

Leseprobe

Der Context ist die Schnittstelle für den Datenaustausch zwischen der Benutzeroberfläche und der Web-Dynpro-Component. Ihm kommt daher in Web Dynpro eine besonders große Bedeutung zu. In Teil 3 erhalten Sie wertvolle Tipps, wie Sie den Context in Web Dynpro möglichst effizient verwenden.



»Web-Dynpro-Context verwenden«



Inhaltsverzeichnis



Index



Der Autor

Dominik Ofenloch

Web Dynpro ABAP – 100 Tipps & Tricks

EPUB-Format, 397 Seiten*, in Farbe, Dezember 2013

44,90 Euro, ISBN 978-3-8362-3161-9

*auch erhältlich als gedrucktes Buch: 49,90 Euro, ISBN 978-3-8362-2274-7

TEIL 3

Web-Dynpro-Context verwenden

Der Context ist die Schnittstelle für den Datenaustausch zwischen der Benutzeroberfläche und der Web-Dynpro-Component. Ihm kommt daher in Web Dynpro eine besonders große Bedeutung zu. In diesem Teil gebe ich Ihnen Tipps, wie Sie den Context möglichst effizient verwenden können.

› Tipps in diesem Teil

Tipp 25	Supply-Funktionen einsetzen	112
Tipp 26	Context-Attributeigenschaften verwenden	115
Tipp 27	Context-Knoten zur Laufzeit anlegen	118
Tipp 28	Rekursionsknoten anlegen	121
Tipp 29	Context nicht als Datenablage verwenden	125
Tipp 30	Mapping zwischen Components anlegen	127
Tipp 31	Range-Context-Knoten verwenden	131
Tipp 32	Context-Change-Log verwenden	135
Tipp 33	Singleton-Eigenschaft verwenden	139

Tipp 25

Supply-Funktionen einsetzen

Supply-Funktionen ermöglichen die einfache Befüllung von Context-Knoten mit Daten, der optimale Aufrufzeitpunkt wird dabei vom Web-Dynpro-Framework bestimmt. In diesem Tipp erfahren Sie mehr über die Anwendungsmöglichkeiten von Supply-Funktionen.

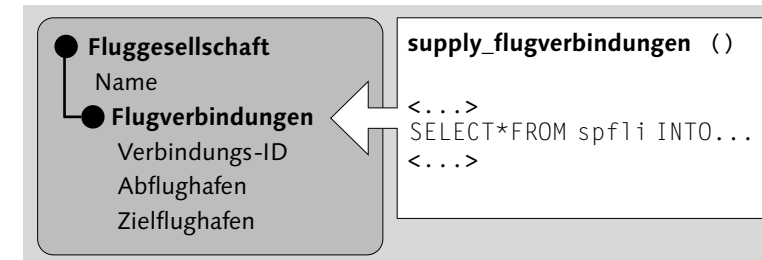
Supply-Funktionen sind Methoden, die zur automatischen Befüllung von Context-Knoten verwendet werden. Sie werden immer dann vom Web-Dynpro-Framework aufgerufen, wenn die Daten des ihnen zugeordneten Context-Knotens gebraucht werden. Dies ist z. B. dann der Fall, wenn die Daten eines Context-Knotens zum ersten Mal auf der Benutzeroberfläche angezeigt werden sollen. Durch diesen Mechanismus des Web-Dynpro-Frameworks sparen Sie in vielen Fällen wertvolle Zeit und Speicherplatz, da die Daten erst bei Bedarf geladen und in den seltensten Fällen alle Daten einer Anwendung benötigt werden.

Besonders im Zusammenhang mit Singleton-Knoten bietet sich die Verwendung von Supply-Funktionen an. Diese haben die Eigenschaft, ihre Kind-Elemente bei jeder Änderung der Lead Selection zu invalidieren, sodass erneut die Supply-Funktionen für die Kind-Elemente aufgerufen werden. Haben Sie also beispielsweise eine zweistufige Knotenhierarchie mit einer Liste von Fluggesellschaften auf der oberen Ebene und den Flugverbindungen der jeweiligen Gesellschaft eine Ebene tiefer, werden die Flugverbindungen immer dann über die Supply-Funktion nachgelesen, wenn Sie die Auswahl ändern.

› Und so geht's

In diesem Tipp zeige ich Ihnen, wie Sie das beschriebene Beispiel mithilfe von Supply-Funktionen realisieren. Beginnen Sie mit dem Anlegen einer Test-Component und Testanwendung. Wechseln Sie anschließend in den View, und legen Sie auf der Registerkarte **Context** den Knoten `FLUGGESELL-`

`SCHAFT` an. Tragen Sie die Dictionary-Struktur `SCARR` ein, und wählen Sie die Kardinalität `1..n` aus. Setzen Sie bei **Init. Lead-Selection** und bei **Singleton** den Wert `Ja`. Tragen Sie zuletzt in das Feld **Supply-Funktion** den Namen `SUPPLY_FLUGGESELLESCHAFT` der noch anzulegenden Supply-Funktion ein, und schließen Sie den Vorgang durch einen Klick auf **Attribute aus Struktur hinzufügen** und die Übernahme der Attribute `CARRID` und `CARRNAME` aus dem ABAP Dictionary in den Knoten ab.



Beispiel für Supply-Funktionen in Web Dynpro: Flugverbindungen

Legen Sie nun die Supply-Funktion für das Lesen der Fluggesellschaft an. Klicken Sie dazu in den Eigenschaften des Context-Knotens doppelt auf die Supply-Funktion `SUPPLY_FLUGGESELLESCHAFT`, und implementieren Sie das folgende Listing zur Initialisierung des Contexts.

```
DATA lt_fluggesellschaft TYPE wd_this->elements_fluggesellschaft.

SELECT * FROM scarr INTO TABLE lt_fluggesellschaft.

node->bind_table(
    new_items      = lt_fluggesellschaft
    set_initial_elements = abap_true ).
```

Supply-Funktion »supply_fluggesellschaft()«: Lesen der Fluggesellschaften

Gehen Sie nun eine Ebene tiefer, und legen Sie unterhalb der Fluggesellschaft den Context-Knoten `FLUGVERBINDUNGEN` an. Tragen Sie die Dictionary-Struktur `SPFLI` ein. Setzen Sie bei **Singleton** den Wert `Ja`. Tragen Sie zuletzt noch in das Feld **Supply-Funktion** den Namen `SUPPLY_FLUGVERBINDUNGEN` der Supply-Funktion ein. Schließen Sie den Vorgang durch einen Klick auf **Attribute aus Struktur hinzufügen** und die Übernahme der Attribute aus dem ABAP Dictionary in den Knoten ab.

