

# Play Sudoku - Using Table Cell Variants in Web Dynpro Java

## Applies to:

Web Dynpro for Java applications for the SAP NetWeaver 7.0, SP 06 onwards.

## Summary

With NetWeaver 7.0 some new table features were introduced in Web Dynpro for Java. This article, based on a sample project, explains three of them: *TableColumnGroup*, *TableStandardCell* and *TableSingleMarkableCell*.

Author: Stefanie Bacher

Company: SAP AG

**Created on:** 29 June 2007

## Author Bio



Stefanie Bacher works as an information developer within the SAP NetWeaver Product Management User Interaction Team. She focuses on knowledge distribution of Web Dynpro for Java.

## Table of Contents

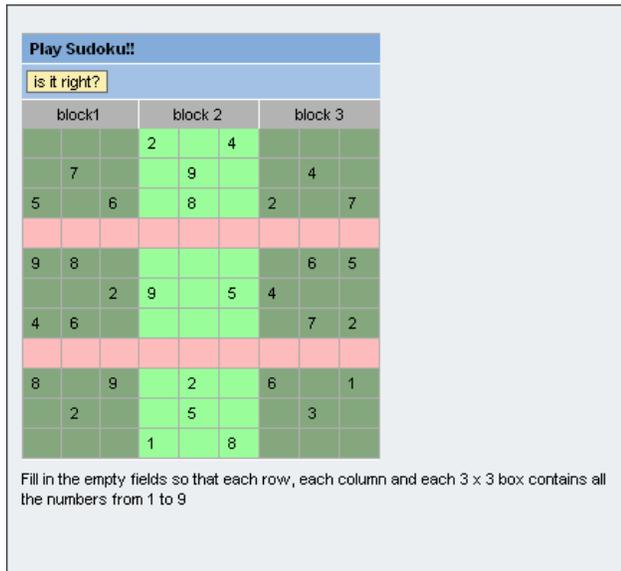
- Running the Sample Application ..... 2
- Creating the Table..... 4
- Setting the Cell Variants Dynamically ..... 5
- Retrieve the x and y Coordinates of a Cell ..... 6

## Using Cell Variants in Web Dynpro Table UI element

### Running the Sample Application

Download the sample project here: [WDJ Table Cell Variants sample project.zip](#)

Unzip and import the project, deploy and run the application:



The table, you see, is built up using the following view elements:

#### **TableColumnGroup.**

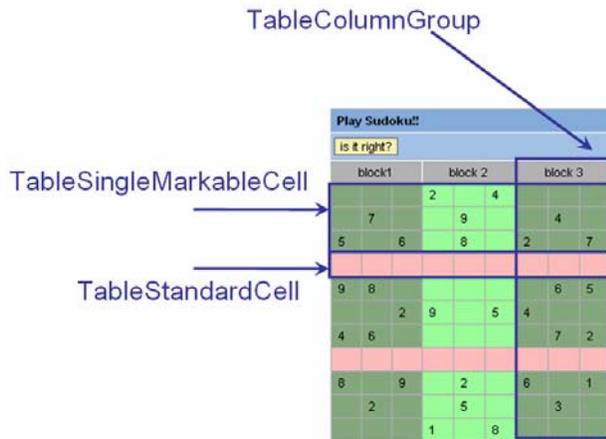
This allows to group table columns under a common header.

#### **TableStandardCell.**

Using this cell variant you can define different designs for cells and even choose another cell editor. This can for example be used to define a row where input fields allow the user to enter new data, whereas the other rows just show the existing data in display mode.

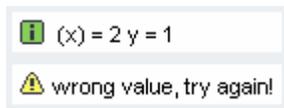
#### **TableSingleMarkableCell**

Allows you to operate on a single cell instead of the row, in this case it is used to check whether the entered number matches the data provided in the solution.



After selecting a single cell, you can enter a number and then check by clicking the “*is it right?*” button.

