GwResourceDeployer - Struttura Risorse

Versione 1.5.0

ReleaseNote

- **issue #2** il deployer Keycloak ora dispiega **tutti** i client il cui file **.json** è presente nella folder **gw_package/keycloak** (in precedenza veniva gestito un solo client, indicato tramite la property *keycloak.clientRepresentationFile*, ora rimossa). Vedi la sezione Folder keycloak qui sotto per i dettagli.

Altre versioni

qui.

Struttura folder & file

Le risorse devono essere preventivamente preparate secondo una struttura convenzionale, in maniera tale che possano essere successivamente correttamente dispiegate dall' eseguibile .

Dentro la folder principale, **gw_package**, avremo la seguente struttura:

[notazione colori: **folder** - file]

- **gw_package**
  - **activiti =>**
    - process_definition_1.bpmn (in numero variabile)
  - **database =>**
    - **sql**
      - R__metadata_schema.sql.
      - V1.0.0.0__metadadata_role.sql.
      - V1.0.0.1__metadadata_schema.sql.
      - V1.0.0.2__data_role.sql.
      - V1.0.0.3__data_schema.sql.
      - V1.0.0.4__data_schema_INSERT.sql.
      - V1.0.1.0__D_1st_fix.sql. (numero variabile, fix su 1 o + file, incrementando il 4th numero)
      - V1.1.0.0__D_1st_minor.sql. (numero variabile)
    - flywayCommands.groovy
  - **mapguide =>**
    - resource_package.mgp (in numero variabile)
    - webconfig.ini
  - **rabbitmq**
    - rabbit_conf.json (un solo file)
  - **keycloak =>**

        - client_1.json (in numero variabile, un file per ogni client Keycloak da dispiegare)
    - configuration.properties =>
    - remapConfiguration.properties =>

Questa è da intendersi come una struttura omnicomprensiva. Nel caso si debba utilizzare il deployer in solo alcune delle sue possibilità si invita a rimuovere le sezioni non utilizzate.

Un template di struttura può essere scaricato qui. I file contenuti sono da considerarsi come puramente di esempio e potrebbero dover essere cancellati o modificati.

Passiamo ad analizzare le varie sezioni in dettaglio.

Folder activiti

Qui possono eventualmente essere posizionate tutte le definizioni di processo. Le definizioni di processo vengono prodotte dentro Eclipse, utilizzando il plugin di Activiti, e devono necessariamente essere dei file con estensione **.bpmn**. Ad ogni lancio dell'eseguibile, viene valutato se sia necessario o meno dispiegare le definizioni qui presenti.

Il deploy è eseguito se:

- non c'è per niente un dispiegamento in Activiti della stessa definizione di processo
- L'ultima definizione di processo dispiegata ha md5 differente da quella della *folder activiti*

Quindi si possono lasciare le definizioni .bpmn contenute nella *folder activiti* anche dopo il primo lancio dell'eseguibili, essendo certi che esse non saranno ridispiegate fino al prossimo cambio della definizione stessa.

Se il progetto non utilizza il workflow, si può rimuovere totalmente questa folder.

Folder database

Qui sono presenti, sotto la folder **sql**, tutti gli script che verranno gestiti da Flyway Flyway è un tool open-source per la migrazione di database.

Convenzioni Nomenclatura

Per quanto riguarda l'uso in Geoweb, useremo le seguenti convenzioni di naming:

- **script repeatable** iniziano con la lettera maiuscola R, seguita da due tratti bassi __, seguita dal nome. Verranno eseguiti oltre che la prima volta, alla fine di ogni migrazione (solo se cambiati), e conterranno tipicamente gli ultimi metadati aggiornati di Geoweb, intesi come il contenuto delle tabelle dello schema dei metadati, cosi da avere una situazione sempre allineata. Sarà cura di chi crea tali script preventivamente eseguire nella giusta sequenza tutti gli SQL statement di delete per svuotare le tabelle, seguiti dagli SQL statement di insert. Esempio:

```
R__metadata_schema.sql
```

- **script versionati** iniziano con la lettera maiuscola V, seguita dalla versione composta da **4 numeri**, seguiti da due tratti bassi **__**, seguita da un nome esplicativo del contenuto. Conterranno gli script per gli aggiornamenti versionati del database. Il primo e secondo script saranno relativo all'user/role ed allo schema dei *metadati*. Esempio:

  ```
  V1.0.0.0__metadata_role.sql
  V1.0.0.1__metadata_schema.sql
  ```

  Il terzo, quarto e quinto saranno relativi all'user/role ed allo schema dei *dati*. Esempio:

  ```
  V1.0.0.2__data_role.sql
  V1.0.0.3__data_schema.sql
  V1.0.0.4__data_schema_INSERT.sql
  ```

```
V1.0.0.3__data_schema.sql
```

serve per creare le tabelle mentre

```
V1.0.0.4__data_schema_INSERT.sql
```

per l'inserimento dei dati.

I successivi saranno o fix relativi allo schema dei *dati* (Per brevità prendiamo la convenzione di usare l'abbreviativo 'D' per 'data_schema' e 'M' per 'metadata_schema'). Esempio:

```
V1.0.1.0__D_1st_fix_part1.sql
V1.0.1.1__D_1st_fix_2nd_block.sql
...
```

o minor change, sempre allo schema dei *dati*. Esempio:

```
V1.1.0.0__D_1st_minor.sql
```

Qui un esempio di un'ipotetica successione di interventi successivi, tutti allo schema dei dati, dove il *name* è esplicativo della *version*

```
V1.1.1.0__D_1st_fix_1st_block.sql
V1.1.1.1__D_1st_fix_2nd_block.sql
V1.2.0.0__D_2nd_minor.sql
V1.3.0.0__D_3rd_minor.sql
V1.3.1.0__D_1st_fix_1st_block.sql
V1.3.1.1__D_1st_fix_2nd_block.sql
V1.3.1.2__D_1st_fix_3rd_block.sql
V1.3.2.0__D_2nd_fix_1st_block.sql
V1.4.0.0__D_4th_minor.sql
V2.0.0.0__M_2th_major.sql
...
```

Una nota sulla versione, nel formato **M.m.f.pf**, composta da quattro cifre, ognuna con uno specifico utilizzo:

1. M: **major**
2. m: **minor**
3. f: **fix**
4. pf: **progressivo interno al fix** Un singolo fix potrebbe essere composto da più script ognuno che gestisce una parte differente

In generale, la documentazione sulle convenzioni di nomenclatura per gli script versionati la trovate nella sezione *SQL-based migrations* qui

Nota

Come procedura interna, al fine di evitare il più possibile problematiche, gli script che verranno rilasciati qui, dovranno essere stati preventivamente collaudati ed approvati. E ne dovrà essere stato rilasciato il tag di versione nel repository di commessa.

Convenzioni Contenuto

FlywayDB si aspetta per funzionare una serie di elementi minimi:

- un database già creato
- un suo *user/role* specifico per il login al database, tipicamente **flyway**, con:
    - permesso di login
    - superuser
        - permesso creazione ruoli
        - permesso creazioni schema
        - permesso gestire tabelle
- lo schema di default **metadata**, anche non creato, ma che **va SEMPRE configurato**: vedi le property *flyway.defaultSchema* / *flyway.schemas* nel configuration.properties qui sotto.

Nei primi script di relativi a *metadata* e *data* bisogna sempre inserire alcuni script di creazione schema (con notazione *IF NOT EXISTS*, od equivalenti), di creazione ruoli e di relativa associazione. Per un bug di *Flyway*, che non riesce a creare correttamente più di uno schema in autonomia (tramite la proprietà *flyway.schemas*), **se ne può impostare solo uno**, il quale verrà creato autonomamente. L'altro dovrà essere creato *manualmente*, tramite un'apposita istruzione sql nello script.

La politica aziendale prevede di dispiegare modularmente ed indipendentemente i vari moduli/soluzioni. Ogni modulo deve avere quindi il suo deployer, che si preoccupa di tenere allineati i dati/metadati specifici per la soluzione.

Come conseguenza di ciò si è deciso che ogni modulo/soluzione avrà il suo deployer, **il quale, in caso di coesistenza, condivide lo schema dei data, mentre ognuno ha il suo schema metadata separato**.

Dato che *FlywayDB* crea la sua tabella *flyway_schema_history* nello schema definito nella proprietà

del configuration.properties *flyway.defaultSchema*, essa va configurata con lo schema **metadata**, per evitare che lo storico dei *deployer* venga sovrapposto a quello di un altro, causando errori.

Tenendo conto dei requisiti:

1. si può istallare un modulo singolo (che in futuro potrà essere affiancato da altri)
2. si possono fin dall'inizio installare più moduli
3. 1) e 2) con o senza *modulo pagine di accesso*

Si prospettano più casistiche, ma c'è una configurazione generale che va bene per tutti i casi.

Regole Generali

Queste regole permettono la creazione di script in un unica maniera, fregandosene del fatto che il modulo associato al deployer, sia principale o secondario.

Regole Generali METADATA

Ecco gli SQL statement necessari per lo script **metadata**:

- l'istruzione di creazione dell'utente metadata con i corretti permessi
- l'istruzione di creazione schema
- l'istruzione di associazione dello schema metadata all'utente.

Esempio metadata **oracle**:

```
-------------------------------
--   ROLE CREATION
-------------------------------
CREATE ROLE cde_metadata WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'cde_metadata';

-------------------------------
--   SCHEMA CREATION
-------------------------------
CREATE SCHEMA cde_metadata ;

-------------------------------
--   SCHEMA-USER ASSOCIATION
-------------------------------
ALTER SCHEMA cde_metadata OWNER TO cde_metadata;
```

Esempio metadata **postgres**:

```
-------------------------------
--   ROLE CREATION
-------------------------------
CREATE ROLE cde_metadata WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'cde_metadata';
```

```
------------------------------
--   SCHEMA CREATION
------------------------------
CREATE SCHEMA IF NOT EXISTS cde_metadata;

------------------------------
--   SCHEMA-USER ASSOCIATION
------------------------------
ALTER SCHEMA cde_metadata OWNER TO cde_metadata;
```

Esempio metadata **sqlserver**:

```
------------------------------
--   ROLE CREATION
------------------------------
CREATE ROLE cde_metadata WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'cde_metadata';

------------------------------
--   SCHEMA CREATION
------------------------------
CREATE SCHEMA cde_metadata;

------------------------------
--   SCHEMA-USER ASSOCIATION
------------------------------
ALTER SCHEMA cde_metadata OWNER TO cde_metadata;
```

Regole Generali DATA

Ecco gli SQL statement necessari per lo script **data**:

- l'istruzione di creazione dell'utente data con i corretti permessi (in caso di unico ruolo di connessione ai dati condiviso fra più moduli, con CON NOTAZIONE 'IF NOT EXISTS')
- l'istruzione di creazione schema CON NOTAZIONE 'IF NOT EXISTS'
- l'istruzione di associazione dello schema data all'utente.