Nun können Sie die Supply-Funktion für das Laden der Flugverbindungen nach der Auswahl der Fluggesellschaft anlegen. Die Supply-Funktion hierzu

muss die im darüberliegenden Context-Knoten ausgewählte Fluggesellschaft als Suchkriterium verwenden. Um diese auszulesen, lesen Sie den Importing-Parameter `PARENT_ELEMENT` aus. Dieser beinhaltet das die Lead Selection tragende Context-Element des Vater-Knotens mit der Fluggesellschaft. Verwenden Sie das folgende Listing zur Implementierung der Supply-Funktion.

```
DATA ls_fluggesellschaft TYPE wd_this->element_fluggesellschaft.
DATA lt_flugverbindungen TYPE wd_this->elements_flugverbindungen.

parent_element->get_static_attributes(
  IMPORTING static_attributes = ls_fluggesellschaft ).

SELECT * FROM spfli INTO TABLE lt_flugverbindungen
  WHERE carrid EQ ls_fluggesellschaft-carrid.

node->bind_table( new_items = lt_flugverbindungen ).
```

Supply-Funktion »supply_flugverbindungen()«: Verbindungen von Gesellschaften

Damit haben Sie den Hauptteil der Übung abgeschlossen. Um die Supply-Funktionen testen zu können, müssen Sie zuletzt noch zwei Tabellen für die Darstellung der Fluggesellschaften und der Flugverbindungen anlegen. Am schnellsten können Sie die Tabellen mithilfe des Web-Dynpro-Code-Wizards anlegen, den ich Ihnen in Tipp 55, »Quellcode mit dem Code Wizard generieren«, vorstelle. Testen Sie anschließend die Anwendung. Nach der Auswahl einer Fluggesellschaft in der oberen Tabelle sollten sich die Flugverbindungen in der unteren Tabelle automatisch aktualisieren.

Fluggesellschaft	Fluggesellschaft									
AC	Air Canada									
AF	Air France									
LH	Lufthansa									
Fluggesellschaft	Flugnummer	Land	Abflugstadt	Startflugh.	Land	Ankunftstadt	Zielflugh.	Flugdauer	Abflug	Ankunftszeit
LH	0400		FRANKFURT	FRA		NEW YORK	JFK	504:00	10:10:00	11:34:00
LH	0402		FRANKFURT	FRA		NEW YORK	EWR	515:00	13:30:00	15:05:00
LH	0454		FRANKFURT	FRA		SAN FRANCISCO	SFO	740:00	10:10:00	12:30:00
LH	0455		SAN FRANCISCO	SFO		FRANKFURT	FRA	810:00	15:00:00	10:30:00
LH	2402		FRANKFURT	FRA		BERLIN	SXF	65:00	10:30:00	11:35:00

Test der Supply-Funktion: Alle LH-Verbindungen wurden geladen.

Tipp 26

Context-Attributeigenschaften verwenden

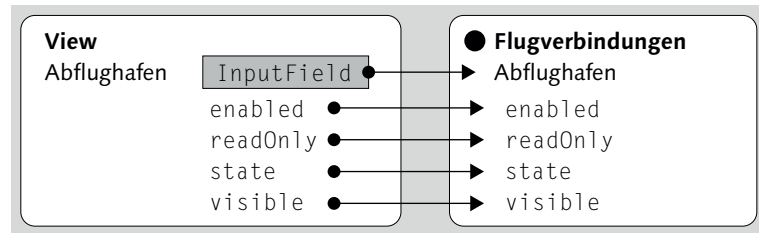
Viele UI-Elemente, wie beispielsweise das Eingabefeld, besitzen grundlegende Eigenschaften, die deren Aussehen steuern. Diese Eigenschaften können Sie direkt an ihre Context-Attribute binden, ohne zusätzliche Werte einzufügen. Wie dies funktioniert, erfahren Sie in diesem Tipp.

Bei der Verwendung von Eingabefeldern stehen Sie immer wieder vor der Aufgabe, diese in bestimmten Situationen auf nur Lesen zu setzen oder nicht anzuzeigen. Hierzu bietet Ihnen das `InputField` neben der primären Eigenschaft `value` für den eigentlichen Wert zusätzlich die UI-Element-Eigenschaften `readOnly` und `visible`. Diese Eigenschaften können Sie im View-Designer statisch setzen und später zur Laufzeit in der View-Methode `wddommodifyview()` dynamisch über einen direkten Zugriff auf das UI-Element ändern.

Praktischer und übersichtlicher ist es, diese Eigenschaften eines Eingabefeldes an ein Context-Attribut zu binden und über den Context zu steuern. Um hierfür keine zusätzlichen Context-Attribute anlegen zu müssen, besitzt jedes Context-Attribut mit `enabled`, `readOnly`, `required` und `visible` vier grundlegende Eigenschaften, die Sie für die Bindung der gleichnamigen UI-Element-Eigenschaften verwenden können.

› Und so geht's

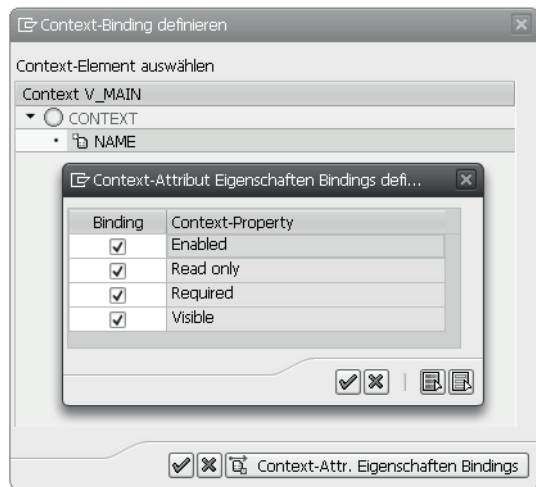
In diesem Tipp zeige ich Ihnen, wie Sie UI-Element-Eigenschaften an deren zugehörige Context-Attributeigenschaften binden und wie Sie diese programmatisch ändern können. Dazu werden Sie ein UI-Element vom Typ `InputField` anlegen und dieses mit seinen Eigenschaften gegen den Context binden. Über einen Button können Sie die Eigenschaften des Feldes steuern. Die folgende Abbildung zeigt Ihnen das Konzept von Context-Attributeigenschaften am Beispiel eines Eingabefeldes für einen Abflughafen.



View mit einem Abflughafen: Verwendung von Context-Attributeigenschaften

Beginnen Sie mit dem Anlegen einer Test-Component und einer Testanwendung. Wechseln Sie in den View der Component, und legen Sie auf der Registerkarte **Context** das Context-Attribut NAME vom Typ STRING an. Gehen Sie dann auf die Registerkarte **Layout**, und fügen Sie ein neues Eingabefeld (InputField) in die UI-Element-Hierarchie ein.

Wählen Sie das neue Eingabefeld aus, und klicken Sie bei der Eigenschaft value auf den Button . Es öffnet sich das Pop-up **Context-Binding definieren**, in dem Sie das Eingabefeld gegen das Context-Attribut NAME binden können. Klicken Sie anschließend auf den Button **Context-Attr. Eigenschaften Bindings**. Im sich daraufhin öffnenden Pop-up-Fenster können Sie auswählen, welche Eigenschaften des Eingabefeldes Sie an das Context-Attribut binden möchten. Wählen Sie alle aus, und schließen Sie beide Pop-ups durch einen Klick auf den Button . Sie gelangen nun zurück in den View-Designer. Wie Sie am Icon erkennen können, sind die UI-Element-Eigenschaften Enabled, Read only, State und Visible des InputField gegen das Context-Attribut NAME gebunden.



