di Tomcat.

Aggiornamento di Together Workflow Server

Si seguito si trova una lista di passaggi necessari per l'aggiornamento di Shark. E' opportuno che la versione di Shark (Together Workflow Server) e quella di CMDBuild siano sempre fra loro coerenti.

1. spegnere Tomcat
2. salvare i file ${tomcat_home_shark}/webapps/${shark_instance}/META-INF/context.xml e ${tomcat_home_shark}/webapps/${shark_instance}/conf/Shark.conf
3. rimuovere la directory {tomcat_home_shark}/webapps/${shark_instance}/
4. cancellare il contenuto della cartella work di Tomcat
5. copiare il nuovo war nelle webapps di Tomcat
6. avviare Tomcat e quando il server risulta è partito (verificare nel file catalina.out), spegnerlo in modo da ottenere all'interno della cartella webapps la directory di shark. In alternativa è possibile semplicemente creare una cartella shark e decomprimerci dentro il war come se fosse un qualsiasi archivio.
7. una volta fermato Tomcat, modificare ${tomcat_home_shark}/webapps/$

{shark_instance}/META-INF/context.xml e ${tomcat_home_shark}/webapps/$ {shark_instance}/conf/Shark.conf in modo che essi facciano riferimento rispettivamente al database di cmdbuild e all'applicazione (è possibile copiare i valori dai file salvati precedentemente, senza sovrascrivere i file)

8. se esiste, rimuovere il file ${tomcat_home_shark}/conf/Catalina/localhost/ {nome_istanza_shark}.xml
9. riavviare Tomcat

Si suggerisce caldamente di rimuovere o rinominare l'estensione di qualsiasi file war presente nella directory webapps di Tomcat una volta completata l'installazione.

Questa operazione va fatta a Tomcat spento.

Configurazione per l'accesso a un DMS tramite CMIS

CMDBuild consente l'integrazione con altri sistemi documentali tramite CMIS, uno standard aperto di interscambio di dati documentali tramite protocollo web, supportato da tutti i principali prodotti di settore (es. Liferay, Sharepoint)

L'adozione di tale protocollo permette di supportare le funzionalità base:

- possibilità di allegare file di qualsiasi tipo a schede dati di qualsiasi classe o processo
- attribuzione di una categoria a ciascun file
- visualizzazione dei file caricati

Nota: attualmente è supportata la gestione delle categorie e dei metadati solamente per il protocollo CMIS 1.1. Sarà possibile utilizzare anche DMS basati su CMIS 1.0, ma non sarà quindi possibile memorizzare in CMDBuild informazioni quali l'autore, la descrizione e la categoria.

Configurazione per la gestione delle categorie

In questa sezione viene indicato come configurare un DMS che supporti il protocollo CMIS 1.1 al fine di poter gestire la categorizzazione degli allegati durante le operazioni di caricamento ed aggiornamento dei medesimi sulle card di CMDBuild.

La gestione delle categorie è interamente contenuta all'interno di due file aspect definiti secondo i requisiti del proprio specifico DMS.

Di seguito si riporta un esempio fatto per configurare Alfresco 5.0.

Nota: i nomi utilizzati non sono vincolanti.

1) cmdbuild-model-context.xml

Questo file contiene la definizione dell'aspect per il salvataggio delle categorie, ed è strutturato come segue:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'
'http://www.springframework.org/dtd/spring-beans.dtd'>
<beans>
   <!-- Registration of new models -->
   <bean id="example.dictionaryBootstrap" parent="dictionaryModelBootstrap" depends-
on="dictionaryBootstrap">
      <property name="models">
         <list>
            <value>alfresco/extension/cmdbuildModel.xml</value>
         </list>
      </property>
   </bean>
</beans>
```

2) cmdbuildModel.xml

Questo file definisce le categorie utilizzabili

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- The important part here is the name - Note: the use of the my: namespace which is
```

```
defined further on in the document -->
<model name="cmdbuild:module" xmlns="http://www.alfresco.org/model/dictionary/1.0">

	<description>Custom Model for CMDBuild</description>
	<author>CMDBuild Team</author>
	<version>1.0</version>

	<imports>
		<!-- Import Alfresco Dictionary Definitions -->
		<import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d" />
		<!-- Import Alfresco Content Domain Model Definitions -->
		<import uri="http://www.alfresco.org/model/content/1.0" prefix="cm" />
	</imports>

	<!-- Introduction of new namespaces defined by this model -->
	<namespaces>
		<namespace uri="org.cmdbuild.dms.alfresco" prefix="cmdbuild" />
	</namespaces>

	<aspects>
		<aspect name="cmdbuild:classifiable">
			<title>Classification</title>
			<properties>
				<property name="cmdbuild:classification">
					<type>d:text</type>
				</property>
			</properties>
		</aspect>
	</aspects>

</model>
```

Entrambi i file devono essere copiati all'interno della directory preposta del DMS utilizzato, ad esempio per il DMS Alfresco:

${ALFRESCO_HOME}/tomcat/shared/classes/alfresco/extension

Nel caso di Alfresco il sistema va riavviato affinché le modifiche abbiano effetto.

Configurazione per la gestione dei metadati

In questa sezione viene descritto come configurare la gestione di metadati personalizzati al fine di poter gestire informazioni aggiuntive durante le operazioni di caricamento ed aggiornamento degli allegati sulle card di CMDBuild.

Anche in questo caso gli aspect devono essere definiti sulla base dei requisiti imposte dal DMS che si sta utilizzando. Quello che segue è un esempio relativo ad Alfresco.

1) Definizione del modello custom

Come previsto da Alfresco, va creato un modello custom in cui definire i metadati ed il loro raggruppamento.

All'interno dei file di progetto di CMDBuild (modulo dms/alfresco) è possibile trovarne un esempio.

Il nome di default per questo file è cmdbuildCustomModel.xml, ma può comunque essere personalizzato specificandolo nel file di configurazione dms.conf di CMDBuild:

alfresco.custom.model.filename=anotherNameForCustomModel.xml

Qui di seguito un esempio (estratto) del file:

```
<?xml version="1.0" encoding="UTF-8"?>
<model
   name="cmdbuild:customModule"
   xmlns="http://www.alfresco.org/model/dictionary/1.0">
   ...
   <namespaces>
        <namespace uri="org.cmdbuild.dms.alfresco" prefix="cmdbuild" />
   </namespaces>
   <constraints>
        <constraint name="cmdbuild:imageFormat" type="LIST">
             <parameter name="allowedValues">
                  <list>
                       <value>bmp</value>
                       ...
                  </list>
             </parameter>
        </constraint>
        ...
   </constraints>
   <types>
        <type name="cmdbuild:Document">
             <title>Document</title>
             <mandatory-aspects>
                  <aspect>cmdbuild:documentStatistics</aspect>
                  ...
             </mandatory-aspects>
        </type>
        ...
   </types>
   <aspects>
        <aspect name="cmdbuild:documentStatistics">
             <title>Document Statistics</title>
             <properties>
                  <property name="cmdbuild:characters">
                       <title>Number of characters</title>
                       <type>d:long</type>
                  </property>
                  ...
             </properties>
        </aspect>
        ...
   </aspects>
```

```
</model>
```

Definizione del namespace

Va definito un namespace come di seguito:

```
<namespaces>
    <namespace uri="org.cmdbuild.dms.alfresco" prefix="cmdbuild" />
</namespaces>
```

I valori sopra indicati sono quelli di default, ma è possibile personalizzarli specificando le seguenti voci nel file di configurazione dms.conf di CMDBuild:

```
alfresco.custom.uri=another.unique.uri
alfresco.custom.prefix=another_prefix
```

Si faccia attenzione che in buona parte degli elementi del documento xml, i nomi saranno preceduti dal prefisso qui definito.

Definizione di liste di valori

Per alcune proprietà di un aspect (vedi paragrafo “Definizione degli aspect” alla pagina successiva) è possibile specificare dei valori predefiniti che potranno essere selezionati attraverso l'uso di una combobox.

```
<constraints>
    <constraint name="cmdbuild:imageFormat" type="LIST">
            <parameter name="allowedValues">
                    <list>
                            <value>bmp</value>
                            ...
                    </list>
            </parameter>
    </constraint>
    ...
</constraints>
```

Importante: l'attributo name dell'elemento constraint dovrà essere uguale all'attributo name dell'elemento property definito all'interno dell'aspect (vedi paragrafo “Definizione degli aspect” alla pagina successiva).

Definizione dei tipi

I tipi qui definiti verranno associati alle categorie definite e gestite lato CMDBuild.

```
<types>
    <type name="cmdbuild:Document">
            <title>Document</title>
            <mandatory-aspects>
                    <aspect>cmdbuild:documentStatistics</aspect>
```

```
            ...
        </mandatory-aspects>
    </type>
    ...
</types>
```

In particolare:

- la definizione di un tipo è necessaria solamente se è necessario associare degli aspect (e quindi dei metadati) ai documenti gestiti da CMDBuild/Alfresco
- il contenuto dell'elemento title dovrà corrispondere all'attributo Description di una Lookup di CMDBuild (es. “Document”).

Definizione degli aspect

Gli aspect raggruppano un insieme di proprietà e più aspect diversi possono essere applicati ad uno stesso documento.

```
<aspects>
    <aspect name="cmdbuild:documentStatistics">
        <title>Document Statistics</title>
        <properties>
            <property name="cmdbuild:characters">
                <title>Number of characters</title>
                <type>d:long</type>
            </property>
            ...
        </properties>
    </aspect>
    ...
</aspects>
```

In particolare:

- per poter essere applicati, gli aspect vanno associati ad un tipo di documento (vedi paragrafo “Definizione dei tipi” alla pagina precedente)
- i tipi delle proprietà attualmente gestite sono: “text”, “int”, “long”, “float”, “date”, “datetime”, “boolean” così come definiti da Alfresco
- è possibile gestire delle liste di valori di tipo “text” (vedi paragrafo “Definizione di liste di valori” alla pagina precedente).

2) Definizione delle regole di autocompletamento

Va creato un file xml ad hoc, che verrà interpretato da CMDBuild, che dovrà contenere le informazioni necessarie per permettere l'autocompletamento dei metadati a partire dagli attributi della card a cui viene allegato un documento.