Esempio data **oracle**:

```
------------------------------
--   ROLE CREATION
------------------------------
CREATE ROLE cde_data WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'cde_data';
--use the procedure below in case of more deployer that creates role with
the same name (es: shared data_role name)
--TODO
```

```
------------------------------
--   SCHEMA CREATION
------------------------------
--TODO find the equivalent of
--CREATE SCHEMA IF NOT EXISTS cde_data;
CREATE SCHEMA cde_data;

------------------------------
--   SCHEMA-USER ASSOCIATION
------------------------------
ALTER SCHEMA cde_data  OWNER TO cde_data;
```

Esempio data **postgres**:

```
------------------------------
--   ROLE CREATION
------------------------------
--CREATE ROLE cde_data WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'cde_data';
--use the procedure below in case of more deployer that creates role with
the same name (es: shared data_role name)
do
$do$
BEGIN
   IF NOT EXISTS (
      SELECT FROM pg_catalog.pg_roles
      WHERE  rolname = 'cde_data') THEN
        CREATE ROLE cde_data WITH
        LOGIN
        NOSUPERUSER
        NOINHERIT
        NOCREATEDB
        NOCREATEROLE
        NOREPLICATION
        ENCRYPTED PASSWORD 'md5bd688ac7df3c2ad4d1270a86ddac9a39';
   END IF;
END
$do$;

------------------------------
--   SCHEMA CREATION
------------------------------
CREATE SCHEMA IF NOT EXISTS cde_data;

------------------------------
--   SCHEMA-USER ASSOCIATION
------------------------------
ALTER SCHEMA cde_data  OWNER TO cde_data;
```

Esempio metadata **sqlserver**:

```
------------------------------
--   ROLE CREATION
------------------------------
CREATE ROLE cde_metadata WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'cde_metadata';
--use the procedure below in case of more deployer that creates role with
the same name (es: shared data_role name)
--TODO


------------------------------
--   SCHEMA CREATION
------------------------------
IF NOT EXISTS ( SELECT  * FROM    sys.schemas  WHERE   name = N'cde_data' )

    EXEC('CREATE SCHEMA [cde_data]');
GO

------------------------------
--   SCHEMA-USER ASSOCIATION
------------------------------
ALTER SCHEMA cde_metadata OWNER TO cde_data;
```

Note Generali

- Per *FlywayDB* si è scelto di configurare lo schema dei *metadata*, e non *data*, perchè esso crea la sua tabella *flyway_schema_history* nello schema definito nella proprietà del configuration.properties *flyway.defaultSchema*, per evitare che lo storico dei *deployer* venga sovrapposto a quello di un altro, causando errori.
- Essendo configurato in *FlywayDB* come *flyway.defaultSchema* quello *metadata*, negli script relativi a *data* va sempre anteposto lo schema quando si fa riferimento alle tabelle
- Essendoci a volte situazioni in cui ci sono viste dello schema data che fanno riferimento a tabelle dello schema metadata (e non il contrario), i primi script devono essere quelli metadata, seguiti da quelli data
- Negli script data/metadata dovrebbero essere anche presenti le istruzioni di creazione delle *sequence* (ove previsto dal tipo di database)
- Per ottimizzare i tempi di esecuzione, nello script relativo allo schema metadata (V1.0.0.1__metadata_schema.sql) andrebbero messe SOLO le istruzioni di creazione tabelle non presenti anche nel file repeatable (R__metadata_schema.sql) dello schema metadata
- Le istruzioni che creano utenti e relative password per data e metadata devono essere tenute su file iniziali separati (es: V1.0.0.0 e V1.0.0.1), per poter in seguito implementare più facilmente integrazioni di sicurezza descritte sotto.

Note Sicurezza

- Solo l'user/role configurato per *FlywayDB* dovrebbe avere tutti i permessi di creazione ruoli e schemi sul database. Gli user/role creati negli script non devono averli per ragioni di sicurezza

- La creazione degli utenti e relative password direttamente nello script, secondo il modus operandi descritto sopra, farebbe rimanere in chiaro dati sensibili come le password negli script. Questo può generare problematiche relative alla sicurezza ed essere oggetto di contestazione da parte dall'IT del cliente. In futuro verrà introdotto un meccanismo dinamico di recupero e sostituzione delle password da *configuration.properties/remapConfiguration.properties*

flywayCommands.groovy

Il file flywayCommands.groovy è un file groovy, reso disponibile qui come punto di estensione per utilizzare le API java messe a disposizione da Flyway. E' preimpostato di base il comando migrate.

In casi tipici di utilizzo, in genere **non ne va cambiato il contenuto**.

Tutte le API java utilizzabili qua dentro sono disponibili nel javadoc

Folder mapguide

Se non si utilizzano mappe/planimetrie si può eliminare questa folder.

Se la folder è necessaria qui vanno inseriti:

- I file resource package (**.mpg**)
- Il file **webconfig.ini**

I file resource package (.mpg)

I resource package di MapGuide. I file, con estensione **.mpg**, vanno esportati da Maestro.

Se non si utilizzano mappe/planimetrie si può eliminare questa folder.

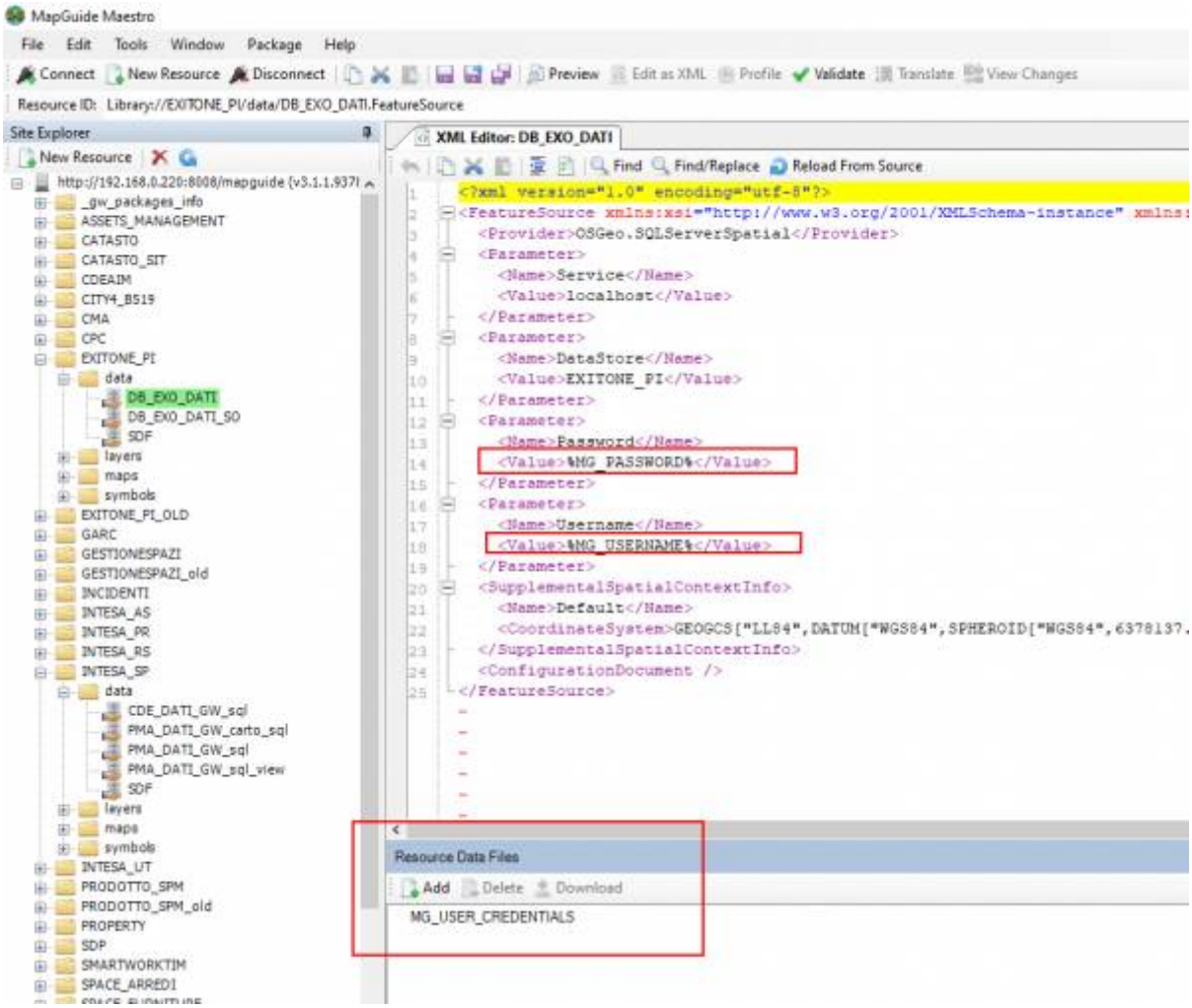
Ci sono delle accortezze da seguire nella produzione dei pacchetti .mpg.

Verifica corretto formato file .mpg)

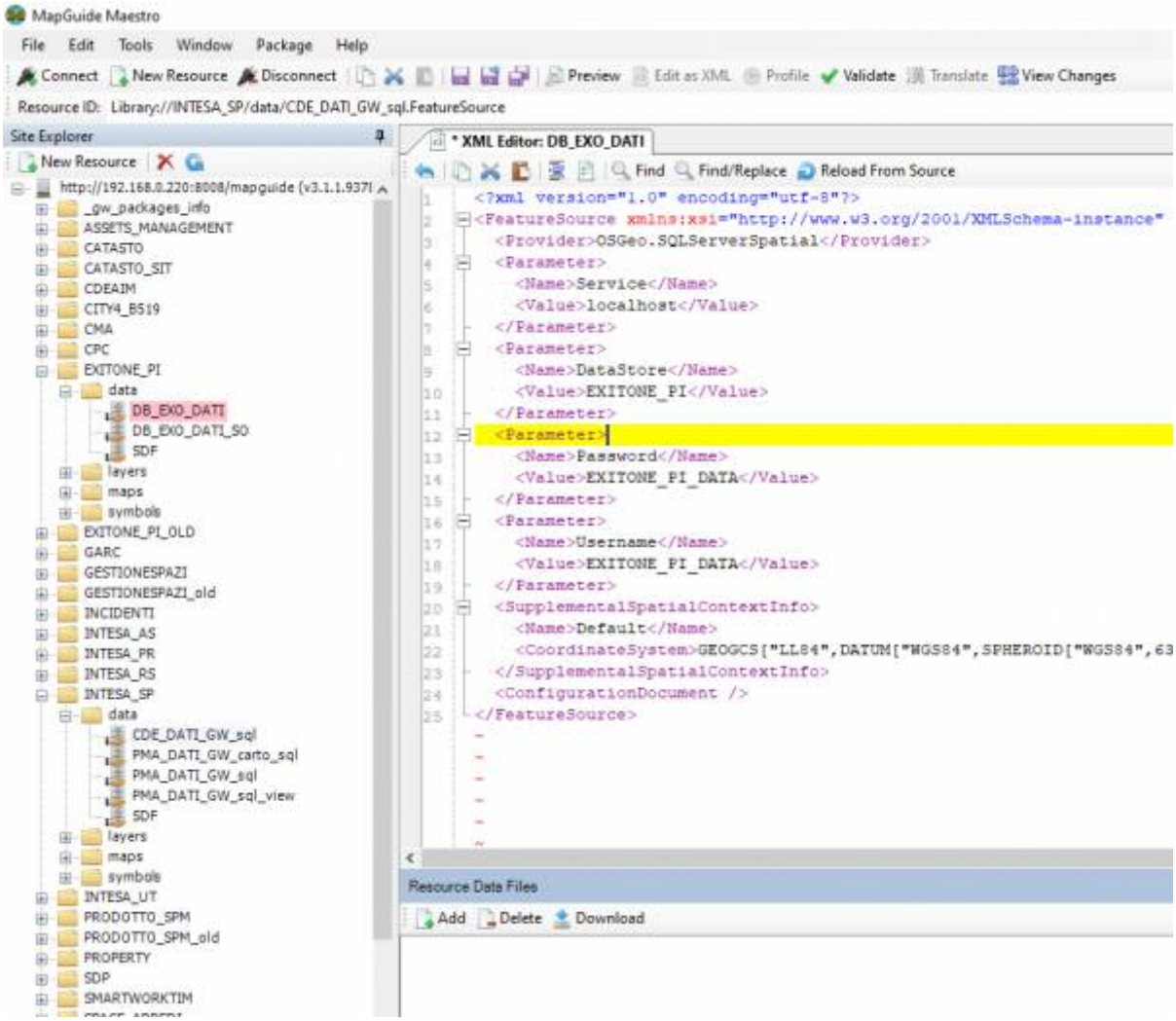
Prima di procedere bisogna assicurarsi che il Feature Source sia nel formato correttamente supportato:

- dalla sub-folder **data** della Library da esportare, aprire il Feature Source in modalità testo XML cosi:
  - *rClick ⇒ Open Resource with XML Editor*
- Assicurarsi che i *<Value>* per i *<Parameter>* con *<Name>* ***Username*** e ***Password*** siano **IN CHIARO** e non rimpiazzati dai place-holder %MG_USERNAME%, %MG_PASSWORD%. Nel caso procedere a scrivere in chiaro Username e Password.
- Nella sezione *Resource Data Files*, in basso, assicurarsi di **rimuovere** la voce **MG_USER_CREDENTIALS**. Nel caso fosse presente, selezionarla e rimuoverla (click su *Delete*)
- Salvare le eventuali modifiche

Prima


MapGuide Maestro
File Edit Tools Window Package Help
Connect New Resource Disconnect Preview Edit as XML Profile Validate Translate View Changes
Resource ID: Library://EXITONE_PI/data/DB_EXO_DATI.FeatureSource
Site Explorer
New Resource
XML Editor: DB_EXO_DATI
Find Find/Replace Reload From Source
<?xml version="1.0" encoding="utf-8"?>
<FeatureSource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<Provider>OSGeo.SQLServerSpatial</Provider>
<Parameter>
<Name>Service</Name>
<Value>localhost</Value>
</Parameter>
<Parameter>
<Name>DataStore</Name>
<Value>EXITONE_PI</Value>
</Parameter>
<Parameter>
<Name>Password</Name>
<Value>%MG_PASSWORD%</Value>
</Parameter>
<Parameter>
<Name>Username</Name>
<Value>%MG_USERNAME%</Value>
</Parameter>
<SupplementalSpatialContextInfo>
<Name>Default</Name>
<CoordinateSystem>GEOGCS["LL84",DATUM["WGS84",SPHEROID["WGS84",6378137.
</SupplementalSpatialContextInfo>
<ConfigurationDocument />
</FeatureSource>
Resource Data Files
Add Delete Download
MG_USER_CREDENTIALS

Dopo



XML Prima

```
<?xml version="1.0" encoding="utf-8"?>
<FeatureSource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:noNamespaceSchemaLocation="FeatureSource-1.0.0.xsd">
  <Provider>OSGeo.SQLServerSpatial</Provider>
  <Parameter>
    <Name>Service</Name>
    <Value>localhost</Value>
  </Parameter>
  <Parameter>
    <Name>DataStore</Name>
    <Value>EXITONE_PI</Value>
  </Parameter>
  <Parameter>
    <Name>Password</Name>
    <Value>%MG_PASSWORD%</Value>
  </Parameter>
  <Parameter>
    <Name>Username</Name>
    <Value>%MG_USERNAME%</Value>
  </Parameter>
```

```
  <SupplementalSpatialContextInfo>
    <Name>Default</Name>
<CoordinateSystem>GEOGCS["LL84",DATUM["WGS84",SPHEROID["WGS84",6378137.000,2
98.25722293]],PRIMEM["Greenwich",0],UNIT["Degree",0.01745329251994]]</Coordi
nateSystem>
  </SupplementalSpatialContextInfo>
  <ConfigurationDocument>config.xml</ConfigurationDocument>
</FeatureSource>
```

XML Dopo

```
<?xml version="1.0" encoding="utf-8"?>
<FeatureSource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:noNamespaceSchemaLocation="FeatureSource-1.0.0.xsd">
  <Provider>OSGeo.SQLServerSpatial</Provider>
  <Parameter>
    <Name>Service</Name>
    <Value>localhost</Value>
  </Parameter>
  <Parameter>
    <Name>DataStore</Name>
    <Value>EXITONE_PI</Value>
  </Parameter>
  <Parameter>
    <Name>Password</Name>
    <Value>EXITONE_PI_DATA</Value>
  </Parameter>
  <Parameter>
    <Name>Username</Name>
    <Value>EXITONE_PI_DATA</Value>
  </Parameter>
  <SupplementalSpatialContextInfo>
    <Name>Default</Name>
<CoordinateSystem>GEOGCS["LL84",DATUM["WGS84",SPHEROID["WGS84",6378137.000,2
98.25722293]],PRIMEM["Greenwich",0],UNIT["Degree",0.01745329251994]]</Coordi
nateSystem>
  </SupplementalSpatialContextInfo>
  <ConfigurationDocument>config.xml</ConfigurationDocument>
</FeatureSource>
```

Solo in questa maniera verranno correttamente usati i parametri in chiaro impostati, o i loro valori sovrascritti tramite i meccanismi presenti nel configuration.properties / remapConfiguration.properties. Queste configurazioni vengono verò impostate una tantum nella prima installazione.

In particolare si faccia riferimento alle proprietà:

1. mapguide.resourcepackage.featuresource.service

2. mapguide.resourcepackage.featuresource.username
3. mapguide.resourcepackage.featuresource.password
4. mapguide.resourcepackage.featuresource.datastore

Creazione Package

Una volta verificata la correttezza del formato si può procedere alla creazione del package:

1. selezionare la Library folder di interesse
2. Menu *Package* ⇒ *Create Package*
3. in 'What to package', lasciando selezionato *Folder* si esporta tutto. Si può anche specificare cartella per cartella
4. in basso si può anche eventualmente deselezionare tipi di risorse che non vi vuole inserire nel package
5. scegliere in *Output file name* il file di destinazione
6. Click su *Ok*
7. spostare il file cosi generato sotto la folder *mapguide*

Errori

In caso di non corretta creazione del package, per mancata esecuzione della procedura, l'errore atteso è: **Error applying package: E:\projects\PRM\gw_package\mapguide\PRM.mgp. A decryption exception occurred.**, che compare nel blocco riepilogativo in fondo.

Esempio

```
--------------------------------------------------------------------------------
--------------------
31-08-2022 10:30:55,198 [main] GEOWEB INFO

GwDeployer - deployMapguideResources()
Now the GwResourcesDeployer wil try to load the webconfig.ini. This is the
loading priority. The GwResourcesDeployer firstly look for the property
mapguide.pathTo.webconfig.ini inside configuration.properties. Secondarily,
if missing, GwResourcesDeployer looks for the file webconfig.ini under the
gw_package folder (provided as parameter)
Property mapguide.pathTo.webconfig.ini not set inside
configuration.properties file
Adjusting path with file name..
adjusted path: E:\projects\PRM\gw_package\mapguide\webconfig.ini
Configured path refers to an existing file
Initializing MapGuide with the file:
E:\projects\PRM\gw_package\mapguide\webconfig.ini
applyResourcePackageInAllMapGuideServer()
Checking if there is a resource package to apply to MapGuide Server..
Loooking for folder E:\projects\PRM\gw_package\mapguide..
Folder exists
building all the resource package in folder:
E:\projects\PRM\gw_package\mapguide
```

```
building resource package named: PRM.mgp
Folder contains number 1 .mgp files..
siteAddressList: [127.0.0.1]
applyResourcePackage() - resourcePackage:
E:\projects\PRM\gw_package\mapguide\PRM.mgp
computeMd5Hex() - file: E:\projects\PRM\gw_package\mapguide\PRM.mgp
md5 computed in nanoseconds: 21581000
computed md5Hex: e8cb8d623db71a88fa6abc2fc291c55c
        nAddressLeft: 1
        attemp number: 1
        currentSiteAddress: 127.0.0.1
Error applying package: E:\projects\PRM\gw_package\mapguide\PRM.mgp. A
decryption exception occurred.
done
--------------------------------------------------------------------------
--------------------
```

Più sopra compare la segnalazione dello stesso mapguide: **2022-08-31 10:30:55,136 [main] ERROR com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService - Probably the file PRM.mgp contains unprocessable contents 2022-08-31 10:30:55,136 [main] ERROR com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService - If PRM.mgp file has been produced using MapGuide Maestro, try to exclude resources of type 'FeatureSource' from the exported .mpg file 2022-08-31 10:30:55,136 [main] ERROR com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService - A decryption exception occurred. org.osgeo.mapguide.MgDecryptionException: A decryption exception occurred.**

