Password di accesso al database criptata

Come rendere più sicura la password di connessione al database?

Le credenziali di accesso al database vengono salvate, di default, in chiaro all'interno di un file di properties (portal-ext.properties o portal-setup-wizard.properties a seconda dei casi). Se vogliamo rendere più sicuro questo meccanismo possiamo fare una semplice modifica al codice sorgente di Liferay per poter gestire anche valori criptati.

La classe che dobbiamo modificare è com.liferay.portal.dao.jdb.DataSourceFactoryImpl, in particolare il metodo initDataSource(Properties properties):

*DataSourceFactoryImpl.java

```
82
83              tomcatDataSource.close();
84          }
85      }
86
87      @Override
88      public DataSource initDataSource(Properties properties) throws Exception {
89          Properties defaultProperties = PropsUtil.getProperties(
90                  "jdbc.default.", true);
91
92          /**
93           * marcorosetti.it - codice aggiuntivo per gestire password criptate - INIZIO
94           */
95          Enumeration<String> propEnum = (Enumeration<String>)defaultProperties.propertyNames();
96
97          while(propEnum.hasMoreElements())
98          {
99              String key = propEnum.nextElement();
100
101             if(key.equalsIgnoreCase("password"))
102             {
103                 boolean isEncrypted = GetterUtil.getBoolean(defaultProperties.getProperty("encrypted.password"));
104
105                 if(isEncrypted)
106                 {
107                     String encryptedValue = defaultProperties.getProperty(key);
108
109                     //Qui implementare la specifica funzione di decodifica
110                     String decryptedValue = new String(Base64.decode(encryptedValue));
111                     properties.setProperty(key,decryptedValue);
112                 }
113             }
114         }
115
116         /**
117          *marcorosetti.it  - codice aggiuntivo per gestire password criptate - FINE
118          */
119
120         PropertiesUtil.merge(defaultProperties, properties);
121
122         properties = defaultProperties;
```

Vediamo le operazioni importanti del codice che abbiamo inserito

1. Scorriamo tutte le proprietà impostate fino a trovare la proprietà password
2. Se è stato impostato a TRUE il valore di jdbc.default.encrypted.password allora consideriamo il valore della password come criptato e lo decriptiamo prima di sovrascriverlo nelle properties

Una volta modificato il codice dobbiamo rigenerare le librerie di liferay utilzzando i task ant messi a disposizione insieme ai sorgenti: possiamo utilizzare compile e poi andare a sostituire la classe generata all'interno del file **portal-impl.jar** nella cartella **tomcat-7.0.42\webapps\ROOT\WEB-INF\lib** oppure rigenerare l'intero file con il task jar.

Alcune considerazioni

- Nell'esempio riportato l'algoritmo di cript-decrypt è un semplice Base64 hashing che serve solo come "placeholder". Possiamo ovviamente complicare come vogliamo l'algoritmo a seconda delle nostre esigenze
- L'approccio applicato alla proprietà "password" si può applicare a qualsiasi proprietà si intenda criptare
- Se non facciamo altre modifiche ai sorgenti potrebbero venire stampati nei log dei messaggi a livello WARN del tipo "*Property encrypted.password is not a valid…*". Possiamo tranquillamente ignorare questi messaggi
- L'esempio vale per Liferay versione 6.2
- La sicurezza di questo approccio potrebbe essere aumentata andando ad offuscare il codice della classe
- Un approccio di sicurezza più architetturale consiste nell'utilizzare datasource esterni e iniettati tramite nome JNDI. In questo modo la gestione della sicurezza delle credenziali di accesso è demandata all'application server