



Utility Classes In Data Mapper

Version 6.6
Release Date April 19, 2017

343 West Erie, Suite 430
Chicago, IL 60654, USA
Phone: (312) 229-1727 x111
Fax: (312) 229-1736

DOCUMENT INFORMATION

Utility Class in Data Mapper

Adeptia Support Information

For support queries, please contact us at support@adeptia.com.

Access the Adeptia Web site at the following URL:

www.adeptia.com

Copyright

© 2000-2017 Adeptia, Inc. All rights reserved.

Trademarks

Adeptia™ is a trademark of Adeptia, Inc. All other trademarks and registered trademarks are the property of their respective owners.

Confidentiality

This document is the confidential and proprietary information of Adeptia. The information set forth herein represents the confidential and proprietary information of Adeptia. Such information shall only be used for the express purpose authorized by Adeptia and shall not be published, communicated, disclosed or divulged to any person, firm, corporation or legal entity, directly or indirectly, or to any third person without the prior written consent of Adeptia.

Disclaimer

Adeptia, Inc. provides this publication "as is" without warranty of any kind, either express or implied. In no event shall Adeptia be liable for any loss of profits, loss of business, loss of use or data, interruption of business, indirect, special, punitive, incidental, or consequential damages of any kind.

No part of this work should be reproduced in any form or by any means—graphic, electronic, or mechanical—including photocopying, recording, taping, or storage in an information retrieval system, without prior written permission of Adeptia Inc.

This publication is subject to replacement by a later edition. To determine if a later edition exists, contact

www.adeptia.com.

TABLE OF CONTENTS

Preface	4
Target Audience	4
Other resource materials	4
<i>How is this guide organized?</i>	4
<i>Conventions</i>	4
Typographical conventions	4
Graphical conventionsabo	5
<i>Contacts/Reporting problems</i>	5
Sales	5
Support	5
Latest updates and information.....	5
Adeptia Web site.....	5
MapperUtilityClass in Data Mapper	6
CharAt(,).....	6
equalsIgnoreCase(,)	6
matches(,)	6
replaceAll(, ,).....	6
replaceFirst(, ,).....	7
toLowerCase().....	7
toUpperCase().....	7
getCurrentDate().....	7
abortMapping()	8
stringToHex().....	8
newLineBefore()	8
newLineAfter()	8
newLineBeforeAndAfter()	8
getRandomNumber().....	9
getNextorPreviousDate()	9
getJulian7fromDate(,)	9
getJulian7IntoDate(,).....	9
dateAdd(, , , , , ,)	9
executeDbUpdateQuery(,)	10
executeDbInsertQuery(,).....	10
executeDbDeleteQuery(,)	10
encodefileAsBase64(,).....	11
invokeREST(, , , , , ,)	11
GlobalMethodCall in Data Mapper	12
getAge()	12
getDaysLeft().....	12
getAccountType().....	12

PREFACE

This document gives you a detailed information about the various Utility Classes that you can use in Data Mapper.

Target Audience

This document is intended for all users of Adeptia Suite, who wants to gain more information about the MapperUtilityClass and GlobalMethodCall of Global Methods in Data Mapper.

Other resource materials

The following other resource materials are available:

Title	Description
Developer Guide	This document covers a detailed description of all activities and services of Adeptia Suite that are available to a developer. It acts as a guideline to use these services seamlessly and use them in a design environment using Adeptia Suite.

HOW IS THIS GUIDE ORGANIZED?

This guide is organized into the following sections:

Section	Description
<i>Preface</i>	Introduction to this document
<i>MapperUtilityClass in Data Mapper</i>	Description about the various methods of the MapperUtilityClass class
<i>GlobalMethodCall in Data Mapper</i>	Description about the various methods of the GlobalMethodCall class

CONVENTIONS

The following tables list the various conventions used in Adeptia documentation. We follow these conventions to help you quickly and easily identify particular elements, processes, and names that occur frequently in documents.


Typographical conventions

This guide uses the following typographical conventions:

Convention	Description
<i>Italic text</i>	Indicates a reference or the title of a publication.
Monospaced text	Indicates code examples, syntax, or system messages.
Monospaced bold text	Indicates system commands that you enter.

Graphical conventions

This guide uses the following graphical conventions:

Convention	Description
	Indicates additional information that may be of interest to the reader.

CONTACTS/REPORTING PROBLEMS

These sections present contact information for a variety of situations.

Sales

In case of any sales queries, please contact us at sales@adeptia.com.

Support

For support queries, please contact us at support@adeptia.com.

Latest updates and information

For the latest updates and information, please visit us at www.adeptia.com.

Adeptia Web site

Access the Adeptia Web site at the following URL:
www.adeptia.com

MAPPERUTILITYCLASS IN DATA MAPPER

This section contains the various methods that are a part of the MapperUtilityClass class. The tables below shows the various method names, descriptions, along with their syntax.