Bindung von Context-Attributeigenschaften

Möchten Sie nun die Eigenschaften eines Context-Attributs während der Laufzeit programmatisch ändern, können Sie hierzu im Interface IF_WD_CONTEXT_ELEMENT die Methode `set_attribute_property()` verwenden. Über den Importing-Parameter PROPERTY können Sie angeben, welche der vier Context-Attributeigenschaften Sie verändern möchten. Das Interface bietet hierzu die Konstantenkomponente E_PROPERTY an. Möchten Sie z. B. das Context-Attribut NAME, und damit letztlich Ihr Eingabefeld, auf Read only setzen, könnte Ihr Listing hierzu wie folgt aussehen:

```
wd_context->set_attribute_property(
    EXPORTING
        attribute_name = 'NAME'
        property       = if_wd_context_element=>e_property-read_only
        value          = abap_true
        index          = wd_context->use_lead_selection
        all_elements   = abap_false ).
```

Context-Attributeigenschaften ändern: NAME wird auf Read only gesetzt.

Gehen Sie zurück zu unserer Beispiel-Component. Erweitern Sie diese um einen Button mit der Beschriftung **auf Read-Only setzen**, der direkt rechts neben dem im vorhergehenden Schritt angelegten Eingabefeld platziert wird. Legen Sie aus den Eigenschaften des Buttons heraus eine neue Aktion an, und implementieren Sie in deren Ereignisbehandler den im vorhergehenden Listing vorgestellten Methodenaufruf zum Setzen des Context-Attributs NAME auf Read only. Speichern und aktivieren Sie die Component. Wenn Sie die Testanwendung starten, können Sie durch Anklicken des Buttons das Eingabefeld auf Read only setzen.

Ihr Name:

Steuerung der Eingabefeld-Eingabebereitschaft über den Context

Tipp 27

Context-Knoten zur Laufzeit anlegen

Alle Operationen, die Sie im View-Designer statisch während der Designzeit vornehmen, können Sie auch dynamisch während der Laufzeit durchführen. Am Beispiel des Context-Knotens zeige ich Ihnen in diesem Tipp, wie Sie Veränderungen der Context-Struktur dynamisch vornehmen können.

Immer wieder kann es notwendig sein, den Context einer Component dynamisch zu modifizieren. So können Sie über das Metadatenmodell des Contexts beispielsweise neue Attribute in einem Knoten ergänzen oder auch neue Knoten erstellen.

Ein Beispiel für die dynamische Erzeugung bzw. Veränderung von Context-Knoten könnte ein Tabellenbrowser sein, der auf Web Dynpro basiert und dem Data Browser (Transaktion SE16) ähnelt. Je nach zugrunde liegender Datenbanktabelle benötigen Sie eine andere Knotenstruktur, und damit müssen Sie einen neuen Knoten anlegen oder die Struktur eines existierenden Knotens ändern. Laden Sie z.B. die Datenbanktabelle T100 in den Context, benötigen Sie einen Context-Knoten mit der gleichnamigen Context-Struktur. Möchten Sie die Tabelle BUT000 anzeigen, benötigen Sie einen anderen Context-Knoten. Wie genau Sie einen Knoten dynamisch erzeugen können, zeige ich Ihnen in diesem Tipp am Beispiel der ABAP-Dictionary-Struktur und der Datenbanktabelle SCARR (Übersicht der Fluggesellschaften). Im Anschluss erzeugen Sie eine Tabelle zur Anzeige der Daten im View.

› Und so geht's

Jeder Context-Knoten besitzt zur Laufzeit ein Metadatenobjekt, das über das Interface `IF_WD_CONTEXT_NODE_INFO` angesprochen werden kann. Jedes Metadatenobjekt beschreibt die Struktur und die Eigenschaften eines Knotens. Dazu bietet das Interface des Objekts eine Vielzahl von Methoden zur

Abfrage und Manipulation der Knoteneigenschaften und der Knotenstruktur. So können Sie beispielsweise über die Methode `add_new_child_node()` neue Kind-Knoten in die Struktur unterhalb des Knotens einfügen.

Genug der Theorie. Beginnen Sie mit dem Anlegen einer neuen Test-Component und Anwendung. Öffnen Sie anschließend den View, und wechseln Sie in die Methode `wddommodifyview()`. In dieser implementieren Sie das folgende Listing zur dynamischen Erzeugung des auf der Struktur der Datenbanktabelle SCARR basierenden Knotens.

```
* Rufe das folgende Listing von wddommodifyview( ) nur 1 x auf
CHECK first_time EQ abap_true.

DATA: lo_context_node_info TYPE REF TO if_wd_context_node_info,
      lo_node_scarr        TYPE REF TO if_wd_context_node.

* Hole Knoten-Info-Objekt des ROOT-Knotens
lo_context_node_info = wd_context->get_node_info( ).

* Erzeuge den Knoten dynamisch
lo_context_node_info->add_new_child_node(
  EXPORTING
    static_element_type = 'SCARR' " ABAP-Dictionary-Struktur des Knotens
    name                 = 'SCARR' " Name des Knotens
    is_multiple          = 'X' " können mehrere Elemente existieren?
  ).

* Hole zuletzt noch den Knoten SCARR
lo_node_scarr = wd_context->get_child_node( 'SCARR' ).
```

Dynamische Erzeugung des Context-Knotens SCARR

Damit haben Sie den eigentlichen Gegenstand dieses Tipps abgeschlossen, das dynamische Anlegen von Context-Knoten zur Laufzeit. Jedoch ist dieser Knoten allein ziemlich nutzlos. Daher zeige ich Ihnen im folgenden Listing noch, wie Sie den Knoten mit Daten füllen und anschließend daraus eine Tabelle in das View-Layout generieren können.

```
DATA: lt_scarr TYPE TABLE OF scarr,
      lo_container TYPE REF TO cl_wd_transparent_container.

* Lese die Datenbanktabelle SCARR aus
SELECT * FROM scarr INTO TABLE lt_scarr.

* Binde den Tabelleninhalt gegen den Context
lo_node_scarr->bind_table( new_items = lt_scarr ).
```

```

* Hole die UI-Wurzel
lo_container ?= view->get_element( 'ROOTUIELEMENTCONTAINER' ).

* Erzeuge dynamisch eine Client-Tabelle
cl_wd_dynamic_tool=>create_c_table_from_node(
  EXPORTING
    ui_parent = lo_container
    node      = lo_node_scarr ).

```

Füllung des SCARR-Knotens mit Daten und Erzeugung einer Tabelle

Aktivieren Sie die Component, und starten Sie die Testanwendung. Hat alles geklappt, sollte die Testanwendung die folgende Tabelle im Browser anzeigen.