Il nome di default per questo file è metadataAutocompletion.xml, ma può comunque essere personalizzato specificandolo nel file di configurazione dms.conf di CMDBuild:

```
metadata.autocompletion.filename=anotherNameForAutocompletionRules.xml
```

Qui di seguito un esempio:

```
<?xml version="1.0" encoding="UTF-8"?>
<autocompletion>
    <metadataGroups>
        <metadataGroup name="documentStatistics">
            <metadata name="character">
                <rules>
                    <rule classname="Employee" value="123" />
                    <rule classname="Asset" value="client:..." />
                </rules>
            </metadata>
            ...
        </metadataGroup>
    </metadataGroups>
</autocompletion>
```

In particolare:

- se non è richiesto alcun autocompletamento allora il file può essere omesso
- l'attributo name dell'elemento metadataGroup rappresenta il nome (a meno del prefisso) di un aspect (vedi paragrafo “Definizione degli aspect” alle pagine precedenti)
- l'attributo name dell'elemento metadata rappresenta il nome (a meno del prefisso) di una proprietà di un aspect (vedi Definizione degli aspect)
- ogni elemento rule indica come completare il valore del metadato, questo valore può essere un valore fisso o un'espressione gestita dal Template Resolver (si veda la documentazione specifica).

3) Installazione

Dati i seguenti file:

- cmdbuildCustomModel.xml
- metadataAutocompletion.xml

All'interno della directory delle estensioni di Alfresco (es. $ALFRESCO_HOME/tomcat/shared/classes/alfresco/extension):

- copiare il file cmdbuildCustomModel.xml
- creare un file cmdbuild-custom-model-context.xml costruito in questo modo (facendo attenzione a specificare il nome del file del modulo custom corretto)

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'
'http://www.springframework.org/dtd/spring-beans.dtd'>
<beans>
    <!-- Registration of new models -->
    <bean
```

```
        id="extension.dictionaryBootstrap"
        parent="dictionaryModelBootstrap"
        depends-on="dictionaryBootstrap">
        <property name="models">
            <list>
                <value>
                    alfresco/extension/cmdbuildCustomModel.xml
                </value>
            </list>
        </property>
    </bean>
</beans>
```

- creare un file web-client-config-custom.xml contenente le definizioni attraverso cui le proprietà degli aspect verranno visualizzate all'interno dell'interfaccia web di Alfresco (si rimanda alla documentazione ufficiale)
- avviare/riavviare Alfresco affinchè le modifiche abbiano effetto, verificare nel log eventuali errori di definizione del modulo
- All'interno della directory di configurazione di CMDBuild:
- copiare il file cmdbuildCustomModel.xml
- copiare il file metadataAutocompletion.xml
- eseguire (nell'ordine) la pulizia della cache del server e del client

Nota: i nomi dei file non sono vincolanti.

Configurazione dell'interfaccia fra CMDBuild e Alfresco

1) Configurazione lato CMDBuild

Per configurare l'utilizzo di un DMS tramite CMIS con CMDBuild posizionarsi nella directory {CMDBUILD}/WEB-INF/conf ed aprire con un editor di testo il file **dms.conf** .

É necessario poi eseguire le seguenti operazioni:

- impostare **enabled**=true
- impostare **dms.service.type**=cmis:
- verificare che alla voce "**dms.service.cmis.url**" sia inserito l'url a cui raggiungere il dms tramite cmis
- impostare alle voci "**dms.service.cmis.user**" "**dms.service.cmis.password**" le credenziali di accesso
- impostare alla voce "**dms.service.cmis.path**" il path del filesystem da usare
- impostare alla voce "**dms.service.cmis.model**" il nome del model da usare
- specificare alla voce "**category.lookup**" la categoria di CMDBuild utilizzata

2) Configurazione lato Alfresco

Fare riferimento al capitolo "Installazione e utilizzo del sistema DMS Alfresco 3.4"

Installazione e utilizzo del sistema DMS Alfresco 3.4

Qualora non si disponga di un DMS che supporti il protocollo CMIS 1.1 è possibile usare il caricamento degli allegati e dei relativi metadati e categorie tramite I servizi FTP e webservice offerti da Alfresco 3.4.

Di seguito si spiega come installare Alfresco 3.4 e come configurarlo per effettuare upload e download dei file, per categorizzarli ed eventualmente per caricare eventuali metadati personalizzati.

Alfresco mette a disposizione per la sua installazione un wizard, sia grafico sia testuale, per semplificare l'installazione e la configurazione delle impostazioni fondamentali.

In particolare è necessario specificare quanto segue:

- le porte utilizzate da Tomcat
- la porta utilizzata dal server FTP
- le impostazioni relative al database

Qualora si decida di procedere utilizzando l'installazione manuale si può comunque fare riferimento alle successive note di installazione, relative all'installazione di Alfresco 3.4.

Nota: nel seguito indicheremo con **{ALFRESCO}** la directory di installazione di alfresco (es. /opt/alfresco).

Configurazioni di base

1) Configurazione di Tomcat

Configurare l'avvio del server Tomcat utilizzato da Alfresco su porte diverse dal server Tomcat utilizzato da CMDBuild.

Accedere alla directory

```
{ALFRESCO}/tomcat/conf
```

Aprire per modifica il file **server.xml** e modificare le proprietà:

```
<Server port="8005" shutdown="SHUTDOWN">
...
<Connector port="8080" protocol="HTTP/1.1" ... />
...
<Connector port="8009" protocol="AJP/1.3" ... />
```

in modo da non creare conflitti con l'istanza di Tomcat usata da CMDBuild.

Ad esempio si possono utilizzare i seguenti (facendo comunque sempre attenzione ad altre istanze di Tomcat presenti sul sistema):

```
<Server port="9005" shutdown="SHUTDOWN">
...
<Connector port="9080" protocol="HTTP/1.1" ... />
...
<Connector port="9009" protocol="AJP/1.3" ... />
```

2) Configurazione del repository e del database

Accedere alla directory **{ALFRESCO}/tomcat/shared/classes/** ed editare il file **alfresco-global.properties** (qualora non sia presente è possibile usare una copia del file **alfresco-global.properties.sample**).

Modificare la seguenti proprietà:

```
dir.root={REPOSITORY DIRECTORY}
```

per definire la posizione del repository in cui verranno salvati i documenti (ad esempio: C:/alfresco/repository per sistemi Windows o /var/alfresco/repository per sistemi *nix).

Alfresco supporta i seguenti database: HSQL (default), MySQL, Oracle, Sybase, SQLServer e PostgreSQL. Si suggerisce di indirizzare Alfresco su un nuovo database gestito dallo stesso server PostgreSQL utilizzato per CMDBuild. Dal momento che in fase di installazione non è possibile effettuare questa scelta, è necessario procedere manualmente.

Impostare le seguenti proproprietà sempre nel file **alfresco-global.properties**:

```
db.driver=org.postgresql.Driver
db.username=postgres
db.password=postgres
db.url=jdbc:postgresql://localhost:5432/alfresco
```

Copiare i driver per l'accesso al database di Postgres (libreria di interfaccia presente nella directory /extras/tomcat-libs/[TomcatVersion] della distribuzione CMDBuild) in **{ALFRESCO}/tomcat/lib**.

Tramite uno strumento con interfaccia grafica (ad esempio pgAdmin3 di PostgreSQL), oppure da linea di comando, creare il database utilizzando il nome specificato nella proprietà "**db.url**" sopra riferita (nell'esempio "alfresco").

Sempre nel file **alfresco-global.properties** editare le seguenti linee per abilitare il server FTP:

```
ftp.enabled=true
ftp.port=1121
ftp.ipv6.enabled=false
```

Fare attenzione perchè se sullo stesso host è già presente un altro server FTP, allora è necessario modificare la porta sulla quale girerà il server FTP di Alfresco.

La porta specificata dev'essere corrispondente a quella indicata nel file:

```
{CMDBUILD}\WEB-INF\conf\dms.conf
```

Al termine, una volta che la creazione del database iniziale è andata a buon fine, arrestare il servizio Alfresco e verificare che nel file **alfresco-global.properties** la proprietà "**db.schema.update**" sia settata a *false* (o eventualmente commentata):

```
db.schema.update=false
```

Configurazioni aggiuntive

E' poi necessario creare in Alfresco lo "spazio" ("space") destinato a contenere gli allegati di CMDBuild e la categoria associata a tali documenti.

Al termine avviare Alfresco e, se non vengono evidenziati problemi, dopo qualche minuto si potrà effettuare l'accesso ad Alfresco tramite web browser all'indirizzo:

```
http://{SERVER}:{PORT}/alfresco
```

Le credenziali di accesso di default sono "admin" sia come nome utente che come password.

Nel pannello "navigator" accedere a "Company Home", quindi a "User Homes" e creare uno spazio con nome uguale a quello indicato nel file:

{CMDBUILD}\WEB-INF\conf\dms.conf (ad esempio "cmdbuild")

Recarsi ora nella consolle di amministrazione di Alfresco (prima icona della barra superiore), accedere alla "Category Management" e creare una nuova categoria con nome e descrizione uguali al nome indicato nel file:

{CMDBUILD}\WEB-INF\conf\dms.conf (ad esempio "AlfrescoCategory")

A questo punto la configurazione è terminata, riavviando Tomcat sarà possibile gestire allegati anche da CMDBuild.

Configurazione per la gestione dei metadati

Si veda la sezione analoga nel capitolo “Configurazione per l'accesso ad un DMS tramite CMIS”

Configurazione dell'interfaccia fra CMDBuild e Alfresco

1) Configurazione lato CMDBuild

Per configurare l'utilizzo di Alfresco 3.4 Community con CMDBuild posizionarsi nella directory {CMDBUILD}/WEB-INF/conf ed aprire con un editor di testo il file **dms.conf** (**legacydms.conf** per versioni di CMDBuild antecedenti alla 1.4).

É necessario poi eseguire le seguenti operazioni:

- impostare **enabled**=true
- impostare **dms.service.type**=cmis:
- verificare che alla voce "**server.url**" sia inserito l'url a cui raggiungere Alfresco tramite webservices
- modificare fileserver.port con il numero di porta su cui è attivo il servizio FTP di Alfresco (vedi paragrafo successivo)
- modificare la voce "**repository.fspath**" con il path dello space di CMDBuild su Alfresco, analogamente per la voce "**repository.app**"
- specificare alla voce "**category.lookup**" la categoria di CMDBuild utilizzata in Alfresco

2) Configurazione lato Alfresco

Fare riferimento al capitolo "Installazione e utilizzo del sistema DMS Alfresco 3.4"

Backup dei dati di CMDBuild

Per effettuare il backup dei dati di CMDBuild basta effettuare il backup del database PostgreSQL.

Se non si vuole procedere tramite tool grafici come, ad esempio, PgAdmin III si può procedere a riga di comando, specificando, ad esempio:

```
pg_dump --host localhost --port 5432 --username "postgres" --format custom --verbose --file /backup/cmdbuild/cmdbuild.backup cmdbuild_db
```

Tale comando farà un backup del db chiamato “cmdbuild_db”, sul server in locale in ascolto sulla porta 5432 utilizzando lo username postgres. Il formato sarà compresso (--format custom) e verrà scritto nella cartella /backup/cmdbuild/cmdbuild.backup. Inoltre il comando stamperà il dettaglio di quello che sta facendo a video (--verbose).

Per spiegazioni più approfondite sulla sintassi di pg_dump vi rimandiamo alla documentazione di postgres (su Linux è possibile digirare “man pg_dump” per avere un'eccellente descrizione del comando).

Un esempio di file per fare il backup dei database:

```
#!/bin/sh
# Executes a backup of the specified postgresql databases
#
# Steps:
# - dumps (compressed) the database into the target directory
# - archive (tgz) the backup
# - removes the backup

PG_DUMP="pg_dump"
PG_DUMP_OPTS="-h localhost -p 5432 -U postgres -F c --column-inserts"

TIMESTAMP=`date +%Y%m%d%H%M%S`

BACKUP_ROOT="$1"

while [ $# -gt 1 ]
do
        shift
        DB_NAME="$1"

        cd "${BACKUP_ROOT}"

        DB_BACKUP="${DB_NAME}-${TIMESTAMP}.backup"
        DB_ARCHIVE="${DB_BACKUP}.tar.gz"
```

```
        ${PG_DUMP} ${PG_DUMP_OPTS} -f "${DB_BACKUP}" "${DB_NAME}"

        if [ ! -r "${DB_BACKUP}" ];
        then
                logger "File '${DB_BACKUP}' not found"
        else
                tar -czf "${DB_ARCHIVE}" "${DB_BACKUP}"
                logger "Database '${DB_NAME}' has been successfully backed up"
                rm "${DB_BACKUP}"
        fi
done
```

Backup dei dati di Alfresco

Per effettuare dei dati contenuti su Alfresco basta effettuare il backup del database (di seguito assumeremo che risieda nel database PostgreSQL).

Come per CMDBuild il comando per effettuare un backup del database a riga di comando è il seguente:

```
pg_dump --host localhost --port 5432 --username "postgres" --format custom --verbose --file /backup/alfresco/alfresco.backup alfresco_db
```

Inoltre, è necessario effettuare un backup, sottoforma di archivio zip o tar.gz, anche dell'intera cartella che contiene il repository in cui vengono salvati i documenti (es: C:/alfresco/repository per sistemi Windows o /var/alfresco/repository per sistemi *nix).

```
#!/bin/sh
ALFRESCO_SERVICE="/etc/init.d/alfresco"

TT_PG_BACKUP="/usr/local/bin/tt_pg_backup.sh"

BACKUP_HOME="/backup"
ALFRESCO_BACKUP_HOME="${BACKUP_HOME}/alfresco"
BACKUP_LOG="/var/log/cmdbuild/crontab-backup.log 2>&1"

ALFRESCO_DATABASE="alfresco"

ALFRESCO_DATA="/var/lib/alfresco/data"
TIMESTAMP=`date +%Y%m%d%H%M%S`
ALFRESCO_DATA_BACKUP="alfresco-data-${TIMESTAMP}.tar.gz"

logger "Stopping Alfresco service"
${ALFRESCO_SERVICE} stop
sleep 5

logger "Backupping Alfresco database"
${TT_PG_BACKUP} "${ALFRESCO_BACKUP_HOME}" "${ALFRESCO_DATABASE}" > ${BACKUP_LOG}

logger "Backupping Alfresco data"
tar czf "${ALFRESCO_BACKUP_HOME}/${ALFRESCO_DATA_BACKUP}" ${ALFRESCO_DATA}
logger "Alfresco database/data have been backed up."
```

```
logger "Starting Alfresco service"
${ALFRESCO_SERVICE} start
```

Schedulazione dei backup

Per effettuare una schedulazione periodica del backup dei dati su sistemi Linux si consiglia di procedere nel seguente modo:

1. creare un file /etc/cron.d/backup
2. nel file backup inserire, con la sintassi propria di cron, i comandi da eseguire.

Per una maggiore pulizia raccomandiamo di creare diversi script singoli, uno per sistema, da chiamare in questo file piuttosto che scrivere direttamente i comandi nel file cron stesso.

Un esempio:

```
00 19 * * *     root   /usr/local/bin/tt_pg_backup.sh /backup/cmdbuild cmdbuild &>> /var/log/cmdbuild/backup/backup.log
10   19   *   *   7            root      /usr/local/bin/alfresco-backup.sh   &>> /var/log/cmdbuild/backup/backup.log
```

Modalità di autenticazione

Introduzione

CMDBuild offre tramite apposite configurazioni la possibilità di demandare il controllo di autenticazione di accesso a servizi esterni.

Tale possibilità riguarda il controllo dell'account (username e password), mentre rimangono gestiti all'interno di CMDBuild i profili e permessi del gruppo a cui appartiene l'utente.

I parametri per configurare il comportamento di CMDBuild in fase di autenticazione sono indicati nel file **auth.conf** presente nella directory WEB-INF della webapps di CMDBuild all'interno di Tomcat.

Da questo file è possibile.

Il file è suddiviso in 4 sezioni:

- configurazione della tipologia di autenticazione
- configurazione header authentication
- configurazione autenticazione tramite protocollo LDAP
- configurazione single sign on tramite CAS

Configurazione della tipologia di autenticazione

Si riportano di seguito i parametri da impostare in fase di configurazione, descrivendone il relativo significato.

1) auth.methods

Con questo parametro è possibile definire la "catena" di autenticazione di CMDBuild.

È possibile, cioè, definire, in cascata, quali tipologie di autenticazione utilizzare per permettere l'accesso all'utente, definendone la priorità.

Esempio

```
auth.methods=LdapAuthenticator,DBAuthenticator
```

Con la configurazione dell'esempio precedente si indica a CMDBuild che ogni volta che un utente esegue il login sul sistema, CMDBuild dovrà verificarne le credenziali prima tramite LDAP e, qualora fallisse, sulla base di dati di CMDBuild.

I parametri accettati sono:

- HeaderAuthenticator (autenticazione tramite controllo dell'header)
- LdapAuthenticator (autenticazione con verifica delle credenziali su LDAP)
- CasAuthenticator (single sign tramite CAS)
- DBAuthenticator (autenticazione standard)

2) serviceusers

Con questo parametro è possibile definire gli utenti "di servizio" di CMDBuild. Questa tipologia di utenti privilegiati è prevista ad uso esclusivo di sistemi esterni quali, ad esempio, la Portlet Liferay,

per cui il login da interfaccia per quella tipologia di utenti sarà disabilitato.

3) force.ws.password.digest

Questo parametro, se impostato a "true", forza l'utilizzo della specifica Username Token con password digest per l'autenticazione tramite webservice.

Impostando invece il parametro a "false" sarà possibile utilizzare anche password plain text per l'autenticazione tramite Username Token. Quest'ultimo caso può essere utile combinato con l'utilizzo di LDAP per il controllo degli accessi in CMDBuild.

Configurazione Header Authentication

Da questa sezione è possibile configurare l'autenticazione tramite verifica dell'header.

Per farlo è sufficiente editare il file **auth.conf** specificando il nome dell'attributo, presente nell'header HTTP, da utilizzare per autenticare l'utente su CMDBuild (header.attribute.name).

Questa tipologia di autenticazione prevede la presenza di un sistema di autenticazione a monte, che si occupa di generare degli header specifici che poi potranno essere utilizzati da CMDBuild per autorizzare l'accesso.

Configurazione autenticazione LDAP

La presente sezione documenta le modalità per configurare l'autenticazione in CMDBuild tramite protocollo LDAP.

CMDBuild attualmente supporta solamente l'autenticazione "simple bind". È tuttavia possibile utilizzare l' "anonymous bind" per la ricerca dell'utente nell'albero LDAP.

Affinché possano essere gestiti i permessi degli utenti in CMDBuild è necessario che gli utenti che dovranno accedere in CMDBuild siano presenti anche all'interno della webapp.

Ad esempio se l'utente con UID su LDAP m.rossi deve poter accedere a CMDBuild come utente del gruppo "Tecnici" devono essere eseguiti questi passi:

- creazione dell'utente m.rossi in CMDBuild con una password di default (non necessariamente quella di LDAP)
- creazione del gruppo Tecnici e definizione dei relativi permessi
- aggiunta dell'utente m.rossi al gruppo Tecnici

A questo punto, quando m.rossi si autenticherà, verranno verificate (in base all'ordine della catena di autenticazione definita in auth.methods) le credenziali rispetto all'albero LDAP.

Di seguito una descrizione dei parametri di configurazione.

1) ldap.server.address

Questo attributo serve a specificare l'indirizzo a cui è possibile raggiungere il server LDAP.

Esempio:

```
ldap.server.address=localhost
```

2) ldap.server.port

Questo attributo serve a specificare la porta del server LDAP. Il default è 389.

Esempio:

```
ldap.server.port=389
```

3) ldap.use.ssl

Indica se va utilizzata una connessione cifrata verso il server LDAP. Disabilitato di default.

Esempio:

```
ldap.use.ssl=true
```

4) ldap.basedn

In questa attributo và specificato il Base DN che sarà utilizzato per effettuare le query sull'albero LDAP.

Esempio:

```
ldap.basedn=dc=example,dc=com
```

5) ldap.bind.attribute

Questo attributo indica l'attributo sul quale dovrà essere eseguito il bind dell'utente.

Ad esempio, indicando come attributo per il bind uid e considerando il base dn indicato al punto precedente, la query LDAP che generata sarà uid=username,dc=example,dc=com.

Esempio:

```
ldap.bind.attribute=uid
```

6) ldap.search.filter

È possibile specificare, con questo attributo, un filtro di ricerca da utilizzare per la ricerca

Configurazione single sign on tramite CAS

La presente sezione documenta le modalità per configurare il single sign on (SSO) su CMDBuild tramite protocollo CAS.

L'autenticazione funziona nel seguente modo:

- l'utente richiede l'url di CMDBuild
- l'autenticatore CAS invia la richiesta al server CAS (${cas.server.url} + ${cas.login.page}) indicando l'url di CMDBuild (nel parametro ${cas.service.param}) a cui vuole accedere
- il server CAS risponde con un ticket (parametro ${cas.ticket.param}) da cui è possibile estrarre lo username
- se lo username viene correttamente validato/estratto allora CMDBuild procede con il login

Affinché possano essere gestiti i permessi degli utenti in CMDBuild è necessario che gli utenti che dovranno accedere in CMDBuild siano presenti anche all'interno della webapp.

Di seguito una descrizione dei parametri di configurazione.

1) cas.server.url

Questo attributo serve a specificare l'indirizzo a cui è possibile raggiungere il server CAS.

Esempio:

cas.server.url=https://cas.mycompany.com/cas

2) cas.login.page

Questo attributo serve a specificare la pagina di login di CAS

Esempio:

cas.login.page=/login

3) cas.service.param

Il nome del parametro che CAS userà per specificare l'url del servizio da autenticare.

Esempio:

cas.service.param=service

4) cas.ticket.param

Il nome del parametro che CAS userà per specificare il riferimento del ticket da lui generato fintanto che l'autenticazione sarà valida.

Esempio:

cas.ticket.param=ticket

Accesso alle risorse di CMDBuild via URL

Introduzione

CMDBuild consente di posizionarsi via URL su una propria risorsa.

Attualmente sono gestite risorse di tipo “classi”, ma in futuro il meccanismo sarà esteso per gestire filtri, report, ecc

Nel caso l'utente sia già autenticato la richiesta di URL compatibile con i criteri implementati provocherà il posizionamento su quella risorsa, altrimenti sarà proposta la pagina di login e poi si verrà posizionati sulla risorsa richiesta.

La funzionalità può essere utile ad esempio per inserire nelle mail di notifica di CMDBuild il link alla pagina del ticket, per consentire al software di gestione di un centralino telefonico di aprire la scheda di un cliente, per visualizzare la scheda di un asset leggendo un codice QR, ecc.

Esempi di URL validi

Seguono esempi di URL validi secondo i criteri di implementazione previsti.

Ricordiamo che attualmente sono utilizzabili gli URL di posizionamento su una classe e su una scheda dato il valore di un suo campo chiave (i primi tre esempi riportati sotto).

Le ulteriori estensioni saranno progressivamente attivate nei prossimi rilasci.

Tutti gli URL si intendono relativi all'indirizzo base di CMDBuild. Nel caso ad esempio dell'istanza demo on-line di CMDBuild l'URL #classes/Employee/cards/ dovrebbe essere quindi inteso come:

http://demo.cmdbuild.org/cmdbuild/index.jsp#classes/Employee/cards/

Seguono alcuni esempi:

#classes/Employee/cards/
visualizzazione card di una classe
#classes/Building/cards/76/
selezione card con “Id” specifico
#classes/Employee/cards/NumeroTelefono~123456/
selezione card da numero di telefono utilizzando il filtro semplice
#classes/Employee/cards?clientFilter={"attribute":{"simple":
{"attribute":"Phone","operator":"equal","value":["76543"]}}}
selezione card da numero di telefono
#classes/Building/cards?clientFilter={"attribute":{"simple":
{"attribute":"Description","operator":"contain","value":["Building"]}}}
selezione card da filtro generico
#classes/Building/cards/76/print?format=pdf
stampa card in versione PDF
#classes/Building/print?format=csv
stampa della griglia della classe in formato CSV

Attivazione dell'interfaccia “mobile”

Introduzione

CMDBuild "mobile" è una interfaccia utilizzabile su smartphone e tablet per eseguire le funzionalità applicative utili durante le attività “sul campo”.

E' realizzata con Sencha Touch (il framework Javascript sviluppato da Sencha, lo stesso produttore del framework Ext JS utilizzato da CMDBuild desktop) ed è in grado di interagire con CMDBuild tramite il webservice REST.

CMDBuild mobile implementa le principali funzionalità dell'interfaccia desktop: multilingua, login multigruppo, menu di navigazione, gestione classi con relazioni e allegati, ricerche e filtri, gestione workflow con i widget più utilizzati, gestione report, utilità (impostazioni "app" e gestione log).

Componenti e architettura

L’APP è realizzata utilizzando i seguenti componenti:

- Sencha Touch
  framework Javascript creato da Sencha
- Cordova
  framework mobile cross-platform
- Deft JS
  estensione per applicazioni mobile enterprise sviluppate con Sencha Touch
- log4javascript
  javascript logging framework
- Crosswalk (Android)
  tool per il deploy dell’applicazione su webview custom indipendente dalla versione di Android
- Siesta
  libreria per unit test / integration test

Lato server lo strato di web service REST è sviluppato utilizzando il framework Apache CXF, già integrato in CDMBuild e utilizzato per lo strato web service SOAP.

Il sistema ricalca l’architettura software di web service REST, con caratteristiche di:

- funzionalità divise in risorse web
- risorse indirizzabili (URI)
- metodi standard HTTP (GET, PUT, POST, DELETE)
- JSON media type
- link per la navigazione delle risorse
- stateless

Segue un disegno dell'architettura:

Compatibilità

- Android 4.0.3 o superiore
- iOS 6 o superiore

Limitazioni di utilizzo

L'interfaccia “mobile” è resa disponibile con licenza non open source, che ne consente l'utilizzo solamente a chi ha sottoscritto un servizio di manutenzione sull'applicazione CMDBuild con Tecnoteca Srl e fino a che tale servizio è attivo, con un limitato costo aggiuntivo.

Attivazione dell'interfaccia GUI Framework

Introduzione

Il GUI Framework risponde alle stesse necessità di rendere disponibile una interfaccia semplificata a personale non tecnico.

Il GUI Framework fornisce le seguenti caratteristiche principali:

- è attivabile in portali basati su tecnologie diverse, in quanto sviluppato in ambiente javascript / JQuery
- consente una libertà pressochè illimitata nella progettazione del layout grafico, definibile tramite un descrittore XML e con possibilità di intervenire sul foglio stile CSS
- garantisce tempi ridotti di configurazione grazie a funzioni predefinite (logiche di comunicazione, di autenticazione, ecc) ed a soluzioni grafiche native (form, grid, pulsanti di upload ed altri widget)
- interagisce con CMDBuild tramite il webservice REST
- è in grado di raccogliere dati da database di altre applicazioni permettendo quindi la gestione di soluzioni miste

Configurazione

Il framework definisce pagine HTML a partire da una loro definizione in XML.

Tali pagine possono essere inserite all'interno di un documento HTML permettendo al framework di inserire all'interno di portali esistenti punti di accesso ai dati di CMDBuild.

La configurabiltà del sistema è ottenibile tramite la definizione di propri fogli di stile e di script Javascript personalizzati.

L'elemento che va inserito nell'html del portale e' un contenitore Html (DIV, IFRAME …) nel seguente formato:

```
    <script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script type="text/javascript" src="http://10.0.0.107:8080/cmdbuild-gui-
framework/api/cmdbuildAP.js"></script>
    <script type="text/javascript">
    $( document ).ready(function(){
        $("#cmdbuilForm").cmdbuildAP({
            apiUrl : 'http://10.0.0.107:8080/cmdbuild/services/rest/',
            appRootUrl : 'http://10.0.0.107:8080/cmdbuild-gui-framework/api/',
            appConfigUrl : 'http://10.0.0.107:8080/cbap/cbap/config/bologna/',
            cqlUrl : 'http://10.0.0.107:8080/cbap/cbap/cql/',
            customjs : [
                'backend/GUIForm.js',
                'backend/Process.js'
        ],
            start: "home.xml",
            theme: [
                "jquery-ui.theme-crt-toscana.min.css",
                "custom.css"
            ],
     });
    });
```

```
</script>
<div id="cmdbuilForm"></div>
```

La configurazione dello stile e' delegata a file CSS direttamente caricabili dal tag Html. Questo rende la GUI estremamente adattabile, dando la possibilità al programmatore di includerla nel portale in modo che non sembri un elemento estraneo.

Sono inoltre definibili i server dove è contenuta l'applicazione, il sistema CMDBuild, i file di configurazione delle maschere ed il motore Cql, il linguaggio di interrogazione nativo in CMDBuild.

Nel tag customjs vengono definiti alcuni file di personalizzazione della GUI.

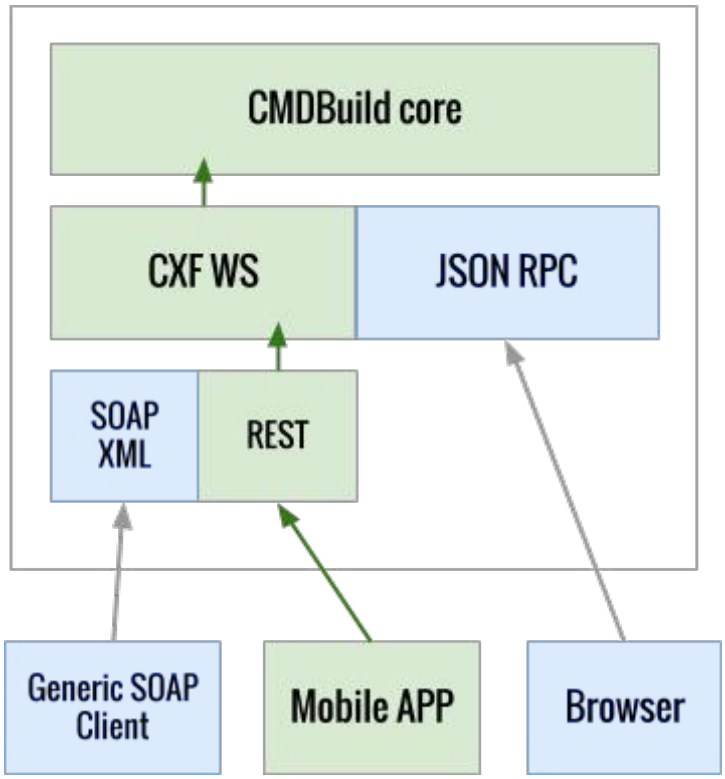
Il tag carica tutto il motore della GUI configurandola con i file visti sopra.

Le maschere vengono definite in XML tramite un linguaggio che emula HTML, con la ulteriore possibilità di poter definire form di gestione dati direttamente collegate ai metadati definiti nell'applicazione CMDBuild.

Tutti i tag, i comandi e comportamenti della GUI sono riconfigurabili dal programmatore secondo i criteri previsti nel sistema.

Poichè la libreria fa riferimento a JQuery, su cui il GUI Framework è stato sviluppato, il programmatore può nativamente utilizzare tutti i plugin compatibili con JQuery arricchendo secondo necessità l'insieme delle funzionalità disponibili.

Un altro elemento che caratterizza il funzionamento della GUI è il meccanismo dei "backend", cioè le classi Javascript che collegano la GUI ai server. In questo modo l'applicazione dispone di una ulteriore libertà data dalla possibilità di definire sia il formato dei dati attesi dal server sia il nome del server a cui collegarsi.



Attivazione della portlet Liferay

Introduzione

Per quanto sia una soluzione superata dalla successiva disponibilità del GUI Framework, CMDBuild rende ancora disponibile una portlet JSR186 tramite cui è possibile esportare alcune sue funzionalità applicative all'interno di portali intranet compatibili (la portlet CMDBuild è al momento certificata con il portale open source Liferay).

Le portlet JSR 168 sono componenti web Java utilizzabili come plugin all'interno di "container" quali i portali web compatibili con tale standard.

Con tale meccanismo è possibile configurare in modo personalizzato il portale disponendo le portlet nelle pagine di interesse e condividendo alcuni servizi del sistema ospite fra cui quello di autenticazione.

Come nel paragrafo precedente le portlet comunicano poi con CMDBuild tramite il relativo webservice.

La portlet CMDBuild consente di rendere disponibili alcune funzionalità ad utenti non tecnici e che avrebbero quindi delle difficoltà ad utilizzare l'interfaccia standard dell'applicazione.

La portlet comprende in particolare le funzioni di:

- gestione di una scheda dati (inserimento, modifica, cancellazione)
- avvio e avanzamento di un processo
- esecuzione di un report

La configurazione prevede due aspetti:

- configurazione della portlet CMDBuild in Liferay

- configurazione dell'applicazione CMDBuild

La portlet CMDBuild JSR 168 include tre diverse possibilità di gestione:

- accesso solamente agli utenti registrati come utenti di sistema di CMDBuild
- accesso anche ad utenti di tipo "guest" (cioè non registrati come utenti di sistema di CMDBuild), ma presenti in una classe "applicativa" configurata in CMDBuild (ad esempio Personale, Fornitore, ecc)
- accesso anche ad utenti di tipo "guest" (cioè non registrati come utenti di sistema di CMDBuild), che non siano nemmeno presenti in una classe "applicativa" configurata in CMDBuild (ad esempio Cittadini, Studenti, ecc)

Configurazione della portlet CMDBuild

Le opzioni di configurazione sono registrate in due file:

- portlet.properties (è un file "template" e non va modificato)
- portlet-ext.properties (sovrascrive le proprietà del file "template", tutte le personalizzazioni vanno inserite in questo file)

I due file sono archiviati nella directory:

WEB-INF/classes

all'interno della webapp della portlet.

I principali parametri di configurazione sono:

- **cmdbuild.url**: URL del webservice di CMDBuild (default http://localhost:8080/cmdbuild/services/soap/Private)
- **cmdbuild.user**: utente di accesso del webservice (default "portlet")
- **cmdbuild.password**: la password di accesso al webservice (default "portlet")
- **cmdbuild.group**: il gruppo di appartenenza dell'utente indicato sopra (default "Guest", da creare in CMDBuild)
- **user.class**: la classe "applicativa" di CMDBuild contenente la lista degli utenti che si vuole far accedere alla portlet
- **user.attribute.username**: l'attributo contenente lo username, all'interno della classe indicata al punto precedente (default "Username")
- **user.attribute.email**: l'attributo contenente l'indirizzo email, all'interno della classe indicata al punto precedente (default Email)
- **auth.method**: il criterio di autenticazione, che può assumere il valore "username" o "email" (default "email")

Esempio:

```
cmdbuild.url=http://localhost:8080/cmdbuild-test/services/soap/Private
user.class=Employee
```

Queste valori sovrascrivono le proprietà di default con un nuovo URL ed un nuovo nome di classe.

Attenzione

Affinché le modifiche ai parametri di configurazione abbiano effetto deve essere riavviato Liferay.

Configurazione di CMDBuild

Nel seguito ci riferiremo con ${CMDBUILD} alla directory contenente la "webapp" di CMDBuild all'interno di Tomcat.

Webservice user

E' necessario accedere al file **${CMDBUILD}/WEB-INF/conf/auth.conf** e modificare la proprietà **serviceusers** in accordo con il valore definito per la proprietà **cmdbuild.user** definito nel file **portlet-ext.properties**. Affinché la modifica abbia effetto deve essere poi riavviato CMDBuild.

Poi, dall'interno del modulo di Amministrazione di CMDBuild, bisognerà:

- creare un nuovo utente (come definito nella proprietà **cmdbuild.user** del file **portlet-ext.properties**)
- creare un nuovo gruppo (come definito nella proprietà **cmdbuild.group** del file **portlet-ext.properties**)
- aggiungere il nuovo utente al nuovo gruppo
- impostare il nuovo gruppo come gruppo di "**default login**" per il nuovo utente (necessario per l'autenticazione di tipo "guest")
- creare un menu "custom" per il nuovo gruppo (opzionale)

GeoServer

Introduzione

CMDBuild include la possibilità di gestire il georiferimento degli asset o di altre entità informative (clienti, fornitori, sedi, ecc) tramite visualizzazione su mappe geografiche e/o planimetrie.

Il georiferimento sul territorio viene effettuato integrando servizi esterni quali OpenStreetMap, GoogleMaps, ecc, mentre la gestione delle planimetrie è stata implementata tramite utilizzo del sistema open source GeoServer.

Attraverso l'utilizzo di GeoServer è possibile aggiungere dei layer personalizzati (es. planimetrie) da poter utilizzare nel modulo GIS.

I formati supportati sono:

- Shape
- GeoTiff
- WorldImage.

Installazione di Geoserver

Per eseguire l'intallazione di Geoserver è necessario:

- scaricare l'applicazione dal sito ufficiale (http://geoserver.org/)
- eseguire il deploy del file war all'interno di Tomcat
- eseguire il login utilizzando lo username **admin** e la password **geoserver**
- eliminare tutti i workspace preinstallati
- crearne uno chiamato, per esempio, “cmdbuild”

Configurazione di CMDBuild

Una volta selezionato il Modulo di Amministrazione di CMDBuild, si dovrà accedere al pagina GIS del menu di Configurazione ed abilitare il modulo GIS.

Quindi, accedendo al menu GIS di dovrà intervenire sulle seguenti voci:

- **External Services**
  - abilitare GeoServer
  - specificare i parametri relativi all'installazione
  - specificare il nome del workspace creato in precedenza
  - specificare le credenziali di amministrazione
- **Geoserver layers**
  - aggiungere i layer necessari, che verranno quindi memorizzati all'interno di Geoserver
- **Layers order**
  - specificare l'ordine dei layer così come dovranno essere presentati nel Modulo di Gestione

BIMServer

Generalità

CMDBuild include la possibilità di gestire il georiferimento degli asset tramite visualizzazione con funzionalità BIM (Building Information Modeling).

Il georiferimento su modelli BIM (esportati nello standard IFC) viene effettuato integrando il servizio esterno open source BIMServer, che deve essere installato separatamente.

Connettore BIM IFC

CMDBuild integra un visualizzatore di file IFC (ISO 16739:2013) e un connettore per l'importazione automatica dei dati dai file IFC.

Il modulo BIM di CMDBuild è basato su uno scambio bidirezionale di dati tra CMDBuild e l'applicazione “BimServer” (Open source Building Information Modelserver).

Per configurare le funzionalità BIM di CMDBuild procedere come segue:

- attivare un'installazione funzionante di BimServer
- configurare la connessione tra CMDBuild e BimServer (Amministrazione → Configurazione → BIM)
- creare un progetto e caricare un file IFC dalla pagina Progetti (Amministrazione → BIM → Progetti)
- definire la classe alla quale legare I file IFC (e.g. Building) dalla pagina Livelli (Amministrazione → BIM → Livelli) impostando Root = true
- impostare il collegamento tra il progetto e la scheda della classe configurata come Root al punto precedente

A questo punto sarà possibile aprire il visualizzatore dal Modulo gestione dati.

Per eseguire un'importazione automatica delle entità definite nel file IFC procedere con i seguenti passi aggiuntivi:

- impostare Attivo = true le classi nelle quali si vuole importare dei dati dalla pagina Livelli
- configurare i criteri di mappatura tra le entità IFC e le classi CMDBuild inserendo un blocco xml nella colonna ImportMapping della tabella _BimProject del database di CMDBuild; un esempio del formato supportato da questo campo è il seguente:

```
<bim-conf>
        <entity name="IfcBuilding" label="Building">
                <attributes>
                        <attribute type="simple" name="Name" label="Code" />
                        <attribute type="simple" name="Description" label="Name" />
                </attributes>
        </entity>
</bim-conf>
```

dove

- “IfcBuilding” è il tipo ifc sorgente, “Building” è la classe CMDBuild di destinazione
- “Name” è l'attributo ifc sorgente, “Code” è l'attributo CMDBuild di destinazione
    - “simple” è il tipo dell'attributo IFC previsto dallo schema di definizione del linguaggio IFC
- cliccare sul pulsante “Importa IFC” nella pagina dei Progetti per avviare l'importazione

Configurazione di CMDBuild in modalità Cluster

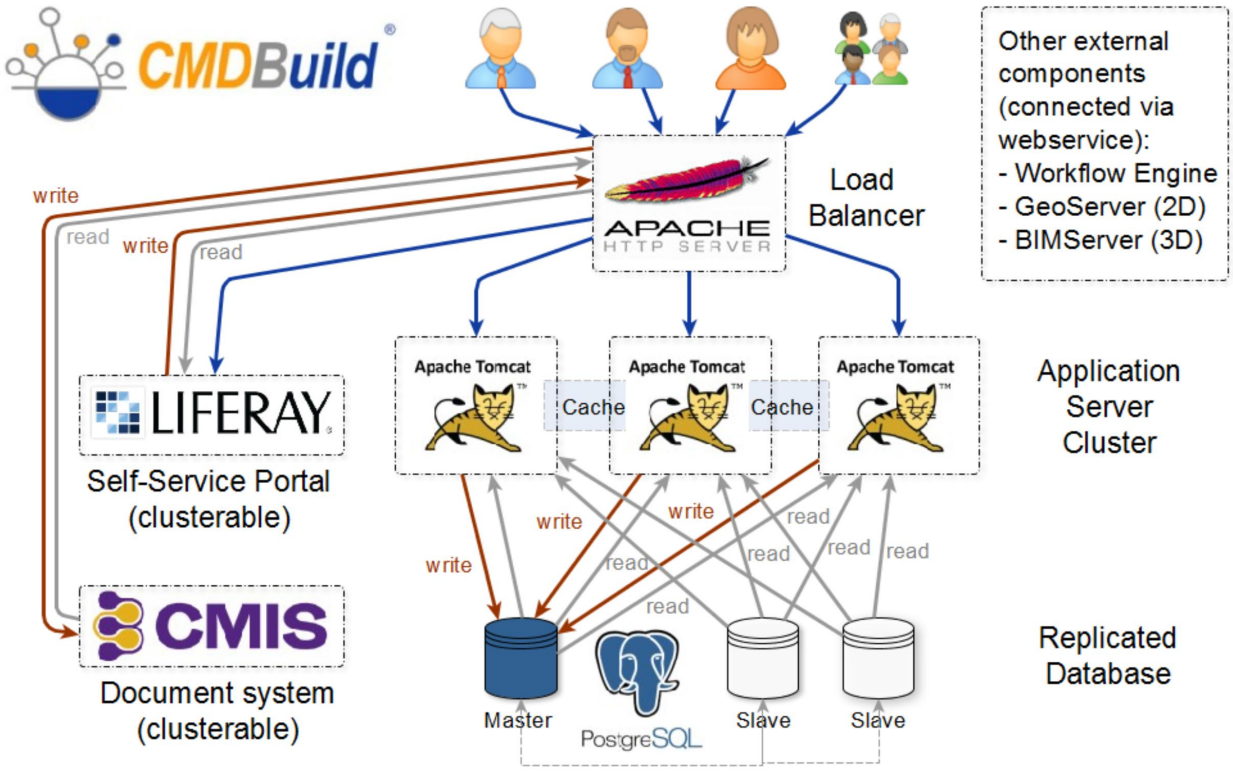
Generalità

CMDBuild consente la configurazione di una architettura in modalità Cluster, in cui viene attivato a monte un webserver bilanciatore che suddivide le richieste a CMDBuild su diverse istanze dell'applicazione (nodi), collegati ad uno stesso database PostgreSQL eventualmente a sua volta configurato in modalità di replicazione Master-Slave.

La clusterizzazione del servizio permette di raggiungere due obbiettivi:

- High Availability: se una istanza di CMDBuild smette di funzionare, altre istanze possono comunque mantenere attivo il servizio
- Scalabilità: la disponibilità di più istanze (nodi) del servizio consente di suddividere il carico e quindi di rispondere più velocemente alle richieste.

Segue uno schema dell'architettura CMDBuild in configurazione Cluster:



Nell'esempio riportato sopra il bilanciatore è costituito dal webserver Apache.

Sia il bilanciatore che il database PostgreSQL possono essere a loro volta ridondati con diverse tecniche, che esulano dal contenuto di questo capitolo.

Configurazione delle Istanze di Tomcat

In questo manuale simuleremo la creazione di una architettura CMDBuild in modalità Cluster costituita da quattro server :

- Server con database PostgreSQL master
- Server con istanza CMDBuild 1: Nodo 1 con indirizzo 10.0.0.100:8080
- Server con istanza CMDBuild 2: Nodo 2 con indirizzo 10.0.0.101:8080
- Server con Apache (Load Balancer)

N.B.: Nel caso di utilizzo del motore di workflow Shark, si consiglia l'installazione su una istanza di Tomcat dedicata (su server diverso o su uno dei server componenti il cluster)

Nel seguito identificheremo con *${cmdbuild_home}* il path della webapp *cmdbuild* all'interno dei server.

La cartella contenente i file di configurazione *${cmdbuild_home}/WEB-INF/conf* deve essere sincronizzata tra le varie Istanze di Tomcat: nel caso le istanze di Tomcat risiedano su più server, è consigliabile montare sulle varie istanze la cartella ***conf*** tramite NFS, in modo da non dover gestire l'allineamento di più file di configurazione.

Lo stesso discorso vale per la cartella *${cmdbuild_home}/**upload***, directory di servizio dove vengono salvate le icone delle classi.

Ipotizziamo di aver già installato e configurato CMDBuild sui due server Nodo1 e Nodo2 (vedi manuale di installazione).

A questo punto è necessario fare in modo che le due istanze di CMDBuild possano comunicare tra loro (nel caso non si utilizzi una unica cartella *conf*, le seguenti operazioni vanno effettuate su tutte le istanze componenti il cluster).

Step 1

Modificare il file:

```
${cmdbuild_home}/WEB-INF/conf/cmdbuild.conf
```

impostando la variabile org.cmdbuild.clustered a true:

```
org.cmdbuild.clustered=true
```

Step 2

A questo punto occorre configurare il componente JGroups che si occuperà di mantenere le comunicazioni tra le varie istanze (sincronizzazione).

Per fare questo occorre modificare il file *jgroups.xml*

```
${cmdbuild_home}/WEB-INF/conf/jgroups.xml
```

Una delle caratteristiche più potenti di JGroups è il suo stack di protocolli flessibile: componendo ed abbinando i vari protocolli, possono essere soddisfatti vari requisiti (discovering, sicurezza, compressione , controllo di flusso, ecc).

La configurazione della comunicazione tra le istanze descritta in questo manuale si basa su uno stack di protocolli standard, e come protocollo di trasporto utilizza TCP.

Si possono configurare altri tipi di stack di protocolli, non descritti in questo manuale.

La configurazione descritta nel seguito permette il discovery tra istanze raggiungibili tramite indirizzo IP o nome di dominio, e dovrebbe coprire la maggior parte delle tipologie di installazione.

Innanzitutto occorre configurare la porta su cui comunica JGroups (default 7800), modificando il parametro *bind_port.*

Poi vanno indicate nella variabile *initial_hosts* tutte le potenziali istanze conosciute di Tomcat che compongono il cluster, con la porta su cui JGroups risponde:

Nel nostro caso, ad esempio, scriveremo:

```
initial_hosts="${jgroups.tcpping.initial_hosts:10.0.0.100[7800],10.0.0.101[7800]}"
```

Segue un esempio del file jgroups.xml della nostra configurazione:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="urn:org:jgroups" xsi:schemaLocation="urn:org:jgroups
http://www.jgroups.org/schema/jgroups.xsd">
        <TCP bind_port="7800" bind_addr="NON_LOOPBACK"/>
        <PDC cache_dir="jgroups.pdc.cache"/>

        <TCPPING async_discovery="true"
                initial_hosts="${jgroups.tcpping.initial_hosts:10.0.0.100[7800],10.0.0.101[7800]}"
                port_range="2"
                use_disk_cache="true"
                return_entire_cache="true"
        />

        <MERGE3 min_interval="10000" max_interval="30000" />
        <FD_SOCK />
        <FD timeout="3000" max_tries="3" />
        <VERIFY_SUSPECT timeout="1500" />
        <pbcast.NAKACK2 use_mcast_xmit="false"
                discard_delivered_msgs="true" />
        <UNICAST3 />
        <pbcast.STABLE stability_delay="1000" desired_avg_gossip="50000"
                max_bytes="400000" />
        <pbcast.GMS print_local_addr="true" join_timeout="2000" />

</config>
```

Step 3

E' consigliato fare in modo che Java utilizzi di default lo stack IPv4, aggiungendo nel file delle proprietà di Java del Tomcat di CMDBuild (setenv.sh o catalina.sh) l'istruzione

```
-Djava.net.preferIPv4Stack=true
```

Configurazione del bilanciatore su Apache

Devono essere attivi i seguenti moduli di Apache:

- proxy
- proxy_http
- proxy_balancer
- lbmethod_byrequests
- headers

E’ necessario ora modificare il file di configurazione del modulo di bilanciamento, che per la distribuzione Linux Ubuntu è ad esempio :

```
mods-available/proxy.conf
```

Segue un esempi del file *proxy.conf*:

```
<IfModule mod_proxy.c>
       ProxyRequests Off
       ProxyPreserveHost On

       # cookie per session affinity
       Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
       env=BALANCER_ROUTE_CHANGED

       ServerName load-balancer

       #Allow everyone to access any proxied content
       <Proxy *>
              AddDefaultCharset off
              Order deny,allow
              Allow from all
       </Proxy>

</IfModule>
```

E' poi necessario configurare il VirtualHost desiderato, attivandolo nel sites-enabled.

Ad esempio, configuriamo il dominio:

cmdbuildcluster.example.com/cmdbuild

in modo che le richieste a CMDBuild vengano ripartite tra le due istanze di Tomcat Nodo1 e Nodo2:

```
<VirtualHost *:80>
       ServerName cmdbuildcluster.example.com

       <Proxy balancer://ms01 >
              BalancerMember http://10.0.0.100:8080 loadfactor=50 route=war1
              BalancerMember http://10.0.0.101:8080 loadfactor=50 route=war2
              ProxySet lbmethod=byrequests
              ProxySet stickysession=ROUTEID
       </Proxy>

       ProxyPass /cmdbuild balancer://ms01/cmdbuild
       ProxyPassReverse /cmdbuild balancer://ms01/cmdbuild

</VirtualHost>
```

Ora si può accedere a CMDBuild clusterizzato tramite l'URL:

http://cmdbuildcluster.example.com/cmdbuild

Verifica funzionamento Cluster

Per verificare che il clustering funzioni è possibile controllare il contenuto della tabella:

quarz_scheduler_state

che deve essere presente nello schema *quarz* nel database di CMDBuild.

Se tutto funziona correttamente per ogni nodo Tomcat configurato (nel nostro caso Nodo1 e Nodo2) comparirà un record nella tabella, ed il valore della colonna *last_checkin_time* deve aggiornarsi ogni pochi secondi.

Applicazione patch in caso di aggiornamento

Nel caso si lavori con più istanze di CMDBuild configurate in modalità Cluster, si debba aggiornare la “webapp” Tomcat di CMDBuild, e questa contenga patch al database, deve essere utilizzata la seguente procedura:

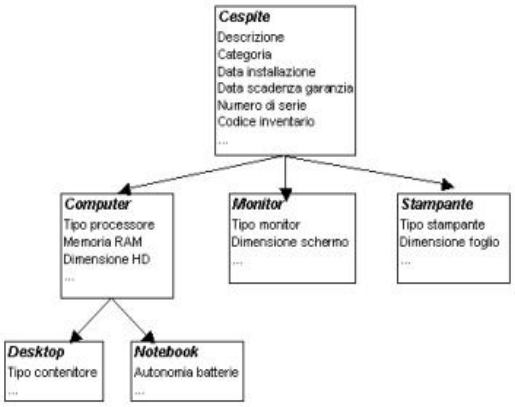
- dopo l'aggiornamento della “webapp” Tomcat su tutte le istanze, avviare una sola istanza
- accedere tramite browser all'istanza avviata, attendere che il sistema proponga di applicare le patch, e confermare
- a quel punto è possibile avviare tutte le altre istanze del cluster

Particolarità nel disegno del Database

Criteri di progettazione

La progettazione del database ha dovuto rispondere ad un primo insieme di requisiti di base:

- gestire la strutturazione gerarchica a più livelli delle classi (superclassi / sottoclassi), al fine di poter specializzare una classe mantenendo nelle superclassi gli attributi generali
- gestire relazioni "molti a molti" fra le classi
- tracciare la storia completa delle modifiche dei dati e delle relazioni nel tempo



Per tutte e tre le esigenze è stato individuato come particolarmente interessante il meccanismo di "derivazione" fra classi reso disponibile dal database object-relational open source PostgreSQL.

Si è quindi deciso di utilizzare PostgreSQL come database di supporto a CMDBuild, ottenendo in tal modo di disegnare in modo estremamente naturale le strutture sopra descritte.

Ereditarietà

1) Strutturazione a più livelli gerarchici

Tramite la parola chiave "**inherits**" è possibile infatti in PostgreSQL creare una tabella che ne "specializza" un'altra, aggiungendo alcuni attributi specifici e ritrovandosi tutti gli attributi definiti nella superclasse.