Esempio log con stack

```
2022-08-31 10:30:52,822 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
applyResourcePackageInAllMapGuideServer()
2022-08-31 10:30:52,822 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
Checking if there is a resource package to apply to MapGuide Server..
2022-08-31 10:30:52,822 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
Loooking for folder E:\projects\PRM\gw_package\mapguide..
2022-08-31 10:30:52,822 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
Folder exists
2022-08-31 10:30:52,822 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
building all the resource package in folder:
E:\projects\PRM\gw_package\mapguide
2022-08-31 10:30:52,838 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
building resource package named: PRM.mgp
2022-08-31 10:30:53,480 [main] INFO
```

```
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
Folder contains number 1 .mgp files..
2022-08-31 10:30:53,495 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
siteAddressList: [127.0.0.1]
2022-08-31 10:30:53,495 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
applyResourcePackage() - resourcePackage:
E:\projects\PRM\gw_package\mapguide\PRM.mgp
2022-08-31 10:30:53,511 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
computeMd5Hex() - file: E:\projects\PRM\gw_package\mapguide\PRM.mgp
2022-08-31 10:30:53,527 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
md5 computed in nanoseconds: 21581000
2022-08-31 10:30:53,527 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
computed md5Hex: e8cb8d623db71a88fa6abc2fc291c55c
2022-08-31 10:30:53,527 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
nAddressLeft: 1
2022-08-31 10:30:53,527 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
attemp number: 1
2022-08-31 10:30:53,698 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
currentSiteAddress: 127.0.0.1
2022-08-31 10:30:55,136 [main] ERROR
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService - A
decryption exception occurred.
2022-08-31 10:30:55,136 [main] ERROR
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
Probably the file PRM.mgp contains unprocessable contents
2022-08-31 10:30:55,136 [main] ERROR
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
If PRM.mgp file has been produced using MapGuide Maestro, try to exclude
resources of type 'FeatureSource' from the exported .mpg file
2022-08-31 10:30:55,136 [main] ERROR
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService - A
decryption exception occurred.
org.osgeo.mapguide.MgDecryptionException: A decryption exception occurred.
        at
org.osgeo.mapguide.MapGuideJavaApiJNI.MgResourceService_ApplyResourcePackage
(Native Method)
        at
org.osgeo.mapguide.MgResourceService.ApplyResourcePackage(MgResourceService.
java:76)
        at
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService.appl
yResourcePackage(MapguideDeployerService.java:955)
        at
```

```
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService.appl
yAllResourcePackage(MapguideDeployerService.java:834)
        at
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService.depl
oy(MapguideDeployerService.java:278)
        at
com.geowebframework.gwResourcesDeployer.GwDeployer.deployMapguideResources(G
wDeployer.java:282)
        at
com.geowebframework.gwResourcesDeployer.GwDeployer.run(GwDeployer.java:142)
        at
org.springframework.boot.SpringApplication.callRunner(SpringApplication.java
:819)
        at
org.springframework.boot.SpringApplication.callRunners(SpringApplication.jav
a:803)
        at
org.springframework.boot.SpringApplication.run(SpringApplication.java:346)
        at
org.springframework.boot.SpringApplication.run(SpringApplication.java:1340)
        at
org.springframework.boot.SpringApplication.run(SpringApplication.java:1329)
        at
com.geowebframework.gwResourcesDeployer.GwDeployer.main(GwDeployer.java:98)
2022-08-31 10:30:55,198 [main] ERROR
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
Error applying package: E:\projects\PRM\gw_package\mapguide\PRM.mgp. A
decryption exception occurred.
org.osgeo.mapguide.MgDecryptionException: A decryption exception occurred.
        at
org.osgeo.mapguide.MapGuideJavaApiJNI.MgResourceService_ApplyResourcePackage
(Native Method)
        at
org.osgeo.mapguide.MgResourceService.ApplyResourcePackage(MgResourceService.
java:76)
        at
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService.appl
yResourcePackage(MapguideDeployerService.java:955)
        at
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService.appl
yAllResourcePackage(MapguideDeployerService.java:834)
        at
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService.depl
oy(MapguideDeployerService.java:278)
        at
com.geowebframework.gwResourcesDeployer.GwDeployer.deployMapguideResources(G
wDeployer.java:282)
        at
com.geowebframework.gwResourcesDeployer.GwDeployer.run(GwDeployer.java:142)
        at
```

```
org.springframework.boot.SpringApplication.callRunner(SpringApplication.java
:819)
        at
org.springframework.boot.SpringApplication.callRunners(SpringApplication.jav
a:803)
        at
org.springframework.boot.SpringApplication.run(SpringApplication.java:346)
        at
org.springframework.boot.SpringApplication.run(SpringApplication.java:1340)
        at
org.springframework.boot.SpringApplication.run(SpringApplication.java:1329)
        at
com.geowebframework.gwResourcesDeployer.GwDeployer.main(GwDeployer.java:98)
2022-08-31 10:30:55,198 [main] INFO
com.geowebframework.gwResourcesDeployer.service.MapguideDeployerService -
done
2022-08-31 10:30:55,198 [main] INFO
com.geowebframework.gwResourcesDeployer.GwDeployer -

------------------------------------------------------------------------
--------------------
31-08-2022 10:30:55,198 [main] GEOWEB INFO

GwDeployer - deployMapguideResources()
Now the GwResourcesDeployer wil try to load the webconfig.ini. This is the
loading priority. The GwResourcesDeployer firstly look for the property
mapguide.pathTo.webconfig.ini inside configuration.properties. Secondarily,
if missing, GwResourcesDeployer looks for the file webconfig.ini under the
gw_package folder (provided as parameter)
Property mapguide.pathTo.webconfig.ini not set inside
configuration.properties file
Adjusting path with file name..
adjusted path: E:\projects\PRM\gw_package\mapguide\webconfig.ini
Configured path refers to an existing file
Initializing MapGuide with the file:
E:\projects\PRM\gw_package\mapguide\webconfig.ini
applyResourcePackageInAllMapGuideServer()
Checking if there is a resource package to apply to MapGuide Server..
Loooking for folder E:\projects\PRM\gw_package\mapguide..
Folder exists
building all the resource package in folder:
E:\projects\PRM\gw_package\mapguide
building resource package named: PRM.mgp
Folder contains number 1 .mgp files..
siteAddressList: [127.0.0.1]
applyResourcePackage() - resourcePackage:
E:\projects\PRM\gw_package\mapguide\PRM.mgp
computeMd5Hex() - file: E:\projects\PRM\gw_package\mapguide\PRM.mgp
md5 computed in nanoseconds: 21581000
computed md5Hex: e8cb8d623db71a88fa6abc2fc291c55c
        nAddressLeft: 1
```

```
        attemp number: 1
        currentSiteAddress: 127.0.0.1
Error applying package: E:\projects\PRM\gw_package\mapguide\PRM.mgp. A
decryption exception occurred.
done
--------------------------------------------------------------------------------
--------------------
```

Folder rabbitmq

In questa cartella va inserito il file **rabbit_conf.json** con le configurazioni necessarie al funzionamento di RabbitMq.

Se il progetto non utilizza questo servizio, la cartella può essere omessa.

Esempio rabbit_conf.json:

rabbit_conf.json

```
{
    "bindings": [{
            "arguments": {},
            "destination": "ifc.xkt.converter.queue",
            "destination_type": "queue",
            "routing_key": "ifc.xkt.rkey",
            "source": "geoweb.exchange",
            "vhost": "/"
        }
    ],
    "exchanges": [{
            "arguments": {},
            "auto_delete": false,
            "durable": true,
            "internal": false,
            "name": "geoweb.exchange",
            "type": "direct",
            "vhost": "/"
        }
    ],
    "global_parameters": [{
            "name": "internal_cluster_id",
            "value": "rabbitmq-cluster-id-9cPEP91aHZS7A_dLPr21eg"
        }
    ],
    "parameters": [],
    "permissions": [{
            "configure": ".*",
            "read": ".*",
```

```
            "user": "geoweb",
            "vhost": "/",
            "write": ".*"
        }, {
            "configure": ".*",
            "read": ".*",
            "user": "guest",
            "vhost": "/",
            "write": ".*"
        }
    ],
    "policies": [],
    "product_name": "RabbitMQ",
    "product_version": "3.8.19",
    "queues": [{
            "arguments": {
                "x-queue-type": "classic"
            },
            "auto_delete": false,
            "durable": true,
            "name": "ifc.xkt.converter.queue",
            "vhost": "/"
        }
    ],
    "rabbit_version": "3.8.19",
    "rabbitmq_version": "3.8.19",
    "topic_permissions": [],
    "users": [{
            "hashing_algorithm": "rabbit_password_hashing_sha256",
            "limits": {},
            "name": "geoweb",
            "password_hash":
"AP0gMGUB1hl5/8Y5FSdUgbvojAIw33VpW7SVvjJUp/kCOpEV",
            "tags": "administrator"
        }, {
            "hashing_algorithm": "rabbit_password_hashing_sha256",
            "limits": {},
            "name": "guest",
            "password_hash":
"6agy9UL9Apm9tIKuwU6x27YZrTM9qEC7s+se4KqMs1TExHQy",
            "tags": "administrator"
        }
    ],
    "vhosts": [{
            "name": "/"
        }
    ]
}
```

Il file webconfig.ini

Il file **webconfig.ini**, del tutto equivalente a quello utilizzato da Geoweb. Le sue proprietà sono sovrascribili attraverso i file configuration.properties / remapConfiguration.properties nella sezione *Mapguide*.

Nota Importante

Il contenuto del file dipende da dove verrà fatto girare l'eseguibile:

- Nel caso l'eseguibile venga fatto girare dalla macchina dell'applicativo, **si può copiare lo stesso *webconfig.ini* che si trova gia configurato nella cartella *conf***. CONSIGLIATO
- Nel caso l'eseguibile venga fatto girare dalla macchina locale di chi si occupa del deploy, bisogna configurare il webconfig.ini attenendosi alle istruzioni contenuti nelle note esplicative dell'eseguibile. SCONSIGLIATO

Folder keycloak

Se il progetto non prevede l'integrazione con Keycloak, questa folder si può omettere (in tal caso lasciare a **false** il flag *gwResourceDeployer.keycloak.enabled* descritto più sotto).

issue #2 — A partire dalla versione **1.5.0** il deployer dispiega **tutti** i client Keycloak descritti dai file **.json** presenti in questa folder, non più un unico client configurato a parte. Nelle versioni precedenti (introdotte dalla 1.4.0) veniva gestito un solo client, il cui file era indicato dalla property *keycloak.clientRepresentationFile* di *configuration.properties*: tale property, insieme a *keycloak.clientId*, è stata **rimossa**, in quanto il *clientId* viene ora letto direttamente da ciascun file json.

Struttura

Ogni file **.json** presente nella folder rappresenta un client Keycloak da creare/aggiornare (in numero variabile, un file per client). Il contenuto è il *Client Representation* così come esportabile dalla console di amministrazione di Keycloak (*Realm Settings* ⇒ *Clients* ⇒ [client] ⇒ *Export*), e deve contenere obbligatoriamente la property **clientId**, univoca per client.

- **keycloak**
  - gw_access_page_id.json
  - altro_client.json
  - ... (uno per ogni client da dispiegare)

Esempio semplificato di file client:

gw_access_page_id.json

```
{
```

```
    "clientId": "gw_access_page_id",
    "rootUrl": "",
    "adminUrl": "http://192.168.0.224/gwaccesspageweb",
    "baseUrl": "http://wsaip/gwaccesspageweb",
    "enabled": true,
    "clientAuthenticatorType": "client-secret",
    "redirectUris": ["*"],
    "webOrigins": ["http://wsaip"],
    "standardFlowEnabled": true,
    "directAccessGrantsEnabled": true,
    "publicClient": true,
    "protocol": "openid-connect"
}
```

Ad ogni lancio dell'eseguibile, per ciascun file trovato nella folder, il deployer:

1. legge il **clientId** dal contenuto del json
2. cerca, nel realm configurato, un client con quel *clientId*
3. se non lo trova lo **crea**, altrimenti lo **aggiorna** (update) con il contenuto del json

Configurazione (sezione Keycloak di configuration.properties)

Esempio:

```
#########################################################################
# K E Y C L O A K
#########################################################################
keycloak.serverUrl=https://kc.k8s.gwcloud.it/
keycloak.realm=pnrrtest
keycloak.clientSecret=************************
keycloak.baseUrl=https://accesspage.k8s.gwcloud.it/gwaccesspagesweb
keycloak.adminUrl=http://gwaccesspagesweb.pnrrtest.svc:8080/gwaccesspagesweb
```

Le property si dividono in due categorie concettualmente distinte:

Proprietà del server Keycloak target (obbligatorie, uniche e condivise per tutti i client dispiegati nell'esecuzione, analogamente a *jdbc.url* per il database):

- **keycloak.serverUrl**: url del server Keycloak (Admin REST API)
- **keycloak.realm**: realm Keycloak su cui operare
- **keycloak.clientSecret**: client secret usato dal deployer per autenticarsi all'Admin REST API di Keycloak (grant *client_credentials*, con client id tecnico fisso interno **gw-client-credentials**, non configurabile). Riguarda solo l'autenticazione del deployer stesso, **non** i client da creare/aggiornare.

Proprietà della singola applicazione client (OPZIONALI, dalla versione 1.5.0):

- **keycloak.baseUrl**
- **keycloak.adminUrl**

A differenza di *serverUrl*/realm *(proprietà del server, per definizione uniche)*, baseUrl/*adminUrl* descrivono dove vive una specifica applicazione client (usate da Keycloak per redirect/logout/link della console di amministrazione), e potrebbero quindi differire da un client all'altro. Il loro comportamento è il seguente:

- se **valorizzate**, vengono applicate come override a **tutti** i client trovati nella folder, sovrascrivendo quanto eventualmente presente nel singolo json. Ha senso solo se tutti i client dispiegati appartengono alla stessa applicazione/servizio (comportamento storico delle versioni precedenti). Se il json di un client contiene già un valore diverso, il deployer emette un **warning** in log (sia su console/log4j sia nel riepilogo del deploy), per segnalare l'override e prevenire misconfigurazioni silenziose su client che rappresentano applicazioni diverse;
- se **lasciate vuote/commentate**, ciascun client mantiene il proprio **baseUrl**/**adminUrl**, così come definito nel suo file json.

Nota sicurezza

keycloak.clientSecret è una credenziale a tutti gli effetti: va gestita con lo stesso livello di attenzione delle altre credenziali presenti in *configuration.properties* (si veda il meccanismo di remapConfiguration.properties per evitare di lasciarla in chiaro nei repository).

file configuration.properties

E' molto simile configuration.properties di Geoweb. Contiene però solo un sotto insieme delle sezioni. In particolare sono presenti solo le sezioni:

- Database
- MapGuide
- Workflow
- Keycloak (vedi Folder keycloak qui sopra)

Esse lavorano nella stesse modalità descritte qui.