Mandant	Fluggesellschaft	Fluggesellschaft	Währ. d. Flugg.
001	AC	Air Canada	CAD
001	AF	Air France	EUR
001	LH	Lufthansa	EUR

Fertige Testanwendung: Dynamisch erzeugte Tabelle und Knoten

Tipp 28

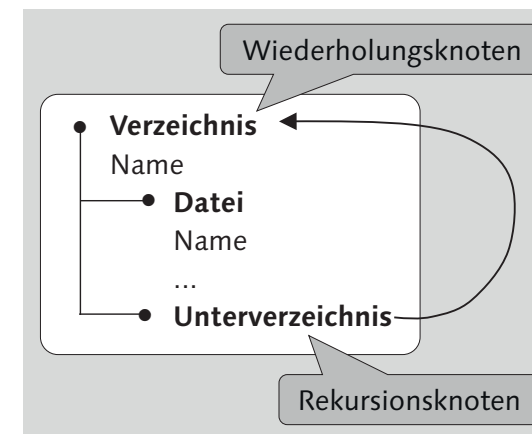
Rekursionsknoten anlegen

Hierarchien und Baumstrukturen finden Sie an vielen Stellen auf Ihrem Computer. Das bekannteste Beispiel ist die Verzeichnisstruktur einer Festplatte. Auch der Web-Dynpro-Context kann Hierarchien darstellen, was ich Ihnen in diesem Tipp zeigen möchte.

Mithilfe von sogenannten *Rekursionsknoten* können Sie im Context beliebig tief geschachtelte Hierarchien abbilden, wie z.B. Baum- oder Verzeichnisstrukturen. Ein Rekursionsknoten stellt dabei keinen Context-Knoten mit Eigenschaften und Attributen im klassischen Sinn dar, sondern er dient als Repeater eines darüberliegenden Knotens. Im folgenden Tipp zeige ich Ihnen, wie Sie einen solchen Rekursionsknoten anlegen.

› Und so geht's

Zum Einstieg möchte ich Ihnen die Funktionsweise des Rekursionsknoten an einem Beispiel erläutern:



Rekursionsknoten schematisch dargestellt

Der Knoten **Verzeichnis** besitzt das Context-Attribut **Name**. Jedes Verzeichnis kann eine beliebige Anzahl von Dokumenten besitzen, die im Sub-Knoten **Datei** abgelegt werden. Der Sub-Knoten **Unterverzeichnis** ist ein Rekursionsknoten, der auf die darüberliegende Struktur des Knotens **Verzeichnis** verweist. Bei einem Wechsel in das **Unterverzeichnis** wiederholt sich die Struktur des Wiederholungsknotens, das heißt, die Attribute des Knotens **Verzeichnis** und dessen Sub-Knoten **Datei** und **Unterverzeichnis** werden in der Hierarchie eine Ebene tiefer angeboten.

Rekursive Hierarchien können durch zwei UI-Elemente dargestellt werden. Für die Darstellung von Hierarchien in Tabellen können Sie das UI-Element `TreeByNestingTableColumn` verwenden, für die Darstellung außerhalb von Tabellen existiert das UI-Element `Tree`. Die Erstellung von rekursiven Knoten, und wie Sie diese im `Tree` verwenden können, zeige ich Ihnen am Beispiel einer Context-basierten Verzeichnisstruktur.

Legen Sie eine neue Test-Component und Testanwendung an. Wechseln Sie dazu in den View, und beginnen Sie mit dem Anlegen der zu wiederholenden rekursiven, Context-basierten Verzeichnisstruktur. Öffnen Sie die Registerkarte **Context**, und legen Sie den Knoten `VERZEICHNIS` mit der Kardinalität `0..n` an. Fügen Sie das Attribut `NAME` vom Typ `STRING` in den Knoten ein. Legen Sie nun eine Ebene tiefer den Knoten `DATEI` an, und fügen Sie die Attribute `NAME` und `TYP`, beide vom Datentyp `STRING`, in den Knoten ein.

Legen Sie nun den Rekursionsknoten an. Klicken Sie dazu mit der rechten Maustaste auf den Knoten `VERZEICHNIS`, und wählen Sie **Anlegen ▶ Rekursionsknoten** aus. Tragen Sie im folgenden Pop-up unter **Knotenname** den Namen des Rekursionsknotens `UNTERVERZEICHNIS` ein, und klicken Sie zur Auswahl des Wiederholungsknotens auf den Button **Auswählen**. Wählen Sie im folgenden Pop-up den Knoten `VERZEICHNIS` aus, und bestätigen Sie die Auswahl durch einen Klick auf ☒. Damit haben Sie den Rekursionsknoten angelegt. Im Anschluss sollte Ihre Context-Struktur aussehen, wie in der folgenden Abbildung dargestellt.

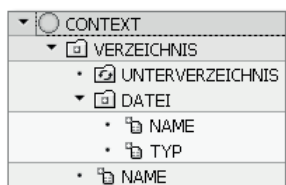


Abbildung von Verzeichnisstrukturen im Context mithilfe von Rekursionsknoten

Um nun die Verzeichnisstruktur als Baum auf der Benutzeroberfläche abbilden zu können, benötigen Sie jetzt noch das UI-Element `Tree`. Dieses besitzt, vergleichbar mit echten Bäumen im Wald, Blätter und Zweige:

- **Blätter** (`TreeItemType`)
Blätter sind Endpunkte eines Baums. Im Beispiel der Verzeichnisstruktur können Blätter z. B. durch Dateien in einem Verzeichnis repräsentiert werden. Jeder Baum kann viele verschiedene Typen von Blättern besitzen, wobei jeder Typ in einem Baum durch das View-Element `TreeItemType` repräsentiert wird. Dies kann beispielsweise dann sinnvoll sein, wenn Sie mehrere verschiedene Dateiformate darstellen möchten.
- **Zweige** (`TreeNodeType`)
Zweige stellen Knotenpunkte in Bäumen dar. Im Beispiel der Verzeichnisstruktur ist ein Zweig daher mit einem Verzeichnis gleichzusetzen. Auch Bäume können viele verschiedene Typen von Zweigen besitzen, diese werden durch View-Elemente vom Typ `TreeNodeType` repräsentiert. Durch die Verwendung verschiedener Zweigttypen können Sie z. B. unterschiedliche Icons für verschiedene Verzeichnistypen implementieren.

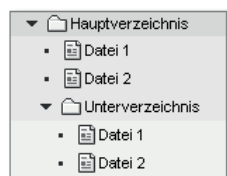
Wechseln Sie nun auf die Registerkarte **Layout**, und legen Sie den Baum an. Fügen Sie das UI-Element `Tree` in die View-Hierarchie ein. Deaktivieren Sie das Ankreuzfeld der Eigenschaft `rootVisible`, und binden Sie die Eigenschaft `dataSource` gegen den Knoten `VERZEICHNIS`. Durch diese Einstellungen setzen Sie den Knoten `VERZEICHNIS` direkt als Wurzel-Element des Baums fest.

