

Dynamic Querying (QBF) using the dbANYWHERE PRO API:

The MultiView.restart () and RelationView.restartMultiView() methods have been overloaded. To provide dynamic querying functionality. The methods are as follows:

MultiView.restart (string sqlWhereClause)	- new to dbANYWHERE Server
RelationView.restartMultiView (string sqlWhereClause)	- new in dbANYWHERE Server
MultiView.restart ()	
RelationView.restartMultiView ()	

Each method now takes a string parameter for a SQL WHERE clause. Passing in a SQL WHERE clause will replace the existing WHERE clause of the SQL statement for the associated RelationView or MultiView object (if a WHERE clause exists). If the SQL statement does not contain a WHERE clause the new WHERE clause will be added to the SQL statement. Once the WHERE clause has been added/updated the SQL statement will be re-executed, using the new WHERE clause.

Once the WHERE clause has been substituted using the overloaded methods, it is part of the RelationView and MultiView Objects, hence, subsequent calls to restart or restartMultiView even without specifying a string WHERE clause will re-execute the LAST version of the SQL statement executed. To return to a SQL statement without a WHERE clause call the restart methods passing an empty string (to return to the default behavior of Visual Café Pro generated Applets).

With these new methods, a new RelationView, MultiView, and / or Request objects for each query is no longer required. A single RelationView, MultiView, and / or request object can be reused receptively changing only the SQL WHERE clause and restarting the View.

Restarting RelationView Methods - assumes a RelationView object is defined

These methods are best suited for use with Visual Café Pro dbAware applets and applications generated via the dbAware Project Wizards

Use this method to re-execute the previous SQL statement with a new SQL WHERE clause, for an existing RelationView object. dbANYWHERE does not validate WHERE clauses, so be sure to construct a valid SQL WHERE clause. Below is one example of how to construct a varying WHERE clause.

```
void relationViewQueryByForm_button_Clicked ( Event event ) {  
    String sqlWhereClause = "where columnName like " + queryTextFieldName.getText () + "%";  
    try {  
        relationViewName.restartMultiView ( sqlWhereClause );  
    }  
    catch ( symjava.sql.SQLException e ) {  
        System.out.println( e.getMessage () );  
    }  
}
```

Use this method to clear the WHERE clause within the SQL statement of an existing RelationView object and return to the initial state of Visual Café Pro generated applets.

```
void restartRelationView_button_Clicked ( Event event ) {  
    String sqlWhereClause = "";  
    try {  
        relationViewName.restartMultiView ( sqlWhereClause );  
    }  
    catch ( symjava.sql.SQLException e ) {  
        System.out.println ( e.getMessage () );  
    }  
}
```

This is an example of the default Restart method generated by the Visual Café Pro Project Wizards. Executing this method will re-execute the previous SQL statement for an existing RelationView object, using the last supplied WHERE clause, without having to re-specify it.

```
void defaultRestartRelationView_button_Clicked ( Event event ) {  
    //{{CONNECTION  
    try {  
        relationViewName.restartMultiView ();  
    }  
    catch ( symjava.sql.SQLException e ) {  
        System.out.println ( e.getMessage () );  
    }  
    //}}  
}
```

Restarting MultiView Methods - assumes a MultiView object is defined

Use this method to re-execute the previous SQL statement with a new SQL WHERE clause, for an existing MultiView object. dbANYWHERE does not validate WHERE clauses, so be sure to construct a valid SQL WHERE clause. Below is one example of how to construct a varying WHERE clause.

```
void mutliViewQuerbyForm_button_clicked ( Event event ) {
    String sqlWhereClause = "where vrchr like '" + queryTextFieldName.getText () + "%'";
    try {
        multiViewName.restart ( sqlWhereClause );
    }
    catch ( symjava.sql.SQLException e ) {
        System.out.println ( e.getMessage() );
    }
}
```

Use this method to clear the WHERE clause within the SQL statement of an existing MultiView object.

```
void restartMultiView_button_clicked ( Event event ) {
    String sqlWhereClause = "";
    try {
        multiViewName.restart ( sqlWhereClause );
    }
    catch ( symjava.sql.SQLException e ) {
        System.out.println ( e.getMessage () );
    }
}
```

This is an example of the default Restart method. Executing this method will re-execute the previous SQL statement for an existing MultiView object, using the last supplied WHERE clause, without having to re-specify it.

```
void defaultRestartMultiView_button_Clicked ( Event event ) {
    try {
        multiViewName.restart ();
    }
    catch ( symjava.sql.SQLException e ) {
        System.out.println ( e.getMessage () );
    }
}
```

Restarting MultiView Via RelationView Methods - assumes a RelationView object is defined, which can then be used to gain access to the MultiView object (the reverse is not true, MultiView object can not gain access to RelationView objects)

Use this method to re-execute the previous SQL statement with a new SQL WHERE clause, for an existing RelationView object, accessing the underlying MultiView object. dbANYWHERE does not validate WHERE clauses, so be sure to construct a valid SQL WHERE clause. Below is one example of how to construct a varying WHERE clause.

```
void multiViewViaRelationViewQuerbyForm_button_clicked ( Event event ) {
    String sqlWhereClause = "where vrchr like " + queryTextFieldName.getText () + "%";
    try {
        symantec.itools.db.pro.MultiView multiViewName = relationViewName.getMultiView ();
        multiViewName.restart ( sqlWhereClause );
    }
    catch ( symjava.sql.SQLException e ) {
        System.out.println ( e.getMessage () );
    }
}
```

Use this method to clear the WHERE clause within the SQL statement of an existing RelationView object, accessing the underlying MultiView object.

```
void restartMultiViewViaRelationView_button_clicked ( Event event ) {
    String sqlWhereClause = "";
    try {
        symantec.itools.db.pro.MultiView multiViewName = relationViewName.getMultiView ();
        multiViewName.restart ( sqlWhereClause );
    }
    catch ( symjava.sql.SQLException e ) {
        System.out.println ( e.getMessage () );
    }
}
```

This is an example of the default Restart method. Executing this method will re-execute the previous SQL statement for an existing RelationView object, accessing the underlying MultiView object, using the last supplied WHERE clause, without having to re-specify it.

```
void defaultRestartMultiViewViaRelationView_button_Clicked ( Event event ) {
    try {
        symantec.itools.db.pro.MultiView multiViewName = relationViewName.getMultiView ();
        multiViewName.restart ();
    }
    catch ( symjava.sql.SQLException e ) {
        System.out.println ( e.getMessage () );
    }
}
```