Le parti aggiuntive sono:

flag per abilitare/disabilitare le singole componenti del deployer

Per ogni *flag* il default è **false**. Abilitare solo le parti esclusivamente necessarie.

```
gwResourceDeployer.activiti.enabled=false
gwResourceDeployer.database.enabled=true
gwResourceDeployer.mapguide.enabled=false
gwResourceDeployer.rabbitmq.enabled=false
gwResourceDeployer.keycloak.enabled=false
```

Note

Per una problematica ancora in fase di risoluzione, in caso di presenza del workflow, non si può abilitare tramite flag il workflow fin dalla prima esecuzione (gwResourceDeployer.activiti.enabled=true). In questo caso si procede lanciando l'eseguibile una prima volta, con il flag a false. Poi, dopo aver modificando il flag a true, lo si rilancia.

property di Flyway

C'è una sezione aggiuntiva contenente le stesse property supportate da Flyway qui. Queste per comodità di configurazione sono state riportate tutte in un'unico file di configurazione. Quasi tutte le property sono commentate e, nel caso generale, non andrebbero decommentate. Qui verranno esposte solo le property sulle quali bisogna prestare attenzione:

- **flyway.url** equivalente a *jdbc.url* in Geoweb. Configurazione:
  - Oracle:

    ```
    jdbc:oracle:thin:@<ip_address or server_name>:<port>:<DB_name>
    ```

    . Esempio:

    ```
    flyway.url=jdbc:oracle:thin:@ora11dev.gruppoesc.it:1521:ORA11DEV
    ```

  - Postgres:

    ```
    jdbc:postgresql://<ip_address or server_name>:<port>/<DB_name>
    ```

    . Esempio:

    ```
    flyway.url=jdbc:postgresql://127.0.0.1:5432/CDE35
    ```

  - Sqlserver:

    ```
    jdbc:sqlserver://<ip_address or
    server_name>:<port>;databaseName=<DB_name>
    ```

    . Esempio:

    ```
    flyway.url=jdbc:sqlserver://wexitenergy:1433;databaseName=EXITONE_
    EN
    ```

- **flyway.driver** equivalente a *jdbc.driverClassName* in Geoweb. possibili valori:
  - Oracle:

    ```
    flyway.driver=oracle.jdbc.driver.OracleDriver
    ```

  - Postgres:

    ```
    flyway.driver=org.postgresql.Driver
    ```

    - Sqlserver:

```
flyway.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
```

- **flyway.user** equivalente a *jdbc.username* in Geoweb. E' uno specifico utente creato ad uso esclusivo di Flyway, per esempio **flyway**. Non è l'utente dati/metadati con cui si connetteranno i vari applicativi Geoweb. Ha le guenti caratteristiche:
    - permesso di login
    - superuser
    - permesso creazione ruoli
    - permesso creazioni schema
- **flyway.password** equivalente a *jdbc.password* in Geoweb. Password per l'utente esclusivo di Flyway.
- **flyway.defaultSchema** Impostare lo schema di default per l'utente dei METADATI
- **flyway.schemas** Impostare lo schema di default per l'utente dei METADATI
- **flyway.locations** Impostare il path assoluito alla folder che contiene gli script. Esempio:

```
flyway.locations=filesystem:C:/Path/to/gw_package/database/sql
```

- **flyway.placeholderReplacement** va settato **sempre a false**. Questo perchè spesso nei nostri script abbiamo nei dati delle quesry espressioni del tipo **${}**, usate internamente a Geoweb. Queste espressioni, con il flag a true, sarebbero erroneamnete soggette al tentativo di risoluzione da parte di Falyway, prima dell'esecuzione degli script.
- **flyway.jdbcProperties.X** settare queste property come le equivalenti jdbc.X in Geoweb. Esempio genrico:

```
flyway.jdbcProperties.maxActive=6
flyway.jdbcProperties.minIdle=2
flyway.jdbcProperties.maxIdle=6
flyway.jdbcProperties.validationQuery=select 1
```

  Esempio Oracle (validationQuery custom):

```
flyway.jdbcProperties.maxActive=6
flyway.jdbcProperties.minIdle=2
flyway.jdbcProperties.maxIdle=6
flyway.jdbcProperties.validationQuery=select 1 from dual
```

Note property di Flyway

Attraverso la property **flyway.schemas**, secondo la documentazione ufficiale, sarebbe possibile inserire più schemi, separati da virgola ',', i quali dovrebbero essere tutti creati da flyway. Il primo dovrebbe anche essere quello di default. Ciò eviterebbe di inserire in entrambi i primi script, di dati e metadati, lo SQL statement di creazione dello schema stesso. In realtà accade che impostando anche la property **flyway.defaultSchema**, con più di uno schema configurato su *flyway.schemas*, si ottiene l'errore:

```
The defaultSchema property is specified but is not a member of the schemas
```

```
property
```

Invece, non configurando la property **flyway.defaultSchema**, che comunque sia comporterebbe di impostare lato database lo schema di default per l'utente *FlywayDB* (schema che ancora deve essere creato!), vengono creati degli schema con nome sbagliato: in particolare il primo ha concatenato un charattere '[' all'inizio, mentre l'ultimo ha concatenato un carattere ']' alla fine. Esempio:

```
nomi schema creati da Flyway quando flyway.schemas=cde_data,cde_metadata:
        [cde_data
        cde_metadata]
```

Questo non permette un uso come avrebbe gradito Geoweb. Si tratta certamente di un bug. Non ancora risolto alla versione 8.0.5.

```
<dependency>
  <groupId>org.flywaydb</groupId>
  <artifactId>flyway-core</artifactId>
  <version>8.0.5</version>
</dependency>
```

Questo costringe a mettere lo SQL statement di creazione dello schema nel primo script dei metadati, ma non nel primo script dei dati. Vanno invece sempre inseriti gli script di creazione dell'utente dello schema, e lo script SQL di associazione schema-user.

Template

Qui di seguito un template di esempio. Le proprietà hanno commenti auto esplicativi. Sotto la sezione *Databases*, rimuovere/de-commentare in maniera da lasciare solo il database utilizzato.