Fügen Sie nun einen neuen Zweig in den Baum ein, indem Sie mit der rechten Maustaste auf den Baum klicken und im Kontextmenü den Eintrag **Knotentyp einfügen** auswählen. Geben Sie im folgenden Pop-up dem Zweig den Namen `VERZEICHNIS`, und wählen Sie den Knotentyp `TreeNodeType` aus. Nachdem Sie in den View-Designer zurückgekehrt sind, binden Sie dessen Eigenschaft `dataSource` gegen den Context-Knoten `VERZEICHNIS`. Binden Sie die Eigenschaft `text` gegen das Attribut `NAME` im Verzeichnis. Um dem Verzeichnis ein verzierendes Icon zu geben, empfehle ich Ihnen, als `iconSource` den Wert `~Icon/FolderFile` einzutragen.

Wiederholen Sie nun die letzten Schritte für die Blätter des Baums, das heißt für die Dateien. Wählen Sie erneut an der gewünschten Stelle in der Hierarchie den Eintrag **Knotentyp einfügen** im Kontextmenü aus, und fügen Sie ein Blatt vom Typ `TreeItemType` in den Baum ein. Binden Sie dessen Eigenschaft `dataSource` gegen den Knoten `DATEI`. Wiederholen Sie das Data

Binding anschließend für die Eigenschaft `text` mit dem Knotenattribut `NAME`. Als `iconSource` empfehle ich Ihnen das Attribut `~Icon/DocumentFile`.

Damit ist die Test-Component fast fertig: Sie haben eine rekursive Context-Struktur angelegt, die in Form eines Baums als Verzeichnisse mit Dateien angezeigt wird. Bisher ist der Context jedoch noch leer, Sie müssen noch Dateien und Verzeichnisse anlegen. Implementieren Sie hierzu die View-Methode `wddoinit()`, und füllen Sie die Context-Struktur mit beliebigen Daten. Speichern Sie die Änderungen, und aktivieren Sie die Component.



Test der fertigen Testanwendung: Rekursiver Knoten



Tipp 29

Context nicht als Datenablage verwenden

Halten Sie Ihren Context sauber! Nicht selten werden im Context Daten abgelegt, die für das UI nicht relevant sind. Diese Zweckentfremdung führt später in vielen Fällen zu Problemen, die nur mit großem Aufwand behoben werden können.

Der Context ist die Datenaustauschnittstelle zwischen dem Controller einer Component und den UI-Elementen im View bzw. im Browser. In der Praxis finden Sie aber auch häufig Knoten und Attribute im Context, die nicht für die Anzeige auf der Benutzeroberfläche gedacht sind. Welche Elemente Sie im Context ablegen sollten und welche nicht, erfahren Sie in diesem Tipp.

› Und so geht's

Nicht selten findet man im Context eine Mischung verschiedenster Daten. Zum Beispiel:

- das Attribut `NAME` zur Eingabe des Benutzernamens in einem Eingabefeld
- die Customizing-Tabelle `ZTC_BOOKING_TYPES` zur Zwischenspeicherung der verfügbaren Buchungstypen
- das Attribut `DATA_CHANGED`, als Indikator dafür, ob beim Verlassen des Views die Daten gespeichert werden müssen

Von diesen drei Beispielen gehört jedoch nur das Attribut `NAME` in den Context, da es einen direkten Bezug zu einem Eingabefeld und somit zur Benutzeroberfläche enthält. Daten, die nur in einer Component zwischengespeichert werden sollen, sei es die zwischengespeicherte Tabelle mit den

Buchungstypen oder das Attribut `DATA_CHANGED`, sollten Sie auf der Registerkarte **Attribute** oder in einer Assistance-Klasse ablegen.

Warum aber werden so häufig Daten ohne direkten UI-Element-Bezug im Context abgelegt? Häufig liegt dies daran, dass die Entwickler von der Context-Mapping-Funktionalität Gebrauch machen möchten. Aus Entwicklersicht ist es häufig sehr bequem, in einer Component an einer zentralen Stelle (meist dem Component-Controller) einen `SETTINGS`-Knoten anzulegen und diesen dann in den Views der Component oder in verwendeten Components in den lokalen Context zu mappen.

Ein großer Nachteil des statischen Mappings ist es, dass Änderungen am Originalknoten nicht automatisch an die gemappten Knoten weitergereicht werden. Fügen Sie also z. B. im Component-Controller am `SETTINGS`-Knoten das Attribut `BUTTONS_ENABLED` nachträglich ein, müssen Sie diese Änderung in allen gemappten Knoten nachziehen, indem Sie **Aktualisieren Mapping** im Kontextmenü der jeweiligen Views in der entsprechenden Component auswählen. Das gilt auch für den Fall, dass Sie ein nicht mehr verwendetes Attribut im Originalknoten löschen.

Ein weiterer wichtiger Grund dafür, dass Sie den Context nicht als Datenablage für nicht UI-relevante Attribute verwenden sollten, ist der fehlende Verwendungsnachweis für Context-Attribute. Dieses Problem ist ein allgemeines Problem im Context, das für alle Context-Attribute gilt. Haben Sie einen Context-Knoten angelegt, sind für den Knoten zwar Konstanten verfügbar, nicht jedoch für die Attribute im Knoten. Für die Abfrage oder das Setzen eines Context-Attributwerts wird der Attributname dann häufig als Literal in einfachen Anführungszeichen geschrieben. Da auf Literale keine Verwendungsnachweise möglich sind, führt dies vor allem bei der Löschung von nicht mehr benötigten Attributen zu Problemen. Häufig kommt es vor, dass ein gelöscht Attribut in irgendeiner Ecke Ihrer Component doch noch verwendet wird, was das System mit einem Shortdump quittiert.

Ein letzter Grund dafür, warum Sie den Context nicht als Datenablage verwenden sollten, ist schlicht und einfach die Performance. Jeder zusätzliche Knoten und jedes zusätzliche Context-Attribut bindet Systemressourcen, die Sie sparen können. So werden Context-Knoten und Context-Elemente je in eigenen Objekten abgebildet. Für jedes Context-Attribut werden darüber hinaus UI-Element-Eigenschaften verwaltet, die zusätzliche Performance benötigen. Daher empfehle ich Ihnen, den Context nur dann zu verwenden, wenn die darin unterzubringenden Daten einen direkten Bezug zu UI-Elementen in einem View haben.

Tipp 30

Mapping zwischen Components anlegen

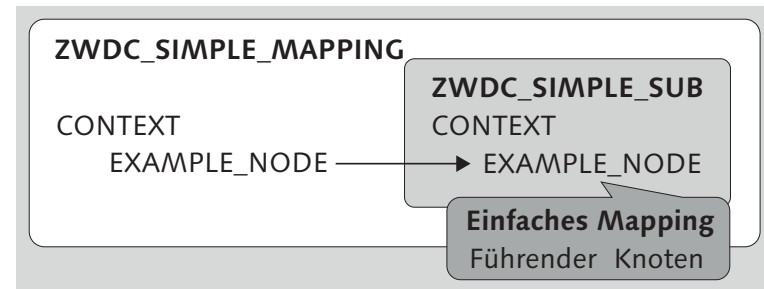
Durch das Component-übergreifende Mapping können Sie Context-Knoten über mehrere Components hinweg teilen, was Ihnen die Erstellung von Multi-Component-Architekturen erleichtert.

