

Design-time Data Set Caching

Design Specification

Version 1.0: Dec 21 2005

Abstract

This is the design specification of BIRT 2.0 Project Design-time Data Set Caching, which is part of 2.0 BPS 7 (Data Engine Performance).

Document Revisions

Version	Author	Date	Description of Changes
Draft 1	Gary Xue	Nov 21, 2005	First draft
Draft 2	Gary Xue	Nov 30, 2005	Updates to DtE API
Version 1.0	Gary Xue	Dec 21, 2005	Updated DtE API; minor updates to DE API

Contents

- 1. Introduction.....3**
- 2. Features and Requirements3**
 - 2.1 User-Configurable Cache Setting (Priority: High)3
 - 2.2 Automatic Invalidation and Update (Priority: High)3
 - 2.3 Visual Indication of Cache Usage (Priority: Medium).....3
- 3. Data Engine Support and API Change3**
 - 3.1 Overview4
 - 3.2 API Changes4
 - 3.2.1 *DataEngine Class*.....4
 - 3.3 Effects of Cache Usage5
- 4. Report Designer UI Changes6**
 - 4.1 Caching Preferences6
 - 4.2 Cache Invalidation.....6
- 5. Engine API Changes.....6**

1. Introduction

While designing a report, the developer specifies a data set to report on. If the number of records in the data set is large, then the iterative process of making changes to a report followed by previewing those changes slows down because the large data set has to be fetched for each preview operation. This slowdown can hinder quick report design and adversely affect the developer experience and productivity associated with BIRT. This project allows a report developer to limit the number of records that are fetched for a data set during the development process, and to allow those results to be cached as much as possible.

Design-time Data Set Caching is a sub feature of BIRT 2.0 BPS 7 (Data Engine Performance and Scalability).

2. Features and Requirements

2.1 User-Configurable Cache Setting (Priority: High)

The report designer should be able to specify, on a per-data set basis, whether the design-time cache is enabled, and the number of data rows to cache.

2.2 Automatic Invalidation and Update (Priority: High)

Cached data should be invalidated whenever the definition of a data set is changed. The data set cache will be repopulated the first time that the report is previewed after the change.

The report designer can also manually invalidate the cached data.

3. Model API Change

The per-data set cache setting is stored with the Data Set definition in the report design. A new model property, *CacheRowCount*, is introduced to the base data set model. This property is of type integer, and can take the following values:

0 (default): Design-time data cache is disabled for this data set.

Positive value: Enables data cache for this data set. The value specifies the number of data rows that should be read and cached for this data set when caching is in effect.

3.1 DataSetHandle Class

org.eclipse.birt.report.model.api.DataSetHandle class is enhanced with two additional methods to access the new *CacheRowCount* property.

```
/**
 * Gets the data cache setting for this data set.
 * @returns 0 if data cache is disabled for this data set (default).
 * If not 0, data cache is enabled for this data set, and the value
 * is the number of data rows that will be cached and used for
 * this data set when caching is in effect at design time.
```

```
*/
public int getCachedRowCount();

/**
 * Changes the data cache setting for this data set.
 * @param count The number of data rows that will be cached and used
 * for this data set when caching is in effect at design time. To
 * disable caching for this data set, set count to 0.
 */
public void setCachedRowCount( int count );
```

4. Data Engine Support and API Change

4.1 Overview

The Data Engine manages lifetime of cached data set data. It provides a few enhanced APIs to allow Engine and Chart components to specify how data set cache should be used.

Cached data set data is only valid for duration of the Eclipse session in which the BIRT report designer is run. All cached data are cleared upon JVM termination.

4.2 API Changes

4.2.1 DataEngineContext Class

Class *org.eclipse.birt.data.engine.api.DataEngineContext* will have these additional methods:

```
public void setCacheOption( int option, int cacheCount );
public int getCacheCount ( );
public int getCacheOption ( );
```

The Data Engine consumer calls *setCacheOption* to inform Data Engine how data set caching should be used by this instance of Data Engine. The *option* parameter can take any of these pre-defined constants:

DataEngineContext.CACHE_USE_DEFAULT :

Data Set Caching is enabled/disabled on a per-request basis. The ApplicationContext specified for each report query determines if caching is enabled for that query. (see next section). Each Data Set's *CachedRowCount* ROM definition is used to determine the cache count.