Method Name	CharAt(,)
Description	Returns the character at the specified index. This method accepts 2 parameters, first one is the data string and second is the integer index.
Syntax	<code>java:MapperUtilityClass.charAt('Adeptia', 3)</code>

Method Name	equalsIgnoreCase(,)
Description	This method compares one String to another String, ignoring case considerations and returns true or false. Two strings are considered equal ignoring case if they are of the same length, and corresponding characters in the two strings are equal ignoring case.
Syntax	<code>java:MapperUtilityClass.equalsIgnoreCase('adeptia', 'ADEPTIA')</code>

Method Name	matches(,)
Description	Checks if the given data string matches the given regular expression. This method accepts 2 parameters, first one is the data string and second is the string regular expression.
Syntax	<code>java:MapperUtilityClass.matches('Welcome to Adeptia.com', "(.*)Adeptia(.*)")</code>

Method Name	replaceAll(, ,)
Description	Replaces all the occurrences of particular string from given data string. This method accepts 3 parameters, first one is the data string and second is the value to be replaced and third parameter is the replaced value.
Syntax	<code>java:MapperUtilityClass.replaceAll('beauty', 'y'</code>

Method Name	<code>replaceAll(, ,)</code>
	<code>, 'iful')</code>

Method Name	<code>replaceFirst(, ,)</code>
Description	Replaces the first occurrence of particular string from given data string. This method accepts 3 parameters, first one is the data string, second is the value to be replaced and third parameter is the replaced value.
Syntax	<code>java:MapperUtilityClass.replaceFirst('adeptia', 'a', 'A')</code>

Method Name	<code>toLowerCase()</code>
Description	Converts the data string to lower case. This method accepts 1 input parameter which is the data string.
Syntax	<code>java:MapperUtilityClass.toLowerCase('ADEPTIA')</code>

Method Name	<code>toUpperCase()</code>
Description	Converts the data string to upper case. This method accepts 1 input parameter which is the data string.
Syntax	<code>java:MapperUtilityClass.toUpperCase('adeptia')</code>

Method Name	<code>getCurrentDate()</code>
Description	Gets the current Date and Time from the server and gives the Date and Time as output in desired Java Time Zone. This method accepts 2 parameters, first one accepts the Date/Time format in data string and second one accepts the Time Zone in data string.
Syntax	<code>java:MapperUtilityClass.getCurrentDate('yyyy-MM-dd HH:mm:ss', 'time-zone')</code> For Example- <code>java:MapperUtilityClass.getCurrentDate('2013-11-29 02:20:07', 'UTC-10:00')</code>

Method Name	abortMapping()
Description	Aborts the mapping. This method accepts one string type parameter only.
Syntax	<code>java:MapperUtilityClass.abortMapping("Error message")</code>

Method Name	stringToHex()
Description	Converts the string to hexadecimal representation. This method accepts 1 input parameter which is the data string.
Syntax	<code>java:MapperUtilityClass.stringToHex('Adeptia')</code>

Method Name	newLineBefore()
Description	Inserts a newline just before start of the data. This method accepts 1 input parameter which is the data string.
Syntax	<code>java:MapperUtilityClass.newLineBefore('Adeptia')</code>

Method Name	newLineAfter()
Description	Inserts a newline at the end of the data. This method accepts 1 input parameter which is the data string.
Syntax	<code>java:MapperUtilityClass.newLineAfter('Adeptia')</code>

Method Name	newLineBeforeAndAfter()
Description	Inserts a newline just before start of the data as well as at the end of the data. This method accepts 1 input parameter which is the data string.
Syntax	<code>java:MapperUtilityClass.newLineBeforeAndAfter('Adeptia')</code>

Method Name	<code>getRandomNumber()</code>
Description	Generates the random number. It accepts zero input parameter.
Syntax	<code>java:MapperUtilityClass.getRandomNumber()</code>

Method Name	<code>getNextorPreviousDate()</code>
Description	Adds or subtracts the number of days to the current date based on the value passed. If value passed is positive integer then it will add the passed number of days to the current date and vice versa. This method accepts 1 input parameter which is the integer value of number of days.
Syntax	<code>java:MapperUtilityClass.getNextorPreviousDate(2)</code>

Method Name	<code>getJulian7fromDate(,)</code>
Description	Converts the Date to the Julian date format CCYYDDD. This method accepts 2 parameters, first one is the date string and second is the format of the date passed.
Syntax	<code>java:MapperUtilityClass.getJulian7FromDate('dd-MM-yyyy', 'yyyddd')</code>

Method Name	<code>getJulian7IntoDate(,)</code>
Description	Converts the Julian date CCYYDDD to the date format. This method accepts 2 parameters, first one is the Julian date string and second is the format of the date.
Syntax	<code>java:MapperUtilityClass.putJulian7IntoDate('2008354', 'yyyMMdd')</code>

Method Name	<code>dateAdd(, , , , , ,)</code>
Description	This method increases the date and timestamp by the values specified in the parameters passed. The first 3 parameters in this function should be of string format and the rest of the 6 parameters should be integer in value. The first parameter value is date-time in string format. The second parameter value is in string format, it's the format in which you have entered the previous

Method Name**dateAdd(, , , , , , , ,)**

parameter value. The third parameter value is in string format, it's the date-time in which you want the output. The fourth parameter value is the integer value by which you want to increase the year. The fifth parameter value is the integer value by which you want to increase the month. The sixth parameter value is the integer value by which you want to increase the day. The seventh parameter value is the integer value by which you want to increase the hour. The eighth parameter value is the integer value by which you want to increase the minutes. The ninth parameter value is the integer value by which you want to increase the seconds.