Wenn Sie schon einmal vor der Entwicklung einer umfangreichen Web-Dynpro-Anwendung gestanden haben, kommt Ihnen diese Fragestellung sicherlich bekannt vor: Soll ich eine große Component anlegen, oder soll die Anwendung in viele kleine Components aufgeteilt werden? In der Regel ist die Aufteilung in wenige, semantisch zusammengehörende Components die bessere Wahl. Mithilfe des Mappings können Sie in Context-Knoten abgelegte Daten über die Grenze eines Controllers hinweg austauschen. Wie das Component-übergreifende Mapping funktioniert und welche Besonderheiten es dabei gibt, erkläre ich in diesem Tipp.

› Und so geht's

Beim Component-übergreifenden Mapping wird zwischen einfachem und externem Mapping unterschieden. Während beim einfachen Mapping der Informationsfluss vom Context der eingebetteten Component zum Context der verwendenden Component läuft, ist dies beim externen Mapping genau umgekehrt.

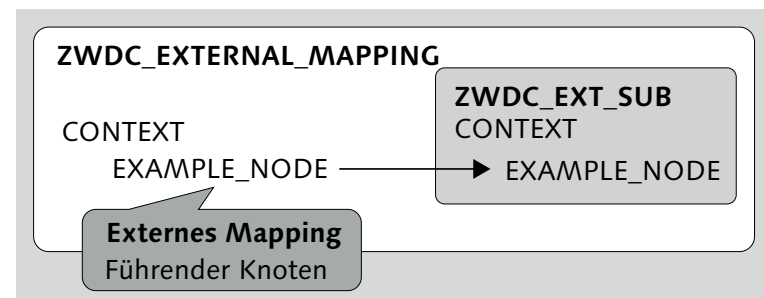
Betrachten wir zuerst das einfache Mapping. Bei diesem befindet sich der primäre Knoten in der eingebetteten Component (`ZWDC_SIMPLE_SUB`). Falls in dieser eine Supply-Funktion existiert, versorgt diese damit den gemappten Knoten in der umgebenden Component (`ZWDC_SIMPLE_MAPPING`). Bei der Erstellung des einfachen Mappings wird die Knotenstruktur von der eingebetteten Component in die umgebende Component übernommen.



Einfaches Mapping

Die Definition eines einfachen Mappings unterscheidet sich kaum von der Erstellung eines Controller-übergreifenden Mappings. Stellen Sie hierzu sicher, dass der zu mappende Knoten – im Component-Controller der eingebetteten Component – als Interface-Knoten definiert ist. Tragen Sie anschließend eine Component-Verwendung für die einzubettende Component in der umgebenden Component ein. Der anschließend durchzuführende Mapping-Vorgang unterscheidet sich nicht mehr vom normalen Controller-übergreifenden Mapping.

Beim externen Mapping verläuft der Informationsfluss im Vergleich zum einfachen Mapping umgekehrt: So ist nicht mehr der eingebettete (ZWDC_EXT_SUB), sondern der lokale Knoten (ZWDC_EXTERNAL_MAPPING) der umgebenden Component der primäre Knoten. Dieser kann dort mithilfe einer Supply-Funktion mit Daten gefüllt werden.



Externes Mapping

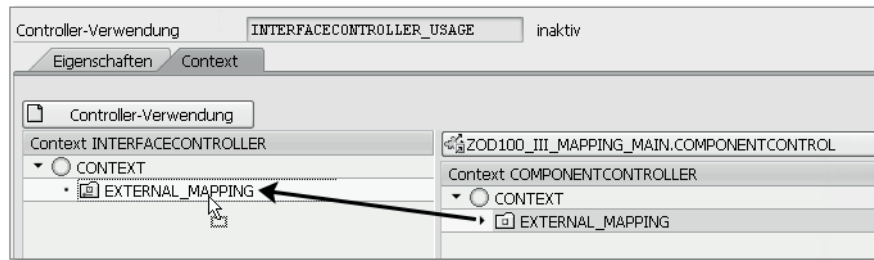
Darüber hinaus sind beim externen Mapping noch einige Besonderheiten zu beachten. So können Sie den extern zu mappenden Knoten der eingebetteten Component bei der Definition völlig untypisiert lassen. Der Knoten erhält in diesem Fall seine komplette Typisierung erst durch den Context-Knoten, für den ein Mapping auf den externen Knoten definiert wird. Gegen

einen so angelegten Knoten kann im zugehörigen Controller jedoch nur dynamisch programmiert werden, da seine Struktur von der umgebenden Component abhängt und somit zur Designzeit noch nicht bekannt ist. Das prominenteste Beispiel für eine dynamische Component ist die ALV-Component.

In vielen Fällen lässt sich ein gewünschtes Resultat sowohl über ein einfaches als auch über ein externes Mapping erreichen, wenn das Design der Anwendung entsprechend umgestellt wird. Haben Sie in Ihrer Architektur eine zentrale Component für den Datenaustausch vorgesehen, bietet sich meist das einfache Mapping der eingebetteten Component auf die umgebende Component an. Haben Sie jedoch eine generische Component für die Anzeige von Daten aus einer lokalen Component vorgesehen, ist das externe Mapping meist die bessere Wahl.

Das Anlegen eines externen Mappings unterscheidet sich deutlich vom Anlegen eines einfachen Mappings. Das externe Mapping wird nicht im Controller der verwendenden Component definiert, sondern in einer extra hierfür vorhandenen Benutzeroberfläche. Gehen Sie dazu wie folgt vor:

1. Stellen Sie sicher, dass der zu mappende Knoten in der eingebetteten Component als Interface-Knoten und als **Input-Element (ext.)** definiert ist. Öffnen Sie hierzu die Eigenschaften des Knotens auf der Registerkarte **Context** im Component-Controller der eingebetteten Component.
2. Tragen Sie eine Component-Verwendung für die einzubettende Component in der umgebenden Component ein.
3. Öffnen Sie den Pfad **Component-Verwendungen** ▶ **<Component-Verwendungsname>**, und wählen Sie dort im Kontextmenü **Controller-Verwendung anlegen** aus. Öffnen Sie anschließend den Sub-Knoten **INTERFACECONTROLLER_USAGE**. Definieren Sie im nun vorliegenden Dialog das externe Mapping:
 - Tragen Sie eine Component-Verwendung des lokalen Component-Controllers ein. Klicken Sie dazu auf der Registerkarte **Eigenschaften** auf das Symbol . Wählen Sie den gewünschten lokalen Controller aus, und schließen Sie das Pop-up.
 - Wechseln Sie anschließend auf die Registerkarte **Context**, und legen Sie das externe Mapping an. Ziehen Sie den lokalen, zu mappenden Knoten von der rechten Bildhälfte per Drag & Drop zum Interface-Controller auf der linken Seite.