configuration.properties

```
#================================================================
#
#         G W   R E S O U R C E S   D E P L O Y E R
#      C O N F I G U R A T I O N   P R O P E R T I E S
#                   T E M P L A T E
#
#================================================================

#enabled, default false, when true the relative section is enabled and
must to be properly configured

gwResourceDeployer.activiti.enabled=false
gwResourceDeployer.database.enabled=true
gwResourceDeployer.mapguide.enabled=false
gwResourceDeployer.rabbitmq.enabled=false
gwResourceDeployer.keycloak.enabled=false
```

```
#####################################################################
# M A P   G U I D E
#####################################################################
mapguide.mgUsername=Administrator
mapguide.mgPassword=admin
#mapguide library path. used to update mapguide layers (destination use
layer)
mapguide.library.path=Library://AEC

# webconfig.ini path
# Geoweb tries to load webconfig.ini firstly from static contents, and
secondarily from classPath.
# If all ways before fail, Geoweb uses this absolute path.
# This is generally used in Geoweb deployments that involve Linux +
WebLogic
#mapguide.pathTo.webconfig.ini=C:\\Users\\giorgio.scali\\Desktop\\gw_pa
ckage\\mapguide\\webconfig.ini

#Next 4 parameters are used to dynamically overrides parameters set
inside webconfig.ini. They works in independent way each others. Remove
comment to override
#mapguide.webconfig.ini.AdministrativeConnectionProperties.Port=2810
#mapguide.webconfig.ini.ClientConnectionProperties.Port=2811
#mapguide.webconfig.ini.SiteConnectionProperties.Port=2812
#mapguide.webconfig.ini.SiteConnectionProperties.IpAddress=10.31.219.23
6

# resource package auto loading. Feature source
#Next 4, optional, parameters are used to auto configure feature source
database parameters
#during the procedure that applies resource package to all configured
MapGuide servers
#if not configured here the existing values inside .mpg file will not
to be overridden
#service often refers to the same machine in configured in jdbc.url,
but here MUST to be expressed like an IPV4 format address.
mapguide.resourcepackage.featuresource.service=127.0.0.1:5432
mapguide.resourcepackage.featuresource.username=cde_data
mapguide.resourcepackage.featuresource.password=cde_data
mapguide.resourcepackage.featuresource.datastore=CDE35

#@Deprecated from 4.4.0 (but supported yet)
#mgUsername=
#mgPassword=
#geowebalias=
#pathTo.webconfig.ini=

#####################################################################
# R A B B I T M Q
#####################################################################
```

```
gwRabbitConfUri=http://localhost:15672/api/definitions

#########################################################################
# K E Y C L O A K
#########################################################################
# serverUrl/realm/clientSecret: proprieta' del server Keycloak target,
obbligatorie e uniche per l'esecuzione.
# clientSecret e' usato solo per autenticare il deployer alla Admin
REST API (non riguarda i client dispiegati).
keycloak.serverUrl=https://kc.example.com/
keycloak.realm=myrealm
keycloak.clientSecret=CHANGE_ME

# baseUrl/adminUrl: OPZIONALI. Proprieta' della singola applicazione
client, non del server.
# Se valorizzate vengono applicate, come override condiviso, a TUTTI i
client json trovati in gw_package/keycloak
# (ha senso solo se rappresentano la stessa applicazione/servizio: in
caso di valore diverso nel json viene loggato un warning).
# Se lasciate commentate, ogni client mantiene il proprio
baseUrl/adminUrl come definito nel suo file json.
#keycloak.baseUrl=https://app.example.com/mywebapp
#keycloak.adminUrl=http://mywebapp.svc:8080/mywebapp

#########################################################################
# W O R K F L O W
#########################################################################
#This flag, default false, force to deploy all Process Definition
#inside Activiti at every server startup, even if already deployed.
#Usually all Activiti Process Definition, contained in one or more
#.bpmn files, under the folder [geowebfolder]/activiti/ will to be
#deployed inside Activiti, in a automatic way, during webapp
initialization
#If the Process Definition results already deployed, the deploy is
skipped
#This flag overrides this mechanism
workflow.activiti.forceDeploy.enabled=false


#########################################################################
# D A T A B A S E S
#########################################################################

#########################################################################
# database ORACLE
#########################################################################

#jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbc.url=jdbc:oracle:thin:@127.0.0.1:1521:CDE35
#jdbc.username=CDE_DATA
```

```
#jdbc.password=CDE_DATA
#jdbc.maxActive=6
#jdbc.minIdle=2
#jdbc.maxIdle=6
#jdbc.validationQuery=select 1 from dual
#
#jdbcmetadata.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbcmetadata.url=jdbc:oracle:thin:@127.0.0.1:1521:CDE35
#jdbcmetadata.username=CDE_METADATA
#jdbcmetadata.password=CDE_METADATA
#jdbcmetadata.maxActive=6
#jdbcmetadata.minIdle=2
#jdbcmetadata.maxIdle=6
#jdbcmetadata.validationQuery=select 1 from dual


#####################################################################
# database POSTGRES
#####################################################################

jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://127.0.0.1:5432/CDE35
jdbc.username=cde_data
jdbc.password=cde_data
jdbc.maxActive=6
jdbc.minIdle=2
jdbc.maxIdle=6
jdbc.validationQuery=select 1

jdbcmetadata.driverClassName=org.postgresql.Driver
jdbcmetadata.url=jdbc:postgresql://127.0.0.1:5432/CDE35
jdbcmetadata.username=cde_metadata
jdbcmetadata.password=cde_metadata
jdbcmetadata.maxActive=6
jdbcmetadata.minIdle=2
jdbcmetadata.maxIdle=6
jdbcmetadata.validationQuery=select 1


#####################################################################
# database SQLSERVER
#####################################################################

#jdbc.driverClassName=com.microsoft.sqlserver.#jdbc.SQLServerDriver
#jdbc.url=jdbc:sqlserver://127.0.0.1:1433;databaseName=CDE35
#jdbc.username=CDE_DATA
#jdbc.password=CDE_DATA
#jdbc.maxActive=24
#jdbc.minIdle=8
#jdbc.maxIdle=24
```

```
#jdbc.validationQuery=select 1

#jdbcmetadata.driverClassName=com.microsoft.sqlserver.#jdbc.SQLServerDr
iver
#jdbcmetadata.url=jdbc:sqlserver://127.0.0.1:1433;databaseName=CDE35
#jdbcmetadata.username=CDE_METADATA
#jdbcmetadata.password=CDE_METADATA
#jdbcmetadata.maxActive=24
#jdbcmetadata.minIdle=8
#jdbcmetadata.maxIdle=24
#jdbcmetadata.validationQuery=select 1



#JNDI DRIVEN
#jndiName.metadata=java:jboss/drei0/gwMetadata
#jndiName.data=java:jboss/drei0/gwData



#*********************************************************************
#
# F L Y W A Y D B
#
#*********************************************************************

# JDBC url to use to connect to the database
# Examples
# --------
# Most drivers are included out of the box.
# * = JDBC driver must be downloaded and installed in /drivers manually
# ** = TNS_ADMIN environment variable must point to the directory of
where tnsnames.ora resides
# Aurora MySQL      :
jdbc:mysql://<instance>.<region>.rds.amazonaws.com:<port>/<database>?<k
ey1>=<value1>&<key2>=<value2>...
# Aurora PostgreSQL :
jdbc:postgresql://<instance>.<region>.rds.amazonaws.com:<port>/<databas
e>?<key1>=<value1>&<key2>=<value2>...
# CockroachDB       :
jdbc:postgresql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<valu
e2>...
# DB2*              : jdbc:db2://<host>:<port>/<database>
# Derby             :
jdbc:derby:<subsubprotocol>:<database><;attribute=value>
# Firebird          :
jdbc:firebirdsql://<host>[:<port>]/<database>?<key1>=<value1>&<key2>=<v
alue2>...
# H2                : jdbc:h2:<file>
# HSQLDB            : jdbc:hsqldb:file:<file>
# Informix*         : jdbc:informix-
```

```
sqli://<host>:<port>/<database>:informixserver=dev
# MariaDB           :
jdbc:mariadb://<host>:<port>/<database>?<key1>=<value1>&<key2>=<value2>
...
# MySQL             :
jdbc:mysql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<value2>..
.
# Oracle            : jdbc:oracle:thin:@//<host>:<port>/<service>
# Oracle (TNS)**    : jdbc:oracle:thin:@<tns_entry>
# PostgreSQL        :
jdbc:postgresql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<valu
e2>...
# SAP HANA*         : jdbc:sap://<host>:<port>/?databaseName=<database>
# Snowflake*        :
jdbc:snowflake://<account>.snowflakecomputing.com/?db=<database>&wareho
use=<warehouse>&role=<role>...
# SQL Server        :
jdbc:sqlserver://<host>:<port>;databaseName=<database>
# SQLite            : jdbc:sqlite:<database>
# Sybase ASE        : jdbc:jtds:sybase://<host>:<port>/<database>
# Redshift*         : jdbc:redshift://<host>:<port>/<database>
flyway.url=jdbc:postgresql://127.0.0.1:5432/CDE35

# Fully qualified classname of the JDBC driver (autodetected by default
based on flyway.url)
flyway.driver=org.postgresql.Driver

# User to use to connect to the database. Flyway will prompt you to
enter it if not specified, and if the JDBC
# connection is not using a password-less method of authentication.
flyway.user=flyway

# Password to use to connect to the database. Flyway will prompt you to
enter it if not specified, and if the JDBC
# connection is not using a password-less method of authentication.
flyway.password=flyway

# The default schema managed by Flyway. This schema name is case-
sensitive. If not specified, but <i>flyway.schemas</i> is, Flyway uses
the first schema
# in that list. If that is also not specified, Flyway uses the default
schema for the database connection.
# Consequences:
# - This schema will be the one containing the schema history table.
# - This schema will be the default for the database connection
(provided the database supports this concept).
flyway.defaultSchema=cde_metadata

# Comma-separated list of schemas managed by Flyway. These schema names
are case-sensitive. If not specified, Flyway uses
```

```
# the default schema for the database connection. If
<i>flyway.defaultSchema</i> is not specified, then the first of
# this list also acts as default schema.
# Consequences:
# - Flyway will automatically attempt to create all these schemas,
unless they already exist.
# - The schemas will be cleaned in the order of this list.
# - If Flyway created them, the schemas themselves will be dropped when
cleaning.
flyway.schemas=cde_metadata

# Whether Flyway should attempt to create the schemas specified in the
schemas property
# flyway.createSchemas=

# Comma-separated list of locations to scan recursively for migrations.
(default: filesystem:<<INSTALL-DIR>>/sql)
# The location type is determined by its prefix.
# Unprefixed locations or locations starting with classpath: point to a
package on the classpath and may contain
# both SQL and Java-based migrations.
#
# Locations starting with filesystem: point to a directory on the
filesystem, may only
# contain SQL migrations and are only scanned recursively down non-
hidden directories.
# Locations starting with s3: point to a bucket in AWS S3, may only
contain SQL migrations, and are scanned
# recursively. They are in the format
s3:<bucket>(/optionalfolder/subfolder)
# Locations starting with gcs: point to a bucket in Google Cloud
Storage, may only contain SQL migrations, and are scanned
# recursively. They are in the format
gcs:<bucket>(/optionalfolder/subfolder)
# Wildcards can be used to reduce duplication of location paths. (e.g.
filesystem:migrations/*/oracle) Supported wildcards:
# ** : Matches any 0 or more directories
# *  : Matches any 0 or more non-separator characters
# ?  : Matches any 1 non-separator character
#
flyway.locations=filesystem:C:/gw_package/database/sql

# Whether placeholders should be replaced. (default: true)
flyway.placeholderReplacement=false

# Placeholders to replace in SQL migrations.
# flyway.placeholders.user=
# flyway.placeholders.my_other_placeholder=

# Prefix of every placeholder. (default: ${ )
# flyway.placeholderPrefix=
```

```
# Suffix of every placeholder. (default: } )
# flyway.placeholderSuffix=

# JDBC properties to pass to the JDBC driver when establishing a
connection.
# Flyway Teams only
# flyway.jdbcProperties.myProperty=
# flyway.jdbcProperties.myOtherProperty=

flyway.jdbcProperties.maxActive=6
flyway.jdbcProperties.minIdle=2
flyway.jdbcProperties.maxIdle=6
flyway.jdbcProperties.validationQuery=select 1
```

In questa versione sono presenti anche tutti le property ulteriori, commentate, di Flyway.

