SAP Business One UI API Performance Tips

Applies to:

SAP Business One SDK UI API use for development of add-ons.

Summary

This article provides several tips that will help you improve the performance of add-on programs developed using the UI API of SAP Business One SDK.

Author(s): SAP Business One Solution Architects (upgrade of an article from Gali Moualem for 6.5 version)

Company: SAP

Created on: February 2007

© 2006 SAP AG

Table of Contents

© 2006 SAP AG

Events Filtering

By default, add-ons connected to the UI API get all the events triggered by the SAP Business One application when the corresponding event handlers have been defined. Usually, most of these events are not necessary for your add-on program. In addition, handling all of them has a significant negative impact on the performance of your add-on program.

To improve your add-on performance, we recommend using the **`EventFilters`** object to filter out events (i.e. Item and Menu events) that are not handled by the add-on program.

Please note though that concrete performance improvements will depend on the implementation of the Add-On as well – so there's no general rule that allows calculating the improvement when e.g. half of the available event types would be filtered. It may happen that the time it takes until a certain action is cut down to a quarter; obviously it may not improve the performance much when the events that are not included in the filter won't occur anyway…

For more information, see the *08.EventsFilter* UI API sample provided together with the SDK and the eLearning chapter about events.

It is possible to change the event filters "on the fly" – and e.g. to declare that you don't want to receive **any** event by setting an empty `EventFilters` object temporarily – e.g. while you are applying major changes – and later restore the previous state. This is especially helpful when working with B1 system forms – please check carefully that your add-on doesn't need any events it would receive otherwise!

User Defined Forms creation

Creating or updating forms dynamically in run time is very costly to the performance of your add-on program.

To improve performance, we recommend that you create your forms by using xml format.
To create the XML file representing your form you have 3 options:

1. Use the Screen Painter to design your forms.
2. Initially create a form by using UI API objects (`Form`, `Item`,…) in your code and then call the `Form.GetAsXML` method to get the resource file representing your form in XML format. After that load your form always with XML methods.
3. Directly work with XML editors, not recommended. Please be careful with this option since you could create invalid XML files – and it may be quite difficult to find the issue(s).

Once you have an XML file, you can update or load the form from the resource file by using one of the following methods proposed by the UI API in your add-on:

- `XmlData` property of the `FormCreationParams` object. Recommended to be used to load forms.
  For more information, see eLearning presentation about forms.
- `LoadBatchActions` method of the `Application` object. Must be used to update forms.
  For more information, see *04.WorkingWithXML* sample provided together with the SDK.

Forms freezing

When you make visual changes to a form such as entering information in items, adding items, adding rows to a matrix,… we recommend freezing the form by calling the `Form.Freeze(True)` method. After you made all the changes you want, release the form by calling the `Form.Freeze(False)`.

Please note that freezing the form will not keep UI API from sending events to add-ons.

© 2006 SAP AG

UI API objects management optimization

Each UI API action (e.g. setting a property for a particular item) will cause an inter-process call between your add-on and the B1 application, that's why you should reduce calls to UI API to the absolutely required minimum.

The first step to reduce the number of calls is to use XML in all actions where the XML format is accepted: `Menus` and `Forms` actually, cf. section User Defined Forms creation.

A second step is to try to keep variables for references you have to reuse several times in the same method (`Form`, `Items`,...). For example while working with matrixes you will for example fill different lines for the same column; in this case you could keep the cells of each column (`oMatrix.Columns.Item("A").Cells`) and reuse it for all rows.

Use DataSources

Updating and getting values of form items is more efficient when you access them via their bound `DataSource` object (available object types are: `UserDataSource`, `DBDataSource` or `DataTable`) instead of directly accessing the item `Value` property. This is especially true for matrixes containing many items, but should be applied in general.

Please note that on B1 system forms the values in the datasources cannot be changed!!!

`UserDataSources`: Use the `Value` property of the `UserDataSource` bound to an `Item` or a `Column` in a `Matrix` to set/get the values.

`DBDataSources`: Use the `SetValue` and `GetValue` methods of the `DBDataSource` bound to an `Item` or a `Column`. Applies to DataTable as well.

For the matrixes the `FlushToDataSource` and `LoadFromDataSource` methods synchronize all data between the matrix and the `DataSources`. The same mechanism can be used line by line with `GetLineData(row)` and `SetLineData(row)`.

- To update matrix data, when working with `UserDataSources`, perform the following recursively on the matrix lines:

```
GetLineData(Row)
UserDataSource.Value = Value
SetLineData(Row)
```

  `DataTables`: Information of `DataTable` is always the same we can see in the `Grid`, there is no need to use any method to synchronize between the `Grid` and its `DataTable`. To set/get the values we can use the `GetValue` and `SetValue` methods.

© 2006 SAP AG

- To update `CheckBox`, `OptionButton`, and `Folder` items, use the `ValOn/ValOff` properties of their bound `UserDataSource`.

  ```
  UserDataSouce.Value = Item.Specific.ValOn/ValOff
  ```

  When working with valid values and `Grids`/`DataTables` – you have to specify the valid values for the corresponding column in the `Grid` object(s) the `DataTable` is attached to.

For more information, see the *MatrixAndDataSource* Sample in SDK samples and also the eLearning about DataBinding.

Use the EventLogger tool

In the design phase of your add-on and before deciding which events you will capture to launch some actions in your add-ons we recommend you to use the EventLogger tool. This tool gives you the events fired by B1 after some user actions, you can then decide the ones you need taking preferentially the events that are fired not too often.

Use the .NETProfiler tool

In order to identify the calls that are expending more time and optimize your code we recommend you to use the .NET Profiler tool, this tool is provided inside the SAP Business One Test Environment tools package given as a free source code here in SDN.

Just read the detailed article Add-ons performance analysis using .NET Profiler tool.

© 2006 SAP AG

Related Content

The following documents list can help you on different points of the UI API usage:

- UI API Handling Grid Type Data with SDK
- Working With Data Sources in the SAP Business One UI API
- Loading XML elements to System Forms
- eLearning material

© 2006 SAP AG

Copyright

© Copyright 2006 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.