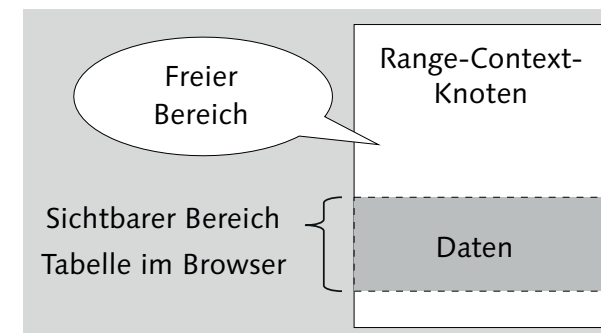
Externes Mapping anlegen

Vor allem die Möglichkeit, extern gemappte Knoten während der Designzeit untypisiert zu lassen, bietet Ihnen eine große Flexibilität beim Mapping unterschiedlicher Components. Von dieser Flexibilität machen daher vor allem generische Sub-Components Gebrauch. Für einfache und weniger komplexe Components reicht meistens das einfache Mapping aus.

Tipp 31 Range-Context-Knoten verwenden

Context-Knoten speichern häufig sehr große Tabellen mit mehreren Tausend Zeilen. Meist wird davon jedoch nur ein Bruchteil der Daten im Context benötigt, typischerweise ist dies der Ausschnitt der sichtbaren Tabellenzeilen. Was liegt daher näher, als nur den sichtbaren Ausschnitt der Daten im Context zur Speicheroptimierung abzulegen?

Mithilfe von Range-Context-Knoten können Sie große Knoten speichereffizient im Context ablegen. Das Grundprinzip ist dabei, dass nicht immer der gesamte Datenbestand eines Knotens im Context liegen muss, da meist immer nur ein kleiner Ausschnitt der Daten gebraucht wird. Dies ist typischerweise bei in Tabellen dargestellten Context-Knoten der Fall, da hier letztlich meist nur ungefähr zehn Zeilen einer Tabelle zum gleichen Zeitpunkt angezeigt werden. Der nicht angezeigte Bereich des Context-Range-Knotens wird dann erst gar nicht in den Speicher geladen.



Grundprinzip von Range-Context-Knoten

In diesem Tipp zeige ich Ihnen am Beispiel von Flugverbindungen aus der Datenbanktabelle `SFLIGHT`, wie Sie Range-Context-Knoten anlegen und verwenden können.

› Und so geht's

Im Unterschied zu normalen Context-Knoten können Sie Range-Context-Knoten nur dynamisch zur Laufzeit erzeugen. Dazu verwenden Sie die Methode `add_new_child_node()` des Interface `IF_WD_CONTEXT_NODE_INFO`. Durch das Setzen des Parameters `IS_RANGE_NODE` auf `X` legen Sie fest, dass der zu erzeugende Knoten ein Range-Context-Knoten sein soll. Über die Parameter `SUPPLY_METHOD` und `SUPPLY_OBJECT` legen Sie fest, welche Range-Supply-Methode verwendet werden soll.

Beginnen Sie mit dem Anlegen des Flugverbindungsbeispiels. Legen Sie eine neue Test-Component und Testanwendung an. Wechseln Sie in den View der neuen Component, und implementieren Sie das folgende Listing zur Erzeugung eines Range-Context-Knotens und einer einfachen Tabelle in der Methode `wddommodifyview()`. Der Knoten wird über eine im folgenden Schritt anzulegende Supply-Methode `supply_sflight()` mit Daten versorgt. Nach der Erzeugung der Range-Tabelle ermitteln Sie den Umfang der verfügbaren Flugverbindungen und übergeben die Zahl über `set_max_element_count()` an den Range-Context-Knoten.

```
CHECK first_time EQ abap_true.

DATA: lo_nd_sflight_rng TYPE REF TO if_wd_context_node_range,
      lo_container      TYPE REF TO cl_wd_transparent_container,
      lv_sflight_count  TYPE i.

* Erzeuge einen Range-Context-Knoten
wd_context->get_node_info( )->add_new_child_node(
  EXPORTING
    supply_method      = 'SUPPLY_SFLIGHT'
    supply_object       = me
    static_element_type = 'SFLIGHT'
    name               = 'SFLIGHT'
    is_range_node       = abap_true ).

lo_nd_sflight_rng ?= wd_context->get_child_node( 'SFLIGHT' ).

* Ermittle Anzahl der insgesamt verfügbaren Elemente
SELECT COUNT(*) FROM sflight INTO lv_sflight_count.
lo_nd_sflight_range->set_max_element_count( lv_sflight_count ).

* Generiere eine dynamische Tabelle
lo_container ?= view->get_root_element( ).
cl_wd_dynamic_tool=>create_c_table_from_node(
  EXPORTING
```

```
ui_parent = lo_container
node      = lo_nd_sflight_range ).
```

Implementierung von Methode »wddommodifyview()«

Legen Sie nun die Supply-Funktion `supply_sflight()` für den Range-Context-Knoten an. Diese Supply-Funktion muss dabei, im Unterschied zu klassischen Supply-Funktionen, nicht vom **Methoden-Typ** Supply-Funktion, sondern vom Typ Methode sein. Ein weiterer Unterschied der Range-Supply-Methode ist, dass sie mit `FROM_INDEX` und `TO_INDEX` zusätzliche Parameter für die Eingrenzung des angefragten Range-Bereichs besitzt. Definieren Sie für die Supply-Funktion daher manuell die folgenden Importing-Parameter:

Importing-Parameter	Referenz- bzw. Datentyp
NODE	IF_WD_CONTEXT_NODE_RANGE
PARENT_ELEMENT	IF_WD_CONTEXT_ELEMENT
FROM_INDEX	I
TO_INDEX	I

Importing-Parameter der Supply-Funktion

Implementieren Sie nun die Supply-Funktion. Im ersten Schritt müssen Sie die Flugverbindungen des angeforderten Ranges von der Datenbank lesen. Hierzu geben Sie beim `SELECT` mit der Ergänzung `UP TO to_index ROWS` den oberen angefragten Indexwert der Supply-Methode an. Da Sie beim `SELECT` leider keine untere Lesegrenze angeben können, müssen Sie die Werte unterhalb des Parameterwerts von `FROM_INDEX` im nächsten Schritt in einem `LOOP` verwerfen. Damit der Range-Context-Knoten den zu übergebenden Ausschnitt in den Gesamtbereich einsortieren kann, muss außerdem jede Tabellenzeile mit einem Zeilenindex versehen werden. Zuletzt können Sie den angeforderten Tabellenbereich an den Range-Context-Knoten über die Methode `set_table_range()` übergeben. Das folgende Listing zeigt Ihnen das gesamte Listing der Range-Supply-Methode `supply_sflight()`.

```
TYPES:
  BEGIN OF ty_s_sflight_index,
    index TYPE int4.
  INCLUDE TYPE sflight.
TYPES: END OF ty_s_sflight_index.

DATA: lt_sflight_range TYPE TABLE OF ty_s_sflight_index,
```



```

ls_sflight_range TYPE ty_s_sflight_index,
lt_sflight       TYPE SORTED TABLE OF sflight
                  WITH UNIQUE KEY carrid connid fldate.

FIELD-SYMBOLS: <sflight> LIKE LINE OF lt_sflight.

SELECT * FROM sflight INTO CORRESPONDING FIELDS OF TABLE
  lt_sflight UP TO to_index ROWS ORDER BY carrid connid fldate.

* Konvertiere die SFLIGHT-Tabelle in eine Range-Context-Zeilen-
* Struktur mit vorangestelltem Zeilenindex
LOOP AT lt_sflight ASSIGNING <sflight> FROM from_index.
  MOVE-CORRESPONDING <sflight> TO ls_sflight_range.
  ls_sflight_range-index = sy-tabix.
  APPEND ls_sflight_range TO lt_sflight_range.
ENDLOOP.

* Übergebe den angeforderten Range an den Context
node->set_table_range(
  new_items      = lt_sflight_range
  index          = from_index
  invalidate_child_nodes = abap_false ).