DataEngineContext.CACHE_USE_DISABLE:

Data Set Caching is disabled for this Data Engine instance, regardless of what the data set cache option setting or ApplicationContext setting is.

DataEngineContext.CACHE_USE_ALWAYS:

Data Set Caching is always enabled for all data sets and all queries. The *cacheCount* parameter defines the data set cache count to be applied to all data sets.

The *getCacheCount()* and *getCacheOption()* methods return the settings provided in the last *setCacheOption* call.

4.2.2 Using Application Context to Specify Cache Option

An application context (AppContext), which is a map of properties, is passed to Data Engine for each query in the following DataEngine method:

```
public IPreparedQuery prepare( IQueryDefinition querySpec, Map appContext );
```

The consumer of Data Engine may pass in a special AppContext key/value pair to enable data set caching for the query being prepared. The key is string "org.eclipse.birt.data.engine.dataset.cache.option", and the value should be String "true"

The Data Engine pays attention to the AppContext setting only when it is being initialized with the CACHE_USE_DEFAULT setting (see previous section).

4.2.3 DataEngine class

A new method is added to class org.eclipse.birt.data.engine.api.DataEngine:

```
public void clearCache( IBaseDataSourceDesign dataSource,  
                        IBaseDataSetDesign dataSet );
```

This method clears the cache for the specified data set, causing subsequent query execution to re-read data for that data set from the outsource.

4.3 Effects of Cache Usage

The Data Engine caches both the metadata and the data rows (up to the configured cache size limit) of a data set. If a data set cache is used, no actual connection to the data source will be made.

The Data Engine does not cache computed columns. These columns are re-computed using cached data if the cache is used.

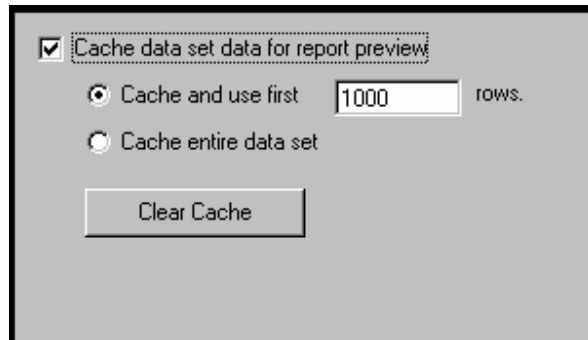
Some but not all event scripts associated with the data set/ data source are executed if cache data is used. Specifically,

- The Data Source associated with the data set will not be opened. None of the scripts associated with the data source will be executed.
- The Data Set's *beforeOpen*, *afterOpen*, *beforeClose* and *afterClose* scripts are executed. The *onFetch* script is also executed for each data row that's read from the cache.
- If the Data Set is a Script Data Set, the *describe*, *open*, *fetch* and *close* scripts are not executed.

5. Report Designer UI Changes

5.1 Caching Preferences

The Data Set Editor dialog will include an additional node “Caching” in the left pane. The following options are presented to the user when Caching is selected



The radio buttons are enabled only when the checkbox is checked. The default state of the checkbox is unchecked (i.e., caching is disabled). The default selection of the radio buttons is to cache and use the first 500 rows.

The Data Set Editor should store the caching setting in Eclipse preferences.

5.2 Cache Invalidation

Cache invalidation for a data set is done by the Designer by calling Data Engine's *clearCache* method. This should happen in any of these events

- Whenever the user saves any modifications to the name, query or data source of a data set.
- When the user selects the “Clear Cache” button.
- When the user disables caching for a data set that is previously cached.

Cache invalidation is not necessary if the user only updates the Computed Columns, Filters or Scripts definition of a data set.