Applies to:
Visual Composer for SAP enhancement package 1 for SAP NetWeaver Composition Environment 7.1
For more information visit the Visual Composer Page

Summary
In Visual Composer, validation rules help assure that data entry is valid and accurate. They allow a modeler to conveniently specify input conditions that must be met before the application flow continues, or simply display a helpful message.

To demonstrate this capability, I created the "Registration Form" application, which enables the user to enroll for events, get a summary view of the selected events, and eventually to register for those events.

I used simulated services when creating this application, so that you can import the model I attached (see related content at the end of this article) and run it in your Visual Composer system.

Author: Yogev Lidor
Company: SAP
Created on: 01 November 2008

Author Bio
Yogev is a Visual Composer Solution Expert working in the Visual Composer Solution Office.
**Common Use Case**

You can define a set of validation rules to check the validity of the values of a specific control, and to respond in case of an invalid value. For example, when creating a registration form model, the modeler wants to make sure that the first name and last name fields are not left empty when registering the person. To do this, we need to define the corresponding validation rules for those fields.

**The Basics**

To create validation rules in Visual Composer, you need to use the following two areas:

1. The validation area of the Configure task panel of the control you wish to validate – where you set the validation rules and the validation behavior (*Normal* or *Severe*).

   To access this area, perform the following steps:
   2. From the Compose task panel, drag a *Form view* onto the Design board.
   3. Right click the form, and from the context menu choose *Define Data*.
   4. Add a field of type *String* and close the dialog box.
   5. Switch to the Layout board, right-click your control, and from the context menu choose *Configure*.
   6. You can now see the validation section of the Configure task panel of the control that you right-clicked:

```
| Required | false |
| Rules    |      |
| Behavior | Normal |
```

2. The validation tab of the *Define Action* dialog box for the control that triggers the validation - where you set the validation scope (*none, Control, Element, Window*) and define the validation behavior (*Normal* or *Permissive*).

   To access this area, perform the following steps:
   1. Switch to the Layout board.
   2. From the Compose task panel, drag a *Button* control onto the workspace.
   4. Right-click the *Button* control and from the context menu choose *Action*.
   5. In the *Define Action* dialog box, select the *Validation* tab.
   6. You can now see the validation tab of the *Define Action* dialog box of the control that you right-clicked:
Data Validation in Visual Composer for SAP NetWeaver Composition Environment

Let’s Take a Closer Look at this Process

The following part is based on the attached model. Before we continue you should import and deploy the Registration Form model.

The validation process is comprised of two major parts:

Part 1

Defining the validation rules of the control you wish to validate.

To demonstrate this procedure, we will use the Registration Form model. In this model, we want to make sure that the user fills in his/her first and last name (at least). For this to happen, we need to define validation rules that will verify that those fields (First Name and Last Name) are not empty.

After deploying the Registration Form model, perform the following steps:
- Switch to the Design board (by clicking the Design button at the bottom of the workspace) and select the Private Details form.
- Switch to the Layout board (by clicking the Layout button at the bottom of the workspace).
- Right-click the First Name control and from the context menu choose Configure.
- Take a look at the validation area of the Configure task panel. As you can see, this area is composed of three fields:
  1. **Required**: Select the Required checkbox to specify that the control is mandatory and must have a value defined for it at runtime. You can also click the fx icon to open the Dynamic Expression Editor dialog box and define a condition that specifies when the control is mandatory. A control marked as Required shows a visual indication at runtime that it is mandatory. At runtime, if the validation fails, an error message “Input data for required field is missing” appears. When selecting the Required checkbox this message is fixed, meaning you cannot customize it, that is why this option is not used in our case (the First Name and Last Name fields of the Registration Form model). To customize the message, use the Rules field.
  2. **Rules**: In the Rules field, click the Browse button to open the Validation Rules dialog box. Define the Condition, Value, and Error Message for each validation rule that you create, and arrange the rules in the order that you want them to be performed. For more information, see Validation Rules.
Dialog Box in the Visual Composer Reference Guide. In our example, you can see that a single rule is defined:

This rule means that as soon as this control is validated, if the First Name field is left empty, an error message “First Name is a mandatory field” appears.

3. **Behavior**: Select the way in which the control should report validation failures.