Syntax

```
java:MapperUtilityClass.dateAdd('2013-11-27  
HH:mm:ss', 'yyyy-MM-dd HH:mm:ss', 'MM-dd-yyyy  
HH:mm:ss', yyyy, MM, dd, hh, mm, ss)
```

For Example -

```
java:MapperUtilityClass.dateAdd('2013-11-29  
02:20:07', 'yyyy-MM-dd HH:mm:ss', 'MM-dd-yyyy  
HH:mm:ss', 1, 1, 0, 2, 2, 0)
```

Method Name**executeDbUpdateQuery(,)****Description**

This method performs the update operation in the database. This method accepts two parameters, first one is dbConnectionInfoID and other is the String database query.

Syntax

```
java:MapperUtilityClass.executeDbUpdateQuery(  
$dbInfo, 'select * from Table')
```

Method Name**executeDbInsertQuery(,)****Description**

This method performs the insert operation on the database. This method accepts two parameters, first one is dbConnectionInfoID and other is the String database query.

Syntax

```
java:MapperUtilityClass.executeDbInsertQuery($dbInfo  
, '')
```

Method Name**executeDbDeleteQuery(,)****Description**

This method performs the delete operation on the database. This method accepts two parameters, first one is dbConnectionInfoID and other is the String database query.

Method Name	<code>executeDbDeleteQuery(,)</code>
Syntax	<code>java:MapperUtilityClass.executeDbDeleteQuery(\$dbInfo, ' ')</code>

Method Name	<code>encodefileAsBase64(,)</code>
Description	This method encodes the data as base64. This method accepts one parameter which is the String fileLocation.
Syntax	<code>java:MapperUtilityClass.encodeFileAsBase64('C:\SourceFile.txt')</code>

Method Name	<code>invokeREST(, , , , , , ,)</code>
Description	<p>This method calls RESTful web service. It accepts 8 parameters. All the parameters should be of String format. Parameter details are as follows:</p> <ol style="list-style-type: none"> 1. URL of the REST web service. 2. Operation (GET, PUT, POST, or DELETE) to be performed. 3. Data to be sent with POST request. This field is optional. For GET request and other Webservice that doesn't require any input, leave this field blank (''). 4. Request media type, e.g., text/html, application/xml etc. 5. Response media type, e.g., text/html, application/xml etc. 6. HTTP header to be sent with the request. Multiple header parameters should be separated by a comma, e.g., name=san, id=3. 7. Username required to perform authentication of your Webservice. Leave this parameter blank ('') if the Webservice doesn't require any authentication. 8. Password required to perform authentication of your Webservice. Leave this parameter blank ('') if the Webservice doesn't require any authentication.
Syntax	<code>'http://www.thomas-bayer.com/sqlrest/PRODUCT/12', 'GET/POST/PUT/DELETE', 'data to be sent (optional for get)', 'application/xml', 'application/xml', 'Accept-Charset=utf-8,Cache-Control=no-cache,Content-Type:application/xml', 'username', 'password')</code>

GLOBALMETHODCALL IN DATA MAPPER

This section contains the various methods that are a part of the MapperUtilityClass class. The tables below shows the various method names, descriptions, along with their syntax.

Method Name	<code>getAge()</code>
Description	This method returns the present age. This method accepts one parameter in String <parameter-name> format.
Syntax	<code>java:GlobalMethodCall.getAge('dd-MM-yyyy HH:mm:ss')</code>

Method Name	<code>getDaysLeft()</code>
Description	This method returns a value (-1 or 0) depending on the value that you pass in it. It returns 0 if the value passed is greater than the current time and it returns -1 if the value passed is less than the current time. You have to pass the value in yyyy-MM-dd HH:mm:ss format. This method accepts one parameter which is the String <parameter-name>.
Syntax	<code>java:GlobalMethodCall.getDaysLeft('yyyy-MM-dd')</code>

Method Name	<code>getAccountType()</code>
Description	This method returns the account type of the parameter passed in it. It returns 3 values AlphabetNumeric , NumericNumeric , and ErrorData . If you pass alphabet followed by some numeral then it would return AlphabetNumeric . If you pass numeral followed by another numeral then it would return NumericNumeric . If you pass any other values or you do not add a delimiter (-) between the two values then, the method would return ErrorData . This method accepts one parameter in String <parameter-name> format.
Syntax	<pre>java:GlobalMethodCall.getAccountType('alphabet-number-alphabet') java:GlobalMethodCall.getAccountType('number-number-alphabet')</pre> <p>For Example -</p> <pre>java:GlobalMethodCall.getAccountType('john-2912-smith') java:GlobalMethodCall.getAccountType('29-121986-john')</pre>