itaSavePoint e itaSavePointFactory

Queste due classi permettono l'applicazione di aggiornamenti e la loro storicizzazione.

itaSavePoint

Un savepoint rappresenta uno snapshot differenziale fra la cartella di installazione di itaEngine e la workingDir dell'updater. Nello specifico all'interno di un savepoint vengono salvati tutti i file aggiunti, cancellati o modificati raffrontando la cartella di installazione di itaEngine e la cartella su cui viene caricata la versione aggiornata dell'updater. Di seguito verranno trattati i soli metodi che dovranno essere invocati direttamente ignorando i rimanenti per evitare confusioni nell'utente finale:

applyDiff()

Questo metodo permette di applicare tutte le modifiche di un savepoint, di fatto portando la cartella di installazione di itaEngine allo stesso stato della workingDir dell'updater al momento della creazione del savepoint. Nel momento in cui un savepoint viene applicato viene salvato come il savepoint attualmente in uso.

revertDiff()

Questo metodo funziona al contrario rispetto ad applyDiff e se invocato riporta la cartella di installazione di itaEngine allo stato precedente alla creazione del savepoint.

getSavePointDiff()

Questo metodo restituisce un array multidimensionale strutturato nel seguente modo:

- ['A']: Array che contiene i file aggiunti dal savepoint. L'array segue una struttura chiave>valore dove la chiave è l'hash del path dei file aggiunti mentre il valore è il path stesso.
- ['M']: Array che contiene i file modificati dal savepoint. L'array segue una struttura chiave>valore dove la chiave è l'hash del path dei file modificati mentre il valore è il path stesso.
- ['D']: Array che contiene i file cancellati dal savepoint. L'array segue una struttura chiave>valore dove la chiave è l'hash del path dei file cancellati mentre il valore è il path stesso.

```
array(
  'A' => array(),
  'M' => array(
    '85136c79cbf9fe36bb9d05d0639c70c265c18d37' => 'Start.php',
    '00ea1da4192a2030f9ae023de3b3143ed647bbab' =>
'apps/CityBase/cwbBgdTipdoc.php'
  ),
```

```
  'D' => array()
)
```

getSavePointHash()

Questo metodo restituisce un hash sha1 dei file toccati dal savepoint.

setSavePointMetaData()

Permette di aggiungere al savepoint dei metadati. Questo metodo accetta in input qualsiasi cosa, ma si consiglia il passaggio di un array associativo o di un stdClass.

getSavePointMetaData()

Restituisce i metadati del savepoint nella stessa forma in cui sono stati passati dal metodo setSavePointMetaData().

---

itaSavePointFactory

Questa classe è quella che va usata per la creazione o per l'apertura di savepoint. Di seguito vengono riportati i metodi esposti che la compongono:

getInstance()

Metodo statico che permette di istanziare la classe. Questa segue il pattern singleton, per essere istanziata è quindi necessario usare la seguente sintassi:

```
$savePointFactory = itaSavePointFactory::getInstance();
```

getCurrentSavePoint()

Il metodo restituisce l'istanza del savepoint attualmente selezionato. Se nessun savepoint dovesse risultare selezionato (ad esempio perché non ne è mai stato applicato uno) verrà restituito l'ultimo savepoint esistente. Se non dovessero essere presenti savepoint verrà sollevata un'eccezione.

getPreviousSavePoint()

Il metodo restituisce l'istanza del savepoint precedente rispetto a quello attualmente selezionato. Se non dovesse risultare un savepoint precedente a quello attuale verrà sollevata un'eccezione.

getNextSavePoint()

Il metodo restituisce l'istanza del savepoint successivo rispetto a quello attualmente selezionato. Se non dovesse risultare un savepoint successivo a quello attuale verrà sollevata un'eccezione.

createNewSavePoint()

Il metodo crea un nuovo savepoint successivo a quello attualmente selezionato. Il sistema prevede una catena e non una struttura ad albero, quindi alla creazione di un savepoint eventuali savepoint successivi a quello attualmente in uso devono venir eliminati. E' previsto un sistema di sicurezza per cui per se si vuole creare un savepoint a partire da quello che non è l'ultimo savepoint disponibile è necessario passare il parametro $forceDelete = true, in caso contrario verrà sollevata un'eccezione.

Esempi d'uso

Per com'è strutturato il programma nel caso si volesse navigare nella catena dei savepoint sarà necessario applicare ogni savepoint in maniera sequenziale.

Esempio: tornare indietro di tre passi

```
$savePointFactory = itaSavePointFactory::getInstance();
$savePoint = $savePointFactory->getCurrentSavePoint();
$savePoint->revertDiff(); //Ho annullato l'ultimo passo
$savePoint = $savePointFactory->getCurrentSavePoint();
$savePoint->revertDiff(); //Ho annullato il penultimo passo
$savePoint = $savePointFactory->getCurrentSavePoint();
$savePoint->revertDiff(); //Ho annullato il terzultimo passo
```

Esempio: Andare avanti di tre passi

```
$savePointFactory = itaSavePointFactory::getInstance();
$savePoint = $savePointFactory->getNextSavePoint();
$savePoint->applyDiff(); //Ho applicato il passo successivo
$savePoint = $savePointFactory->getNextSavePoint();
$savePoint->applyDiff(); //Ho applicato il passo successivo, sono avanti di
due passi
$savePoint = $savePointFactory->getNextSavePoint();
$savePoint->applyDiff(); //Ho applicato il passo successivo, sono avanti di
tre passi
```