configuration.properties

```
#*****************************************************************
#
#          G W   R E S O U R C E S   D E P L O Y E R
#       C O N F I G U R A T I O N   P R O P E R T I E S
#                     T E M P L A T E
#
#*****************************************************************

#enabled, default false, when true the relative section is enabled and
must to be properly configured

gwResourceDeployer.activiti.enabled=false
gwResourceDeployer.database.enabled=true
gwResourceDeployer.mapguide.enabled=false
gwResourceDeployer.rabbitmq.enabled=false
gwResourceDeployer.keycloak.enabled=false


#####################################################################
# M A P   G U I D E
#####################################################################
mapguide.mgUsername=Administrator
mapguide.mgPassword=admin
#mapguide library path. used to update mapguide layers (destination use
layer)
mapguide.library.path=Library://CDE

# webconfig.ini path
# Geoweb tries to load webconfig.ini firstly from static contents, and
secondarily from classPath.
```

```
# If all ways before fail, Geoweb uses this absolute path.
# This is generally used in Geoweb deployments that involve Linux +
WebLogic
#mapguide.pathTo.webconfig.ini=C:\\Users\\giorgio.scali\\Desktop\\gw_pa
ckage\\mapguide\\webconfig.ini

#Next 4 parameters are used to dynamically overrides parameters set
inside webconfig.ini. They works in independent way each others. Remove
comment to override
#mapguide.webconfig.ini.AdministrativeConnectionProperties.Port=2810
#mapguide.webconfig.ini.ClientConnectionProperties.Port=2811
#mapguide.webconfig.ini.SiteConnectionProperties.Port=2812
#mapguide.webconfig.ini.SiteConnectionProperties.IpAddress=10.31.219.23
6

# resource package auto loading. Feature source
#Next 4, optional, parameters are used to auto configure feature source
database parameters
#during the procedure that applies resource package to all configured
MapGuide servers
#if not configured here the existing values inside .mpg file will not
to be overridden
#service often refers to the same machine in configured in jdbc.url,
but here MUST to be expressed like an IPV4 format address.
mapguide.resourcepackage.featuresource.service=127.0.0.1:5432
mapguide.resourcepackage.featuresource.username=cde_data
mapguide.resourcepackage.featuresource.password=cde_data
mapguide.resourcepackage.featuresource.datastore=CDE35

#@Deprecated from 4.4.0 (but supported yet)
#mgUsername=
#mgPassword=
#geowebalias=
#pathTo.webconfig.ini=

#########################################################################
# R A B B I T M Q
#########################################################################

gwRabbitConfUri=http://localhost:15672/api/definitions

#########################################################################
# K E Y C L O A K
#########################################################################
# serverUrl/realm/clientSecret: proprieta' del server Keycloak target,
obbligatorie e uniche per l'esecuzione.
# clientSecret e' usato solo per autenticare il deployer alla Admin
REST API (non riguarda i client dispiegati).
keycloak.serverUrl=https://kc.example.com/
keycloak.realm=myrealm
keycloak.clientSecret=CHANGE_ME
```

```
# baseUrl/adminUrl: OPZIONALI. Proprieta' della singola applicazione
client, non del server.
# Se valorizzate vengono applicate, come override condiviso, a TUTTI i
client json trovati in gw_package/keycloak
# (ha senso solo se rappresentano la stessa applicazione/servizio: in
caso di valore diverso nel json viene loggato un warning).
# Se lasciate commentate, ogni client mantiene il proprio
baseUrl/adminUrl come definito nel suo file json.
#keycloak.baseUrl=https://app.example.com/mywebapp
#keycloak.adminUrl=http://mywebapp.svc:8080/mywebapp

#####################################################################
# W O R K F L O W
#####################################################################
#This flag, default false, force to deploy all Process Definition
#inside Activiti at every server startup, even if already deployed.
#Usually all Activiti Process Definition, contained in one or more
#.bpmn files, under the folder [geowebfolder]/activiti/ will to be
#deployed inside Activiti, in a automatic way, during webapp
initialization
#If the Process Definition results already deployed, the deploy is
skipped
#This flag overrides this mechanism
workflow.activiti.forceDeploy.enabled=false


#####################################################################
# D A T A B A S E S
#####################################################################

#####################################################################
# database ORACLE
#####################################################################

#jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbc.url=jdbc:oracle:thin:@127.0.0.1:1521:CDE35
#jdbc.username=CDE_DATA
#jdbc.password=CDE_DATA
#jdbc.maxActive=6
#jdbc.minIdle=2
#jdbc.maxIdle=6
#jdbc.validationQuery=select 1 from dual
#
#jdbcmetadata.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbcmetadata.url=jdbc:oracle:thin:@127.0.0.1:1521:CDE35
#jdbcmetadata.username=CDE_METADATA
#jdbcmetadata.password=CDE_METADATA
#jdbcmetadata.maxActive=6
#jdbcmetadata.minIdle=2
```

```
#jdbcmetadata.maxIdle=6
#jdbcmetadata.validationQuery=select 1 from dual


#####################################################################
# database POSTGRES
#####################################################################

jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://127.0.0.1:5432/CDE35
jdbc.username=cde_data
jdbc.password=cde_data
jdbc.maxActive=6
jdbc.minIdle=2
jdbc.maxIdle=6
jdbc.validationQuery=select 1

jdbcmetadata.driverClassName=org.postgresql.Driver
jdbcmetadata.url=jdbc:postgresql://127.0.0.1:5432/CDE35
jdbcmetadata.username=cde_metadata
jdbcmetadata.password=cde_metadata
jdbcmetadata.maxActive=6
jdbcmetadata.minIdle=2
jdbcmetadata.maxIdle=6
jdbcmetadata.validationQuery=select 1


#####################################################################
# database SQLSERVER
#####################################################################

#jdbc.driverClassName=com.microsoft.sqlserver.#jdbc.SQLServerDriver
#jdbc.url=jdbc:sqlserver://127.0.0.1:1433;databaseName=CDE35
#jdbc.username=CDE_DATA
#jdbc.password=CDE_DATA
#jdbc.maxActive=24
#jdbc.minIdle=8
#jdbc.maxIdle=24
#jdbc.validationQuery=select 1

#jdbcmetadata.driverClassName=com.microsoft.sqlserver.#jdbc.SQLServerDr
iver
#jdbcmetadata.url=jdbc:sqlserver://127.0.0.1:1433;databaseName=CDE35
#jdbcmetadata.username=CDE_METADATA
#jdbcmetadata.password=CDE_METADATA
#jdbcmetadata.maxActive=24
#jdbcmetadata.minIdle=8
#jdbcmetadata.maxIdle=24
#jdbcmetadata.validationQuery=select 1
```

```
#JNDI DRIVEN
#jndiName.metadata=java:jboss/drei0/gwMetadata
#jndiName.data=java:jboss/drei0/gwData



#*********************************************************************
#
# F L Y W A Y D B
#
#*********************************************************************

# JDBC url to use to connect to the database
# Examples
# --------
# Most drivers are included out of the box.
# * = JDBC driver must be downloaded and installed in /drivers manually
# ** = TNS_ADMIN environment variable must point to the directory of
where tnsnames.ora resides
# Aurora MySQL      :
jdbc:mysql://<instance>.<region>.rds.amazonaws.com:<port>/<database>?<k
ey1>=<value1>&<key2>=<value2>...
# Aurora PostgreSQL :
jdbc:postgresql://<instance>.<region>.rds.amazonaws.com:<port>/<databas
e>?<key1>=<value1>&<key2>=<value2>...
# CockroachDB       :
jdbc:postgresql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<valu
e2>...
# DB2*              : jdbc:db2://<host>:<port>/<database>
# Derby             :
jdbc:derby:<subsubprotocol>:<database><;attribute=value>
# Firebird          :
jdbc:firebirdsql://<host>[:<port>]/<database>?<key1>=<value1>&<key2>=<v
alue2>...
# H2                : jdbc:h2:<file>
# HSQLDB            : jdbc:hsqldb:file:<file>
# Informix*         : jdbc:informix-
sqli://<host>:<port>/<database>:informixserver=dev
# MariaDB           :
jdbc:mariadb://<host>:<port>/<database>?<key1>=<value1>&<key2>=<value2>
...
# MySQL             :
jdbc:mysql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<value2>..
.
# Oracle            : jdbc:oracle:thin:@//<host>:<port>/<service>
# Oracle (TNS)**    : jdbc:oracle:thin:@<tns_entry>
# PostgreSQL        :
jdbc:postgresql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<valu
e2>...
```

```
# SAP HANA*          : jdbc:sap://<host>:<port>/?databaseName=<database>
# Snowflake*         :
jdbc:snowflake://<account>.snowflakecomputing.com/?db=<database>&wareho
use=<warehouse>&role=<role>...
# SQL Server         :
jdbc:sqlserver://<host>:<port>;databaseName=<database>
# SQLite             : jdbc:sqlite:<database>
# Sybase ASE         : jdbc:jtds:sybase://<host>:<port>/<database>
# Redshift*          : jdbc:redshift://<host>:<port>/<database>
flyway.url=jdbc:postgresql://127.0.0.1:5432/CDE35

# Fully qualified classname of the JDBC driver (autodetected by default
based on flyway.url)
flyway.driver=org.postgresql.Driver

# User to use to connect to the database. Flyway will prompt you to
enter it if not specified, and if the JDBC
# connection is not using a password-less method of authentication.
flyway.user=flyway

# Password to use to connect to the database. Flyway will prompt you to
enter it if not specified, and if the JDBC
# connection is not using a password-less method of authentication.
flyway.password=flyway

# The maximum number of retries when attempting to connect to the
database. After each failed attempt,
# Flyway will wait 1 second before attempting to connect again, up to
the maximum number of times specified
# by connectRetries. (default: 0)
# flyway.connectRetries=

# The SQL statements to run to initialize a new database connection
immediately after opening it. (default: none)
# flyway.initSql=

# The default schema managed by Flyway. This schema name is case-
sensitive. If not specified, but <i>flyway.schemas</i> is, Flyway uses
the first schema
# in that list. If that is also not specified, Flyway uses the default
schema for the database connection.
# Consequences:
# - This schema will be the one containing the schema history table.
# - This schema will be the default for the database connection
(provided the database supports this concept).
flyway.defaultSchema=cde_metadata

# Comma-separated list of schemas managed by Flyway. These schema names
are case-sensitive. If not specified, Flyway uses
# the default schema for the database connection. If
<i>flyway.defaultSchema</i> is not specified, then the first of
```

```
# this list also acts as default schema.
# Consequences:
# - Flyway will automatically attempt to create all these schemas,
unless they already exist.
# - The schemas will be cleaned in the order of this list.
# - If Flyway created them, the schemas themselves will be dropped when
cleaning.
flyway.schemas=cde_metadata

# Whether Flyway should attempt to create the schemas specified in the
schemas property
# flyway.createSchemas=

# Name of Flyway's schema history table (default:
flyway_schema_history)
# By default (single-schema mode) the schema history table is placed in
the default schema for the connection
# provided by the datasource.
# When the flyway.schemas property is set (multi-schema mode), the
schema history table is placed in the first
# schema of the list.
# flyway.table=

# The tablespace where to create the schema history table that will be
used by Flyway. If not specified, Flyway uses
# the default tablespace for the database connection.
# This setting is only relevant for databases that do support the
notion of tablespaces. Its value is simply
# ignored for all others.
# flyway.tablespace=

# Comma-separated list of locations to scan recursively for migrations.
(default: filesystem:<<INSTALL-DIR>>/sql)
# The location type is determined by its prefix.
# Unprefixed locations or locations starting with classpath: point to a
package on the classpath and may contain
# both SQL and Java-based migrations.
#
# Locations starting with filesystem: point to a directory on the
filesystem, may only
# contain SQL migrations and are only scanned recursively down non-
hidden directories.
# Locations starting with s3: point to a bucket in AWS S3, may only
contain SQL migrations, and are scanned
# recursively. They are in the format
s3:<bucket>(/optionalfolder/subfolder)
# Locations starting with gcs: point to a bucket in Google Cloud
Storage, may only contain SQL migrations, and are scanned
# recursively. They are in the format
gcs:<bucket>(/optionalfolder/subfolder)
```

```
# Wildcards can be used to reduce duplication of location paths. (e.g.
filesystem:migrations/*/oracle) Supported wildcards:
# ** : Matches any 0 or more directories
# *  : Matches any 0 or more non-separator characters
# ?  : Matches any 1 non-separator character
#
flyway.locations=filesystem:C:/gw_package/database/sql

# Comma-separated list of fully qualified class names of custom
MigrationResolver to use for resolving migrations.
# flyway.resolvers=

# If set to true, default built-in resolvers (jdbc, spring-jdbc and
sql) are skipped and only custom resolvers as
# defined by 'flyway.resolvers' are used. (default: false)
# flyway.skipDefaultResolvers=

# Comma-separated list of directories containing JDBC drivers and Java-
based migrations. (default: <INSTALL-DIR>/jars)
# flyway.jarDirs=

# File name prefix for versioned SQL migrations (default: V)
# Versioned SQL migrations have the following file name structure:
prefixVERSIONseparatorDESCRIPTIONsuffix,
# which using the defaults translates to V1_1__My_description.sql
# flyway.sqlMigrationPrefix=

# The file name prefix for undo SQL migrations. (default: U)
# Undo SQL migrations are responsible for undoing the effects of the
versioned migration with the same version.
# They have the following file name structure:
prefixVERSIONseparatorDESCRIPTIONsuffix,
# which using the defaults translates to U1.1__My_description.sql
# Flyway Teams only
# flyway.undoSqlMigrationPrefix=

# File name prefix for repeatable SQL migrations (default: R)
# Repeatable SQL migrations have the following file name structure:
prefixSeparatorDESCRIPTIONsuffix,
# which using the defaults translates to R__My_description.sql
# flyway.repeatableSqlMigrationPrefix=

# File name separator for Sql migrations (default: __)
# SQL migrations have the following file name structure:
prefixVERSIONseparatorDESCRIPTIONsuffix,
# which using the defaults translates to V1_1__My_description.sql
# flyway.sqlMigrationSeparator=

# Comma-separated list of file name suffixes for SQL migrations.
(default: .sql)
# SQL migrations have the following file name structure:
```

```
prefixVERSIONseparatorDESCRIPTIONsuffix,
# which using the defaults translates to V1_1__My_description.sql
# Multiple suffixes (like .sql,.pkg,.pkb) can be specified for easier
compatibility with other tools such as
# editors with specific file associations.
# flyway.sqlMigrationSuffixes=

# Whether to stream SQL migrations when executing them. (default:
false)
# Streaming doesn't load the entire migration in memory at once.
Instead each statement is loaded individually.
# This is particularly useful for very large SQL migrations composed of
multiple MB or even GB of reference data,
# as this dramatically reduces Flyway's memory consumption.
# Flyway Teams only
# flyway.stream=

# Whether to batch SQL statements when executing them. (default: false)
# Batching can save up to 99 percent of network roundtrips by sending
up to 100 statements at once over the
# network to the database, instead of sending each statement
individually. This is particularly useful for very
# large SQL migrations composed of multiple MB or even GB of reference
data, as this can dramatically reduce
# the network overhead. This is supported for INSERT, UPDATE, DELETE,
MERGE and UPSERT statements.
# All other statements are automatically executed without batching.
# Flyway Teams only
# flyway.batch=

# Encoding of SQL migrations (default: UTF-8). Caution: changing the
encoding after migrations have been run
# will invalidate the calculated checksums and require a `flyway
repair`.
# flyway.encoding=

# Whether placeholders should be replaced. (default: true)
flyway.placeholderReplacement=false

# Placeholders to replace in SQL migrations.
# flyway.placeholders.user=
# flyway.placeholders.my_other_placeholder=

# Prefix of every placeholder. (default: ${ )
# flyway.placeholderPrefix=

# Suffix of every placeholder. (default: } )
# flyway.placeholderSuffix=

# Target version up to which Flyway should consider migrations.
```

```
# Defaults to 'latest'
# Special values:
# - 'current': designates the current version of the schema
# - 'latest': the latest version of the schema, as defined by the
migration with the highest version
# flyway.target=

# Comma separated list of migrations that Flyway should consider when
migrating.
# Migrations are considered in the order that they are supplied to
cherryPick
# Leave blank to consider all discovered migrations.
# Values should be the version for versioned migrations (e.g. 1, 2.4,
6.5.3) or the description for repeatable migrations (e.g. Insert_Data,
Create_Table)
# Flyway Teams only
# flyway.cherryPick=

# Whether to automatically call validate or not when running migrate.
(default: true)
# flyway.validateOnMigrate=

# Whether to automatically call clean or not when a validation error
occurs. (default: false)
# This is exclusively intended as a convenience for development. even
though we
# strongly recommend not to change migration scripts once they have
been checked into SCM and run, this provides a
# way of dealing with this case in a smooth manner. The database will
be wiped clean automatically, ensuring that
# the next migration will bring you back to the state checked into SCM.
# Warning! Do not enable in production!
# flyway.cleanOnValidationError=

# Whether to disable clean. (default: false)
# This is especially useful for production environments where running
clean can be quite a career limiting move.
# flyway.cleanDisabled=

# The version to tag an existing schema with when executing baseline.
(default: 1)
# flyway.baselineVersion=

# The description to tag an existing schema with when executing
baseline. (default: << Flyway Baseline >>)
# flyway.baselineDescription=

# Whether to automatically call baseline when migrate is executed
against a non-empty schema with no schema history
# table. This schema will then be initialized with the baselineVersion
before executing the migrations.
```

```
# Only migrations above baselineVersion will then be applied.
# This is useful for initial Flyway production deployments on projects
with an existing DB.
# Be careful when enabling this as it removes the safety net that
ensures
# Flyway does not migrate the wrong database in case of a configuration
mistake! (default: false)
# flyway.baselineOnMigrate=

# Whether Flyway should skip actually executing the contents of the
migrations and only update the schema history table. (default: false)
# This should be used when you have applied a migration manually (via
executing the sql yourself, or via an IDE), and
# just want the schema history table to reflect this.
#
# Use in conjunction with flyway.cherryPick to skip specific migrations
instead of all pending ones.
# Flyway Teams only
# flyway.skipExecutingMigrations=

# Allows migrations to be run "out of order" (default: false).
# If you already have versions 1 and 3 applied, and now a version 2 is
found,
# it will be applied too instead of being ignored.
# flyway.outOfOrder=

# Whether Flyway should output a table with the results of queries when
executing migrations (default: true).
# Flyway Teams only
# flyway.outputQueryResults=

# This allows you to tie in custom code and logic to the Flyway
lifecycle notifications (default: empty).
# Set this to a comma-separated list of fully qualified class names of
org.flywaydb.core.api.callback.Callback implementations, or packages to
scan for these classes.
# flyway.callbacks=

# If set to true, default built-in callbacks (SQL) are skipped and only
custom callback as
# defined by 'flyway.callbacks' are used. (default: false)
# flyway.skipDefaultCallbacks=

# Ignore missing migrations when reading the schema history table.
These are migrations that were performed by an
# older deployment of the application that are no longer available in
this version. For example: we have migrations
# available on the classpath with versions 1.0 and 3.0. The schema
history table indicates that a migration with
# version 2.0 (unknown to us) has also been applied. Instead of bombing
```

```
out (fail fast) with an exception, a
# warning is logged and Flyway continues normally. This is useful for
situations where one must be able to deploy
# a newer version of the application even though it doesn't contain
migrations included with an older one anymore.
# Note that if the most recently applied migration is removed, Flyway
has no way to know it is missing and will
# mark it as future instead.
# true to continue normally and log a warning, false to fail fast with
an exception. (default: false)
# flyway.ignoreMissingMigrations=

# Ignore ignored migrations when reading the schema history table.
These are migrations that were added in between
# already migrated migrations in this version. For example: we have
migrations available on the classpath with
# versions from 1.0 to 3.0. The schema history table indicates that
version 1 was finished on 1.0.15, and the next
# one was 2.0.0. But with the next release a new migration was added to
version 1: 1.0.16. Such scenario is ignored
# by migrate command, but by default is rejected by validate. When
ignoreIgnoredMigrations is enabled, such case
# will not be reported by validate command. This is useful for
situations where one must be able to deliver
# complete set of migrations in a delivery package for multiple
versions of the product, and allows for further
# development of older versions.
# true to continue normally, false to fail fast with an exception.
(default: false)
# flyway.ignoreIgnoredMigrations=

# Ignore pending migrations when reading the schema history table.
These are migrations that are available
# but have not yet been applied. This can be useful for verifying that
in-development migration changes
# don't contain any validation-breaking changes of migrations that have
already been applied to a production
# environment, e.g. as part of a CI/CD process, without failing because
of the existence of new migration versions.
# (default: false)
# flyway.ignorePendingMigrations=

# Ignore future migrations when reading the schema history table. These
are migrations that were performed by a
# newer deployment of the application that are not yet available in
this version. For example: we have migrations
# available on the classpath up to version 3.0. The schema history
table indicates that a migration to version 4.0
# (unknown to us) has already been applied. Instead of bombing out
(fail fast) with an exception, a
# warning is logged and Flyway continues normally. This is useful for
```

```
situations where one must be able to redeploy
# an older version of the application after the database has been
migrated by a newer one.
# true to continue normally and log a warning, false to fail fast with
an exception. (default: true)
# flyway.ignoreFutureMigrations=

# Whether to validate migrations and callbacks whose scripts do not
obey the correct naming convention. A failure can be
# useful to check that errors such as case sensitivity in migration
prefixes have been corrected.
# false to continue normally, true to fail fast with an exception
(default: false)
# flyway.validateMigrationNaming=

# Whether to allow mixing transactional and non-transactional
statements within the same migration.
# Flyway attempts to run each migration within its own transaction
# If Flyway detects that a specific statement cannot be run within a
transaction, it won't run that migration within a transaction
# This option toggles whether transactional and non-transactional
statements can be mixed within a migration run.
# Enabling this means for 'mixed' migrations, the entire script will be
run without a transaction
# Note that this is only applicable for PostgreSQL, Aurora PostgreSQL,
SQL Server and SQLite which all have
# statements that do not run at all within a transaction.
# This is not to be confused with implicit transaction, as they occur
in MySQL or Oracle, where even though a
# DDL statement was run within within a transaction, the database will
issue an implicit commit before and after
# its execution.
# true if mixed migrations should be allowed. false if an error should
be thrown instead. (default: false)
# flyway.mixed=

# Whether to group all pending migrations together in the same
transaction when applying them
# (only recommended for databases with support for DDL transactions).
# true if migrations should be grouped. false if they should be applied
individually instead. (default: false)
# flyway.group=

# The username that will be recorded in the schema history table as
having applied the migration.
# <<blank>> for the current database user of the connection. (default:
<<blank>>).
# flyway.installedBy=

# Rules for the built-in error handler that let you override specific
```

```
SQL states and errors codes in order to
# force specific errors or warnings to be treated as debug messages,
info messages, warnings or errors.
# Each error override has the following format: STATE:12345:W.
# It is a 5 character SQL state (or * to match all SQL states), a
colon,
# the SQL error code (or * to match all SQL error codes), a colon, and
finally
# the desired behavior that should override the initial one.
# The following behaviors are accepted:
# - D  to force a debug message
# - D- to force a debug message, but do not show the original sql state
and error code
# - I  to force an info message
# - I- to force an info message, but do not show the original sql state
and error code
# - W  to force a warning
# - W- to force a warning, but do not show the original sql state and
error code
# - E  to force an error
# - E- to force an error, but do not show the original sql state and
error code
# Example 1: to force Oracle stored procedure compilation issues to
produce
# errors instead of warnings, the following errorOverride can be used:
99999:17110:E
# Example 2: to force SQL Server PRINT messages to be displayed as info
messages (without SQL state and error
# code details) instead of warnings, the following errorOverride can be
used: S0001:0:I-
# Example 3: to force all errors with SQL error code 123 to be treated
as warnings instead,
# the following errorOverride can be used: *:123:W
# Flyway Teams only
# flyway.errorOverrides=

# The file where to output the SQL statements of a migration dry run.
If the file specified is in a non-existent
# directory, Flyway will create all directories and parent directories
as needed.
# Paths starting with s3: point to a bucket in AWS S3, which must
exist. They are in the format
s3:<bucket>(/optionalfolder/subfolder)/filename.sql
# Paths starting with gcs: point to a bucket in Google Cloud Storage,
which must exist. They are in the format
gcs:<bucket>(/optionalfolder/subfolder)/filename.sql
# <<blank>> to execute the SQL statements directly against the
database. (default: <<blank>>)
# Flyway Teams only
# flyway.dryRunOutput=
```

```
# When attempting to get a lock for migrating, the number of attempts
(at 1 second intervals) to make before
# abandoning the migration. Specify -1 to try indefinitely. (default:
50)
# flyway.lockRetryCount=

# JDBC properties to pass to the JDBC driver when establishing a
connection.
# Flyway Teams only
# flyway.jdbcProperties.myProperty=
# flyway.jdbcProperties.myOtherProperty=

flyway.jdbcProperties.maxActive=6
flyway.jdbcProperties.minIdle=2
flyway.jdbcProperties.maxIdle=6
flyway.jdbcProperties.validationQuery=select 1

# Whether Flyway's support for Oracle SQL*Plus commands should be
activated. (default: false)
# Flyway Teams only
# flyway.oracle.sqlplus=

# Whether Flyway should issue a warning instead of an error whenever it
encounters an Oracle SQL*Plus
# statement it doesn't yet support. (default: false)
# Flyway Teams only
# flyway.oracle.sqlplusWarn=

# Your Flyway license key (FL01...). Not yet a Flyway Teams Edition
customer?
# Request your Flyway trial license key st
https://flywaydb.org/download/
# to try out Flyway Teams Edition features free for 30 days.
# Flyway Teams only
# flyway.licenseKey=
```