Esempio:

```
CREATE TABLE "Asset"
(
    "Id" integer NOT NULL DEFAULT nextval('Asset_SEQ'::text),
    "Code" varchar(100),
    "Description" varchar(250),
    "SerialNo" varchar(40),
    "VersionNo" varchar(32),
    "InstallationDate" timestamp,
    "WarrantyExpireDate" timestamp,
    "State" varchar(16),
    "StateDate" timestamp,
    CONSTRAINT asset_pkey PRIMARY KEY ("Id")
)
CREATE TABLE "Monitor"
(
    "MonitorType" varchar,
    "ScreenSize" varchar(16),
    "MaxScreenRes" varchar(16)
```

) **inherits** ("Asset")

2) Relazioni "molti a molti"

Le diverse tipologie di relazioni fra classi vengono implementate ciascuna con una apposita tabella di relazioni "molti a molti", creata tramite derivazione da una superclasse predefinita allo scopo di semplificare la creazione delle sottoclassi e di garantirne l'omogeneità strutturale.

Esempio:

```
CREATE TABLE "Map"
(
    "Id" integer NOT NULL DEFAULT nextval('Map_SEQ'::text),
    "IdDomain" regclass,
    "IdClass1" regclass,
    "IdObj1" integer NOT NULL,
    "IdClass2" regclass,
    "IdObj2" integer NOT NULL,
    "Status" character(1),
    "User" character varying(40),
    "BeginDate" timestamp without time zone NOT NULL DEFAULT now(),
    "EndDate" timestamp without time zone,
    CONSTRAINT map_pkey PRIMARY KEY ("Id")
)
CREATE TABLE " Map_aggregazione"
(
) inherits ("Map")
```

3) Storia delle modifiche

Anche per la gestione della storia delle modifiche viene sfruttato il meccanismo di derivazione delle classi in PostgreSQL, creando una classe derivata per ogni tipologia di oggetto, nella quale tramite appositi trigger del database viene archiviato il record corrente prima di modificarne gli attributi, associandogli l'"Id" del record base, la data di modifica ed il login dell'operatore che ha effettuato la modifica.

Esempio:

```
CREATE TABLE "Monitor"
(
    "MonitorType" varchar,
    "ScreenSize" varchar(16),
    "MaxScreenRes" varchar(16)
) inherits ("Asset")
CREATE TABLE "Monitor_history"
(
    "CurrentId" integer NOT NULL,
    "EndDate" timestamp NOT NULL DEFAULT now()
) inherits ("Monitor")
```

Tramite lo stesso meccanismo viene gestita la storia delle modifiche nelle relazioni, creando cioè una classe derivata per ogni tipologia di relazione, nella quale tramite appositi trigger di database vengono archiviati i record di relazione correnti prima di eliminarli, associandogli l'"Id" del record base, la data di modifica ed il login dell'operatore che ha effettuato la modifica.

Superclassi primitive: "Class" e "Map"

Sulla base della filosofia sopra descritta sono state definite due superclassi di sistema denominate rispettivamente:

- "Class", che rappresenta la generica classe dati da cui derivare le schede specifiche dei diversi modelli dati
- "Map", che rappresenta la generica tabella di relazioni fra coppie di classi

Il significato degli attributi delle due superclassi è descritto alla tabella seguente:

Superclasse "Class"		
Nome	**Tipo SQL**	**Descrizione**
Id	integer	chiave primaria della tabella
IdClass	regclass	OID della tabella
BeginDate	timestamp	data di inserimento del record
User	varchar(40)	utente che ha inserito il record
Status	character(1)	stato logico del record (A = attivo, N = cancellato, U = aggiornato)
Code	varchar(100)	campo dati: può essere utilizzato per archiviare il codice identificativo di un record (ad esempio il serialnumber di un computer) il campo "Code" può essere rinominato (ad esempio "PartitaIVA" in una classe contenente dei clienti o fornitori) oppure può essere disattivato.
Description	varchar(250)	campo dati: rappresenta la descrizione del record in CMDBuild assume un significato particolarmente importante essendo il valore mostrato nelle liste di selezione per la valorizzazione degli attributi di tipo "reference" (foreign key su altre classi)
Notes	text	note libere

Superclasse "Map"		
Nome	**Tipo SQL**	**Descrizione**
IdDomain	regclass	OID della tabella
IdClass1	regclass	OID della prima classe in relazione
idObj1	integer	indice dell'oggetto in relazione per la prima classe
IdClass2	regclass	OID della seconda classe in relazione
idObj2	integer	indice dell'oggetto in relazione per la seconda classe

User	varchar(40)	utente che ha inserito la relazione
Status	character(1)	stato logico del record (A = attivo, N = cancellato, U = aggiornato)
BeginDate	timestamp	data di inserimento della relazione
EndDate	timestamp	data di modifica della relazione

Tutte le tabelle definite nell'ambito dell'applicazione CMDBuild ereditano dalla tabella "Class", aggiungendo campi specifici descrittivi delle informazioni che andranno a rappresentare.

Le tabelle derivate possono essere "Super Classi" o "Classi" normali. Solo le ultime contengono dati, le prime vengono utilizzate come un raggruppamento logico di classi utile per definire attributi comuni a più classi e domini in modo più generale.

Nella base dati ogni classe normale è accompagnata da una ulteriore tabella, dedicata alla storicizzazione dei dati modificati, il cui nome è lo stesso ma completato con il suffisso "_history". Le tabelle "storiche" ereditano dalle tabelle condividono tutti i campi con le tabelle da cui derivano, completate solo con il link alla scheda dati attiva e con la data di fine validità.

Ogni volta che un record della tabella principale subisce una modifica o una cancellazione il record originale viene spostato nella tabella storica. Durante tale operazione vengono aggiornati il valore delle colonne "Status" (secondo la tabella riportata sopra) e "EndDate" (con il timestamp dell'operazione).

Allo stesso modo tutti domini tra Classi o Super Classi ereditano dalla tabella "Map" e ogni dominio ereditato avrà la corrispondente tabella per il versioning (con lo stesso suffisso "_history").

Per convenzione tutti i Domini hanno il prefisso "Map_".

Metadati

Per la definizione del modello dati CMDBuild utilizza una serie di metadati, associati alle tabelle ed ai loro attributi, che estendono i metadati base gestiti da PostgreSQL.

Per la memorizzazione di tali metadati vengono utilizzati i commenti associabili a tabelle e colonne, archiviati nella tabella di sistema "pg_description".

La struttura dei metadati segue una sintassi rigida ed obbligata, strutturata secondo un sistema a chiave/valore che segue la sintassi definita dalla seguente regular expression:

```
(([A-Z0-9]+): ([#A-Za-z0&-9éèìùòà_\-\:\s]*)|+)*
```

Ad esempio un commento può essere:

```
MODE: read|DESCR: some text
```

I commenti validi relativi ad una classe nel suo complesso sono:

Metadati per definizione classi			
Chiave	**Significato**	**Valori possibili**	**Note**
MODE	modalità di accesso	reserved|read|write	obbligatorio
TYPE	tipo tabella	class|domain	obbligatorio
SUPERCLASS	indica se la tabella è una superclasse	true|false	default = false
DESCR	descrizione mostrata	qualsiasi valore	obbligatorio

	all'utente		
STATUS	stato di utilizzo della tabella	active|not active	obbligatorio

Tutte le altre chiavi vengono ignorate dal sistema di interpretazione dei metadati.

I commenti validi relativi ad un attributo sono:

Metadati per definizione attributi			
Chiave	**Significato**	**Valori possibili**	**Note**
MODE	modalità di accesso	reserved|read|write	obbligatorio
DESCR	descrizione mostrata all'utente sulla form di gestione (label)	qualsiasi valore	obbligatorio
INDEX	ordine di visualizzazione dell'attributo sulla form di gestione	un numero	posizione fisica nel DB, se assente
BASEDSP	indica se l'attributo va mostrato nella visualizzazione "a griglia"	true|false	default = false
GROUP	"pagina" di visualizzazione dell'attributo (nel caso di paginazione)	una stringa contenente la label attribuita alla pagina	
FIELMODE	Modalità di visualizzazione del campo	write|hidden|read	
REFERENCEDOM	dominio utilizzato per valorizzare un campo reference	un dominio valido per la classe	non obbligatorio
REFERENCEDIRECT	direzione della relazione rispetto al dominio corrispondente	true|false	obbligatorio solo se valorizzato REFERENCEDOM
REFERENCETYPE	tipo del reference rispetto le operazioni di cancellazione	restrict	obbligatorio solo se valorizzato REFERENCEDOM
LOOKUP	lista lookup associata all'attributo	una lista lookup	non obbligatorio
STATUS	stato di utilizzo della tabella	active|not active	obbligatorio

Tutte le altre chiavi vengono ignorate dal sistema di interpretazione dei metadati.

Si raccomanda agli utilizzatori di non intervenire manualmente su tali metadati, al fine di non provocare malfunzionamenti anche gravi nei meccanismi di CMDBuild e nella conseguente consistenza dei dati memorizzati nel sistema.

APPENDICE: Glossario

ALLEGATO

Per “allegato” si intende un qualunque file associabile ad una scheda dati inserita nel sistema.

Per la gestione degli allegati CMDBuild utilizza in modalità embedded un qualunque sistema documentale compatibile con il protocollo standard CMIS (oppure il DMS Alfresco fino alla versione 3 tramite il proprio webservice nativo).

La gestione degli allegati supporta il versioning di file caricati più volte, con numerazione automatica.

ATTIVITA'

Per “attività” si intende uno dei passaggi che costituiscono il processo.

Una attività è caratterizzata da un nome, un esecutore, un tipo, eventuali attributi, eventuali metodi associati ad API di CMDBuild per poter essere eseguiti.

Per “istanza di attività” si intende una specifica attivazione di una attività, effettuata automaticamente dal sistema o manualmente da un operatore.

Vedi anche: Processo

ATTRIBUTO

Il termine indica nel sistema CMDBuild la generica tipologia di informazione descrittiva di una determinata classe.

CMDBuild consente tramite il Modulo Schema di creare nuovi attributi in una classe o in un dominio e di modificarne alcune caratteristiche.

Nella classe “Fornitore” gli attributi sono ad esempio il nome, l'indirizzo, il numero di telefono, ecc.

Ogni attributo corrisponde nel Modulo di Gestione a campi di inserimento dati sulla apposita scheda di gestione della classe e a colonne della corrispondente tabella nel database.

Vedi anche: Classe, Dominio, Relazione, Superclasse, Tipo di attributo

BIM

Metodologia che si pone l'obiettivo di supportare l’intero ciclo di vita di un edificio, dall'idea iniziale alla fase di costruzione, di utilizzo e manutenzione, fino alla eventuale demolizione finale.