- **Normal**: If the control validation fails, indicate this visually at runtime in a nonintrusive manner. For example, a warning message may be displayed but the user is not prevented from changing the focus or moving on to a different step in the application. Note that the actual behavior depends on the runtime implementation. In our specific example, this type of behavior is assigned to some of the fields of the *Private Details* form. For example the *Zip Code* and the *Region* fields. This behavior is required when we want to inform the user of a certain problem and still to enable him/her to finish the task.

- **Severe**: If the control validation fails, indicate this at runtime in a more intrusive manner. For example, an error message may be displayed in a modal dialog box, or the runtime user may be prevented from moving the focus to another control until the value is changed and the validation is successful. Note that the actual behavior depends on the runtime implementation. In our specific example, this type of behavior is assigned to the *First Name* and the *Last Name* fields of the *Private Details* form. This behavior is required when we want to inform the user of a certain problem and prevent him/her from moving on until the problem is handled.

**Note:**

This behavior will occur only when both of the following conditions are met:

1. If a control validation fails.
2. If the validation behavior of the triggering control is set to *Permissive* (a detailed explanation regarding this issue is coming up next).
Part 2
Triggering the validation.

For this we need to define two things:

1. **Validation Scope** (*none, Control, Element, Window*) of the control that triggers the validation. This scope defines the controls that will undergo validation when the action is triggered. To set the validation scope, right-click your triggering control (the control that will undergo validation when clicked, usually a button). From the context menu select *Action*, select the validation tab and select the validation scope:
   - *None*: No validation occurs. For example, you may want to use this scope for a *Cancel* button, since no validation is needed when a *Cancel Window* action is triggered.
   - *Control*: The control for which this action is defined is validated.
   - *Element*: All controls in the view in which the action is defined are validated.
   - *Window*: All controls in the window in which the action is defined are validated. For example, if a component contains views such as forms and tables, all of their controls are validated, but not controls in nested components or popups.

In our specific example (the *Registration Form* model), you can see that the validation scope of the *Next* button of the *Private Details* form is set to *Element*, meaning this button will trigger validation of all the controls in the *Private Details* form as soon as it is clicked.

2. **Validation Behavior** (*Normal* or *Permissive*) of the control that triggers the validation. This behavior defines whether the action should be performed if a control fails validation.

   To set the validation behavior, right-click your triggering control (the control that will undergo validation when clicked, usually a button). From the context menu select *Action*, select the validation tab, and select the validation behavior:
   - *Normal*: An invalid control prevents the action from being performed, irrelevant of any validation defined for the control in the *Behavior* field of the *Validation* group in the Configure task panel.
   - *Permissive*: An invalid control may or may not prevent the action from being performed, depending on the value defined for the control in the *Behavior* field of the *Validation* group in the Configure task panel.

In our specific example (the *Registration Form* model), you can see that the validation behavior of the *Next* button of the *Private Details* form is set to *permissive*, meaning that in case of a validation failure the behavior will be determined by the control that caused the validation failure (by its validation behavior – *Normal* or *Severe*). For example, the validation behavior of the *First Name* control is set to *Severe*, meaning that in case of a validation failure caused by this control an error message will appear and the user will be prevented from moving on unless he/she handles the error. A different behavior can be seen when inspecting the validation behavior of the *Zip Code* control. This control's validation behavior is set to *Normal*, meaning that in case of a validation failure caused by this control a warning message may appear but the user will still be able to move on and finish its task.
Run the Registration Form model and watch the validation behavior in action!

Suggestions for using the model:

1. Click the Next button on an empty form. In this case, you can see the differences between the indications of Severe (the First Name and Last Name fields) behavior and Normal behavior (the rest of the fields).

2. Fill in the First Name and Last Name fields only and click the Next button. In this case, you can see that even though a warning is generated, you can still move on with the registration process due to the Normal validation behavior of the fields (all except the First Name and Last Name fields).

3. Fill in all the fields except the First Name and Last Name fields and click the Next button. The action is prevented since the validation behavior of the First Name and Last Name fields is set to Severe.

4. Fill in all the fields and click the Clear Form button. The field values in the form are cleared without any validation failure. This happens since the validation scope of this button is set to None meaning that no validation occurs when clicking this button and the action is triggered anyway.
Related Content
Validation Rules Dialog Box
Registration Form Model
For more information visit the Visual Composer Page
Copyright
© 2008 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, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, 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.