file remapConfiguration.properties

Questo file ha lo stesso proposito e lo stesso medesimo utilizzo del remapConfiguration.properties.

Tale meccanismo si presta bene ad applicativi di successo che vanno installati e successivamente mantenuti su un numero considerevole di clienti. Si fa un'unica configurazione iniziale con le parti variabili erogate come variabili d'ambiente. La parte di specifica dell'applicativo è trattata una sola volta e può essere replicata facilmente, ed in futuro in maniera sempre più automatica. Gli effort residui per gestione del sistema sono quindi molto snelliti, ed ora di natura prettamente sistemistica, la cui gran parte si è conclusa nel setting iniziale delle variabili d'ambiente.

Rimane sempre una buona idea utilizzare tale meccanismo anche per i nuovi prodotti, in ottica di un più facile dispiegamento al crescere dei clienti.

Vantaggi ci sono non solo in ottica dei *prodotti*, ma anche nei *progetti* su particolari clienti, dotati di un reparto IT strutturato. Anche in particolari contesti, dove il cliente spesso fornisce ambienti di *sviluppo*, *collaudo* e *produzione* in macchine virtuali, è sicuramente vantaggioso utilizzare il meccanismo di re-mapping delle variabili. In pratica si fa una singola configurazione dove tutti le property *critiche* (tipicamente indirizzi di database, MapGuide e documentali, oltre che le credenziali, etc..) che variano da un ambiente all'altro, vengono rimappate su variabili di sistema. Le variabili di sistema sono rese disponibili, con lo stesso nome, su tutti gli ambienti. A cambiare invece è il loro contenuto, contestualizzato di volta in volta. Quindi, di fronte alla configurazione di un unico applicativo, ad ogni aggiornamento, lo si può testare in ambiente di collaudo per poi passarlo in produzione essendo abbastanza certi del successo dell'operazione (dato che eventuali problemi di variabili d'ambiente sono stati risolti preventivamente).