Two messages will be displayed:



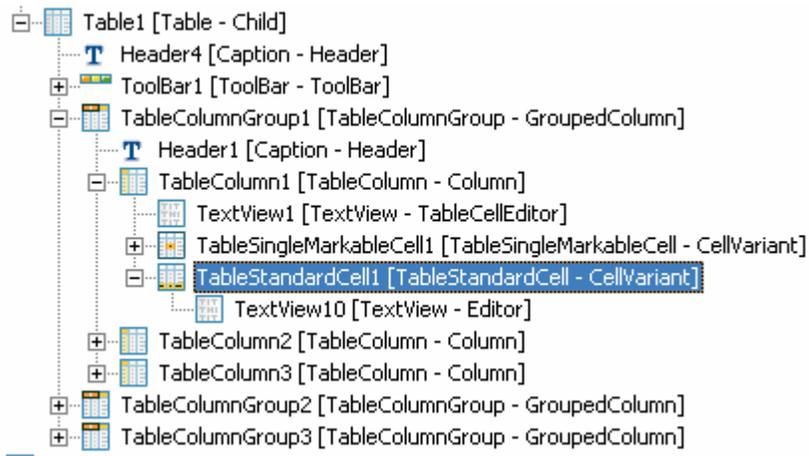
What does that mean?

The first message is for developers: It tells you the coordinates of the cell you operated on

The second message is for players and gives you the information whether the entered value matches the solution.

Now, what has to be done to achieve this?

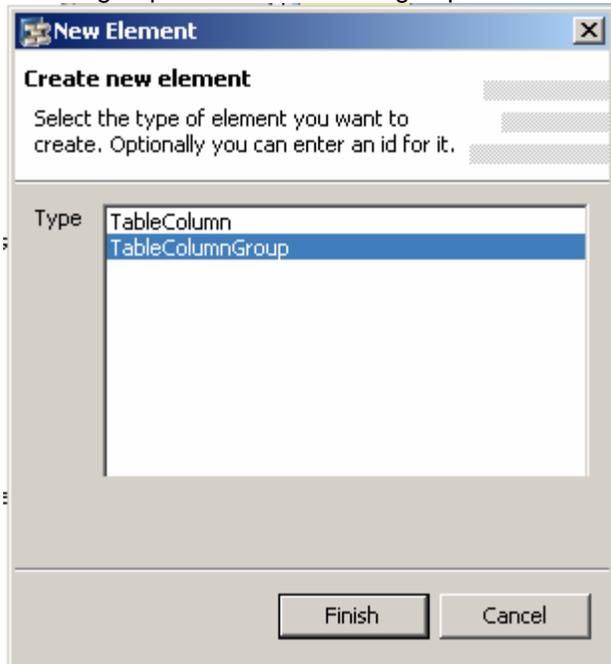
### Creating the Table



The figure above shows the structure of the Sudoku table:

First of all, the Table's `selectionMode` property has to be set to `none`; otherwise the `TableSingleMarkableCell` won't work.

To define the table columns in groups, select the table, open the context menu, choose *Insert Grouped Column* and then *TableColumnGroup*. Select this *TableColumnGroup*, open the context menu, choose *Insert Grouped Column* again and select *TableColumn*. Repeat these steps until the table contains three table column groups and each column group contains three table columns.



A *TableColumnGroup* contains per default a *Header*, whose text property allows you to specify a title stretched over all contained columns.

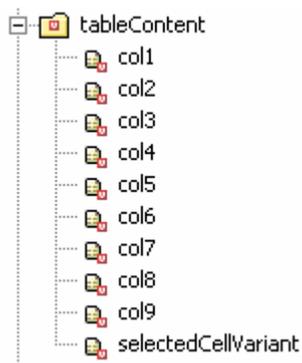
Into each *TableColumn* insert two cell variants: A *TableStandardCell* and a *TableSingleMarkableCell*. Each cell variant needs a *TableCellEditor*. For the *TableStandardCell* choose *TextViews*, for the *TableSingleMarkableCell* *InputFields*.

