Uso degli Storage da parte di eXtraWay Server

Con la versione Enterprise di eXtraWay Server è stato introdotto l'utilizzo di Storage Esterni.
Essi assolvono a tre diversi compiti:

- Conservazione di informazioni riferite ai singoli utenti, con particolare accezione dei loro eventuali diritti applicativi e funzionali.
- Conservazione degli allegati d'archivio su Storage Server al posto del file system usato di default.
- Conservazione dei record di Metadati al posto dei file XML di default.

Vediamo i seguenti casi e le relative configurazioni.

Informazioni utente

Per la registrazione delle informazioni utente, il server eXtraWay si avvale di un semplice LMBD che trova collocazione in una delle cartelle di servizio di eXtraWay. Anche se si tratta di una componente pressoché fissa, essa è configurata come tutti gli altri Storage, perché possa essere sostituita al bisogno.

Rientra nella configurazione delle `caches` usate da eXtraWay, e non in quella degli `storages`.

Allegati su Storage Server

Record su Storage

Configurazione

La configurazione di `caches` e `storages` utilizzati da eXtraWay si trova nel file `storage.conf.xml` collocato nella cartella...

 `/opt/3di.it/extraway/xwee/conf`

 `<drive>:\3di.it\extraway\xwee\conf`

Esso si compone di diverse sezioni.

caches e storages

Gli Storage utilizzati assolvono a due scopi: Fungere da cache di dati utili ad eXtraWay ovvero essere veri e propri contenitori di metadati o allegati.

Per la prima tipologia, le `caches` esiste un apposita sezione che contiene uno o più elementi `cache`.
Per ciascuno di essi si deve compilare una configurazione:

- '@id': Identificatore che consente ad eXtraWay Server di riconoscere la configurazione di quale `cache` sta accedendo. Obbligatorio
- '@persistor': Identificatore del persistor che si utilizzerà per questa cache. Obbligatorio
- '@expiry_secs': numero di secondi trascorsi i quali il dato conservato nella cache si considera non più valido. Facoltativo. Default: 24 ore

La seconda tipologia, gli `storages`, racchiude uno o più elementi `storage`.
Per ciascuno di essi si deve compilare una configurazione:

- '@id': Identificatore dello storage che si desidera utilizzare. Il legame che si costituisce con questi storage è registrato nei file di configurazione d'archivio. Obbligatorio
- '@persistor': Identificatore del persistor che si utilizzerà per questo storage. Obbligatorio

Si vedano gli esempi per maggior dettaglio.

persistors

È l'elenco di tutti i `persistor` e quindi di ciascuno strumento di storage. La forma assunta da ciascuno di essi, vale a dire le impostazioni in esso contenute sotto forma di elementi ed attributi, varia a seconda della sua tipologia.

Ciascun `persistor` si caratterizza per mezzo di tre attributi:

- '@name': il nome del `persistor` così come utilizzato nella sezione delle cache e degli storages
- '@class': la classe del `persistor` da utilizzare per questo storage, da essa dipendono le configurazioni di dettaglio
- '@policy': il comportamento atteso per il `persistor`, riferito principalmente a come i record debbano essere salvati e così via.

Le classi attualmente disponibili sono:

- **xw::strg::persistor_lmdb**: Persistor basato su LMBD.
- **xw::strg::persistor_mongo**:Persistor basato su MongoDB.

Si vedano gli esempi per maggior dettaglio.

policies

Elenco delle diverse tipologie comportamentali per i vari `persistors`.
Ciascuna `policy` è riconosciuta per mezzo di un attributo '@name' che la qualifica. Al suo interno avremo alcuni elementi atti ad indicare le diverse caratteristiche.