La metodologia BIM (Building Information Modeling) è supportata da numerosi programmi informatici che possono interagire tramite un formato aperto di scambio dati denominato IFC (Industry Foundation Classes).

Vedi anche: GIS

CI

Si definisce Configuration Item (Elemento della Configurazione) ogni elemento che concorre a fornire il servizio IT all’Utente, considerato ad un livello di dettaglio sufficiente per la sua gestione tecnica e patrimoniale.

Esempi di CI sono: server, workstation, programma applicativo, sistema operativo, stampante, ecc

Vedi anche: Configurazione

CLASSE

Il termine rappresenta un tipo di dati complesso caratterizzato da un insieme di attributi che nel loro insieme descrivono quel tipo di dato.

Una classe modella una tipologia di oggetto da gestire nel CMDB, quale ad esempio un computer, una applicazione software, un servizio, un fornitore, ecc

CMDBuild consente all'Amministratore del Sistema, attraverso il Modulo Schema, di definire nuove classi e di cancellare o modificare la struttura di classi già definite.

Una classe è rappresentata a video da una apposita scheda di gestione dati e nel database da una tavola generata automaticamente al momento della definizione della classe.

Vedi anche: Scheda, Attributo

CONFIGURAZIONE

Il processo di Gestione della Configurazione ha lo scopo di mantenere aggiornata e disponibile per gli altri processi la base di informazioni relativa agli oggetti informatici gestiti (CI), alle loro relazioni ed alla loro storia.

E' uno dei principali processi gestiti dal sistema ITIL.

Vedi anche: CI, ITIL

DASHBOARD

Una dashboard corrisponde in CMDBuild ad una raccolta di grafici di diversa tipologia, tramite cui avere immediata evidenza di alcuni parametri chiave (KPI) relativi ad un particolare aspetto di gestione del servizio IT.

Vedi anche: Report

DATABASE

Il termine indica un insieme di informazioni strutturato ed organizzato in archivi residenti sull'elaboratore server, nonché l'insieme dei programmi di utilità dedicati alla gestione dei tali informazioni per attività quali inizializzazione, allocazione degli spazi, ottimizzazione, backup, ecc.

CMDBuild si appoggia sul database PostgreSQL, il più potente, affidabile e completo database Open Source, di cui utilizza in particolare le sofisticate funzionalità e caratteristiche object oriented.

DOMINIO

Un dominio rappresenta una tipologia di relazione fra una coppia di classi.

E' caratterizzato da un nome, dalle descrizioni della funzione diretta ed inversa, dai codici delle due classi e dalla cardinalità (numerosità degli elementi relazionabili) ammessa, nonché dagli eventuali attributi configurati.

CMDBuild consente all'Amministratore del Sistema, attraverso il Modulo Schema, di definire nuovi domini e di cancellare o modificare la struttura di domini già definiti.

E' possibile caratterizzare ciascun dominio tramite definizione di attributi custom.

Vedi anche: Classe, Relazione

FILTRO DATI

Un filtro dati è una restrizione della lista degli elementi contenuti in una classe, ottenuta specificando condizioni booleane (uguale, diverso, contiene, inizia, ecc) sui possibili valori assumibili da ciascun attributo della classe.

I filtri dati possono essere definiti ed utilizzati “una tantum”, oppure possono essere memorizzati dall'operatore e richiamati successivamente (dallo stesso operatore o da operatori di altri gruppi di utenti ai quali l'Amministratore del sistema abbia concesso l'utilizzo).

Vedi anche: Classe, Vista

GIS

Un sistema GIS è un sistema informatico in grado di produrre, gestire e analizzare dati spaziali associando a ciascun elemento geografico una o più descrizioni alfanumeriche.

Le funzionalità GIS implementate in CMDBuild consentono di creare attributi geometrici, in aggiunta a quelli testuali, tramite cui rappresentare su scala locale (planimetrie) o su scala più estesa (mappe esterne) elementi puntuali (ad esempio gli asset IT), poligonali (ad esempio linee dati) o aree (piani, stanze, ecc).

Vedi anche: BIM

GUI FRAMEWORK

E' una interfaccia utente completamente personalizzabile e orientata a fornire un accesso semplificato all'applicazione, pubblicabile su portali web di qualsiasi tecnologia ed interoperabile con CMDBuild tramite il webservice REST standard.

Vedi anche: Mobile, Webservice

ITIL

Sistema di "best practice" ormai affermatosi come "standard de facto", non proprietario, per la gestione dei servizi informatici secondo criteri orientati ai processi (Information Technology Infrastructure Library).

Fra i processi fondamentali coperti da ITIL ci sono quelli del Service Support, comprendenti l'Incident Management, il Problem Management, il Change Management, il Configuration Management ed il Release Management.

Per ogni processo considera la descrizione, i componenti di base, i criteri e gli strumenti consigliati per la misura della qualità del servizio, i ruoli e le responsabilità delle risorse coinvolte, i punti di integrazione con gli altri processi (per eliminare duplicazioni e inefficienze).

Vedi anche: Configurazione

LOOKUP

Con il termine “LookUp” si indica una coppia di valori del tipo (Codice, Descrizione) impostabili dall'Amministratore del Sistema tramite il Modulo Schema.

Tali valori vengono utilizzati dall'applicazione per vincolare la scelta dell'utente, al momento della compilazione del relativo campo sulla scheda dati, ad uno dei valori preimpostati.

Il Modulo Schema consente la definizione di nuove tabelle di “LookUp” secondo le necessità dell'organizzazione.

MOBILE

E' una interfaccia utente ottimizzata per strumenti “mobile” (smartphone e tablet), implementata come “app” multipiattaforma (iOS, Android) ed interoperabile con CMDBuild tramite il webservice REST standard.

Vedi anche: GUI Framework, Webservice

PROCESSO

Per “processo” (o workflow) si intende una sequenza di passaggi (“attività”) descritti nel sistema per svolgere in forma guidata e secondo regole prestabilite una determinata azione.

Per ogni processo saranno avviate in CMDBuild una serie di “istanze di processo”, una per ogni necessità di effettiva esecuzione dell'azione corrispondente, che avrà luogo su “asset” specifici e sarà svolta da utenti specifici.

Una ”istanza di processo” viene attivata tramite avvio e conferma del primo passaggio previsto e termina alla esecuzione dell'attività finale prevista nella definizione.

Vedi anche: Attività

RELAZIONE

Per “Relazione” si intende in CMDBuild un collegamento effettivo di due schede appartenenti a due classi, o in altri termini una istanza di un dato dominio.

Una relazione è quindi definita da una coppia di identificativi univoci delle due schede collegate e dall'identificativo del dominio utilizzato per il collegamento, nonché dalla valorizzazione degli eventuali attributi previsti nel dominio.

CMDBuild consente agli operatori del Sistema, attraverso il Modulo Gestione Dati, di definire nuove relazioni fra le schede archiviate nel database.

Vedi anche: Classe, Dominio

REPORT

Il termine indica in CMDBuild una stampa (in formato PDF o CSV) riportante in forma analitica le informazioni estratte da una o più classi fra le quali sia definita una catena di domini.

I report possono essere generati e modificati dagli operatori di CMDBuild tramite una apposita funzione del Modulo di Gestione Dati e la relativa definizione viene memorizzata nel database per poter essere riutilizzata successivamente.

Vedi anche: Classe, Dominio, Database

SCHEDA

Con il termine “Scheda” in CMDBuild si riferisce un elemento archiviato in una determinata classe.

Una scheda è caratterizzata da un insieme di valori assunti da ciascuno degli attributi definiti per la sua classe di appartenenza.

CMDBuild consente agli operatori del Sistema, attraverso il Modulo Gestione Dati, di archiviare nuove schede nel database e di aggiornare schede già archiviate.

Le informazioni di ogni scheda saranno memorizzate nel database alle opportune colonne di una riga della tavola generata per la classe di appartenenza della scheda.

Vedi anche: Classe, Attributo

SUPERCLASSE

Una superclasse è una classe astratta utilizzabile per definire una sola volta attributi condivisi fra più classi. Da tale classe astratta è poi possibile derivare classi reali che conterranno i dati effettivi e che comprenderanno sia gli attributi condivisi (specificati nella superclasse) che quelli specifici della sottoclasse.

Ad esempio è possibile definire la superclasse “Computer” con alcuni attributi base (RAM, HD, ecc) e le sottoclassi derivate “Desktop”, “Notebook”, “Server”, ciascuna delle quali con i soli attributi specifici.

Vedi anche: Classe, Attributo

TIPO DI ATTRIBUTO

Ogni attributo definito per una determinata classe è caratterizzato da un “Tipo” che determina le caratteristiche delle informazioni contenute e la loro modalità di gestione.

Il tipo di attributo viene definito con il Modulo Schema e può essere poi modificato entro alcuni limiti dipendenti dalla tipologia dei dati già archiviati.

CMDBuild gestisce i seguenti tipi di attributo: “Boolean” (booleano, Si / No), “Date” (data), “Decimal” (decimale), “Double” (virgola mobile in doppia precisione), “Inet” (indirizzo IP), “Integer” (numero intero), “LookUp” (tabellato da lista configurabile in “Impostazioni” / “LookUp”), “Reference” (riferimento o foreign key), “String” (stringa), “Text” (testo lungo), “TimeStamp” (data e ora).

Vedi anche: Attributo

VISTA

Una vista è un insieme di schede definito in modo “logico” anziché dal fatto di costituire l'intero contenuto di una classe nel CMDB.

In particolare una vista può essere definita in CMDBuild applicando un filtro ad una classe (quindi conterrà un insieme ridotto delle stesse righe) oppure specificando una funzione SQL che estragga attributi da una o più classi correlate.

La prima tipologia di vista mantiene tutte le funzionalità disponibili per una classe, la seconda consente la sola visualizzazione e ricerca con filtro veloce.

Vedi anche: Classe, Filtro

WEBSERVICE

Un webservice è un'interfaccia che descrive una collezione di operazioni, accessibili attraverso una rete mediante messaggistica XML.

Tramite un webservice una applicazione può rendere accessibili le proprie funzionalità ad altre applicazioni operanti attraverso il web.

CMDBuild dispone di un webservice SOAP e di un webservice REST.

WIDGET

Un widget è un componente grafico di una interfaccia utente di una applicazione software, che ha lo scopo di facilitare all'utente l'interazione con l'applicazione stessa.

CMDBuild prevede l'utilizzo di widget sotto forma di “pulsanti” posizionabili su schede dati o su schede di avanzamento di processi. I pulsanti aprono finestre di tipo “popup” tramite cui inserire se richiesto informazioni aggiuntive e visualizzare poi l'output della funzione richiamata.