In this sample *TableStandardCell* is used to separate the blocks by bars in the forth and in the eighth table row. For every *TableStandardCell* the same *variantKey* is defined, in this example *variant2* and – by the way – the same *cellDesign* property is set: *badvalue\_light*.

In the same way all *TableSingleMarkableCells* have the same *variantKey*, here: *variant1*.

### Setting the Cell Variants Dynamically

The view controller context contains a node called *tableContent*. Beside the context attributes called *col1*, *col2*... (they are needed for the data to be displayed), there is a context attribute: *selectedCellVariant* of type *string*.



Using this context attribute you can define by coding in which row which cell variant is displayed. To do this, you need some lines of code. Inside a for loop, where the node elements for the table are created, the respective cell variants are set:

```

...
switch (i) {
    case 3 :
        elem.setSelectedCellVariant("variant2");
        break;
    case 7 :
        elem.setSelectedCellVariant("variant2");
        break;
    default :
        elem.setSelectedCellVariant("variant1");
}
...

```

Now the relevant cell variants are displayed at the right position.

## Retrieve the x and y Coordinates of a Cell

A *TableSingleMarkableCell* has two properties that need to be defined to get things to work:

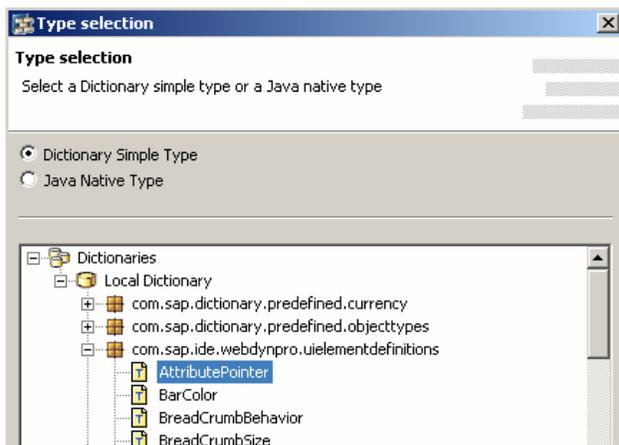
- `attributeToMark`
- `markedData`

Properties	
Property	Value
[-] Element Properties [ViewElement]	
<code>attributeToMark</code>	<code>tableContent.col2</code>
<code>cellDesign</code>	<code>cellDesign</code>
<code>hAlign</code>	<code>auto</code>
<code>id</code>	<code>TableSingleMarkableCell2</code>
<code>markedData</code>	<code>attributePointer</code>
<code>variantKey</code>	<code>variant1</code>

In the properties of each *TableSingleMarkableCell*, the `attributeToMark` property has to be set to that context attribute, which stores the data of this table column where the *TableSingleMarkableCell* is located. To give an example: The *TableSingleMarkableCell2* that is inserted into the second *TableColumn* has to be bound to the context attribute `col2`.

The `markedData` property – again for each *TableSingleMarkableCell* – has to be bound to a context attribute of type *AttributePointer*.

To define this type for a context attribute, click on the three dots on the right hand side of the `type` property row (supposed you are in the *Property view* of your context) and select *Dictionary SimpleType* in the wizard that opens. Open the package `com.sap.ide.webdynpro.uelementdefinitions` and select *AttributePointer*. If a single cell is selected, a pointer to this cell is stored in the context element *attributePointer*.



In this example this is done in an action handler called `onActiontestASingleValue`:

To retrieve this pointer, you need the following coding:

```
IWDAttributePointer attPointer =
wdContext.currentContextElement().getAttributePointer();
```

To identify the x coordinate, retrieve the column name and set the respective value for x:

```
String colName = attPointer.getAttributeInfo().getName();  
if (colName.equals("col1"))  
    x = 1;
```

To identify the y-coordinate, retrieve the index of the node element:

```
int y = attPointer.getNodeElement().index()+1;
```

## Related Content

Online Documentation on Sap Help Portal: [Cell Variants](#)

## Copyright

© Copyright 2007 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.