- **update_existing_only**: Quando si richiede l'aggiornamento di un record nello Storage, ma il record con la chiave indicata non esiste, lo Storage deve:
  - Tornare errore in quanto la chiave richiesta non esiste quando il valore dell'attributo è **true**.
  - Consentire il salvataggio del record con la chiave indicata quando il valore dell'attributo è **false**.
- **generate_auto_key**: Quando si procede al salvataggio di un record nuovo senza indicare una chiave per il detto record, lo Storage deve:
  - Tornare errore in quanto la chiave non è stata indicata quando il valore dell'attributo è **false**.
  - Consentire il salvataggio del record generando (e tornando) una chiave univoca quando il valore dell'attributo è **true**.

Il più classico degli esempi è il seguente:

- Per uno storage di tipo `cache` è necessario che la chiave con la quale si registra un record sia sempre esplicitata mentre l'aggiornamento di una chiave inesistente è tollerabile e si trasforma, in pratica, in inserimento.
- Per uno storage di metadati non è ammesso modificare record inesistenti mentre è lecito (ed anzi normale) che la chiave di un nuovo record venga generata dal `peristor` prescelto.
- Per uno storage di allegati si opera esattamente all'opposto: è ammesso aggiornare un allegato non esistente (quindi inserirlo) ma la chiave con la quale lo si genera dev'essere sempre esplicitata in quanto viene calcolata da eXtraWay.

Si vedano gli esempi per maggior dettaglio.

Esempi

Di seguito un esempio “commentato” di file di configurazione.

```
<?xml version="1.0"?>
<config>
   <policies>
      <policy name="default">
         <!-- Politica di base, minimale.
              Si aggiorna solo quello che esiste, ma una chiave non dichiarata viene generata.
         -->
            <update_existing_only>true</update_existing_only>
            <generate_auto_key>true</generate_auto_key>
      </policy>

      <policy name="db_mode_policy">
         <!-- Politica per Storage di METADATI.
              Si aggiorna solo quello che esiste, ma una chiave non dichiarata viene generata.
         -->
         <update_existing_only>true</update_existing_only>
         <generate_auto_key>true</generate_auto_key>
      </policy>

      <policy name="media_mode_policy">
         <!-- Politica per Storage di ALLEGATI.
              Si opera in 'upsert' ma la chiave va sempre dichiarata.
         -->
         <update_existing_only>false</update_existing_only>
         <generate_auto_key>false</generate_auto_key>
      </policy>

      <policy name="lmdb_cache_policy">
         <!-- Politica per CACHE.
              Si opera in 'upsert' ma la chiave va sempre dichiarata.
         -->
         <update_existing_only>false</update_existing_only>
         <generate_auto_key>false</generate_auto_key>
      </policy>
   </policies>

   <persistors>
```

```
      <!-- Persistor per la CACHE. Usa la 'lmdb_cache_policy' -->
      <persistor name="eXtraWayUsers_persistor" class="xw::strg::persistor_lmdb"
policy="lmdb_cache_policy">>
         <path>/var/lib/3di.it/usersData</path>
      </persistor>

      <persistor name="dcwData" class="xw::strg::persistor_mongo" policy="db_mode_policy">
         <host>localhost</host>
         <dbname>dcwRecords</dbname>
            <!--
             see mongo doc: https://docs.mongodb.com/manual/reference/write-concern/
         -->
         <write_concern w="1" j="true" wtimeout="0" />
      </persistor>
      <persistor name="dcwMedia" class="xw::strg::persistor_mongo" policy="media_mode_policy">
         <host>localhost</host>
         <dbname>content_storage</dbname>
            <!--
             see mongo doc: https://docs.mongodb.com/manual/reference/write-concern/
         -->
         <write_concern w="1" j="true" wtimeout="0" />
      </persistor>
   </persistors>

   <caches>
      <cache id="usersData" persistor="eXtraWayUsers_persistor" expiry_secs="1800"/>
   </caches>

   <storages>
      <storage id="dcwData" persistor="dcwData"/>
      <storage id="dcwMedia" persistor="dcwMedia"/>
    </storages>
</config>
```