```

Supply-Methode »supply_sflight()«

Damit ist das Beispiel mit dem Range-Supply-Knoten fertig. Aktivieren Sie die Component. Um das Beispiel besser nachvollziehen zu können, empfiehlt es sich, zu Beginn von `wddommodifyview()` und `supply_sflight()` je einen Breakpoint zu setzen. Starten Sie anschließend die Testanwendung, und testen Sie die Tabelle.

Fluggesellschaft	Flugnummer	Flugdatum	Flugpreis	währ. d. Flugg.
AC	0820	20.12.2002	1.222,00	CAD
AF	0820	23.12.2002	2.222,00	EUR
LH	0400	28.02.1995	899,00	DEM
LH	0454	17.11.1995	1.499,00	DEM
LH	0455	06.06.1995	1.090,00	USD

Test der auf einem Range-Context-Knoten basierenden Tabelle mit Flugverbindungen

Tipp 32

Context-Change-Log verwenden

Ein Benutzer ändert in einer Tabelle mit mehreren Hundert Zeilen einige Datensätze. Mithilfe des Context-Change-Logs können Sie diese Änderungen sehr elegant speichern.

Beim Context-Change-Log handelt es sich um ein Protokoll, das Änderungen am Context durch den Benutzer überwacht und diese in einer Tabelle aufzeichnet. Mithilfe des hierbei entstehenden Protokolls können Sie anschließend gezielt Benutzereingaben weiterverarbeiten und dabei z. B. nur geänderte Daten auf der Datenbank speichern. Wie genau dies funktioniert, erfahren Sie am Beispiel von Flugbuchungen in diesem Tipp.

› Und so geht's

Legen Sie zur Vorbereitung des Change-Log-Beispiels eine neue Test-Component und Testanwendung an. Wechseln Sie in den View, und legen Sie den Context-Knoten SBOOK mit der Kardinalität 0..n und der zugrunde liegenden ABAP-Dictionary-Struktur SBOOK an. Übernehmen Sie eine kleine Auswahl der Datenbankfelder als Attribute in den Context-Knoten. Generieren Sie anschließend mit dem Code Wizard eine auf der Knotenstruktur basierende Tabelle, wobei Sie als **Standard Cell-Editor** das UI-Element `TextEdit` auswählen. Fügen Sie nun eine Toolbar in die Tabelle ein, und legen Sie in dieser einen Button mit der Beschriftung `Neue Zeile` zum Hinzufügen eines neuen Context-Elements in den Knoten an. Legen Sie aus den Eigenschaften des Buttons heraus eine neue Aktion für das Ereignis `onAction` an, und implementieren Sie das folgende Listing zur Erzeugung von neuen Context-Elementen im SBOOK-Knoten.

```

DATA: lo_nd_sbook TYPE REF TO if_wd_context_node,
      lo_el_sbook TYPE REF TO if_wd_context_element.

lo_nd_sbook = wd_context->get_child_node( wd_this->wdctx_sbook ).
lo_el_sbook = lo_nd_sbook->create_element( ).

lo_nd_sbook->bind_element(
  new_item      = lo_el_sbook
  set_initial_elements = abap_false ).

```

Aktionsimplementierung für den Button »Neue Zeile«

Wenden wir uns nun dem Context-Change-Log zu. Standardmäßig ist das Change-Log in jedem Controller deaktiviert. Bei Bedarf muss es daher im jeweiligen Context der Component aktiviert werden. Dies geschieht über das Interface IF_WD_CONTEXT. Sie erhalten die Referenz auf den Context durch den Aufruf der Controller-Methode `wd_context->get_context()`. Anschließend können Sie das Change-Log durch einen Aufruf der Methode `enable_context_change_log()` aktivieren. Öffnen Sie daher im View die Methode `wddoinit()`, und implementieren Sie das folgende Listing.


```

DATA lo_context TYPE REF TO if_wd_context.
lo_context = wd_context->get_context( ).
lo_context->enable_context_change_log( ).

```

Methode »wddoinit()«: Aktivierung des Context-Change-Logs

Wenn Sie nun die Component aktivieren und die Anwendung starten, können Sie durch den Klick auf den Button **Neue Zeile** eine neue Flugbuchung in die SBOOK-Tabelle einfügen und anschließend Buchungen eingeben. Um nun die Änderungen des Change-Logs auszulesen, müssen Sie die Änderungen vom Context abholen. Hierzu können Sie die Methode `get_context_change_log()` des Interface IF_WD_CONTEXT verwenden.

Fügen Sie hierzu in der Toolbar einen zweiten Button mit der Beschriftung **Änderungen auslesen** ein, und legen Sie aus dem Button heraus eine neue Aktion für das Ereignis `onAction` an. Zum Auslesen und Ausgeben des Context-Change-Logs, das in der Praxis typischerweise beim Drücken der Taste  oder vor dem Speichervorgang stattfinden würde, implementieren Sie das folgende Listing.

```

DATA: lo_msg_mgr TYPE REF TO if_wd_message_manager,
      lv_text     TYPE string,
      lv_index    TYPE string,
      lo_context  TYPE REF TO if_wd_context,

```

```

      lt_changes TYPE wdr_context_change_list,
      ls_change  TYPE wdr_context_change.
FIELD-SYMBOLS: <ls_new_value> TYPE any.

* Vorbereitung
lo_msg_mgr =
  wd_comp_controller->wd_get_api( )->get_message_manager( ).

* Lese das Change-Log aus
lo_context = wd_context->get_context( ).
lt_changes =
  lo_context->get_context_change_log( and_reset = abap_true ).

* Gebe die einzelnen Attributänderungen in einer Nachricht aus
LOOP AT lt_changes INTO ls_change
  WHERE change_kind EQ 'A'. " Nur Attributänderungen anzeigen

  ASSIGN ls_change-new_value->* TO <ls_new_value>.
  lv_index = ls_change-element_index.

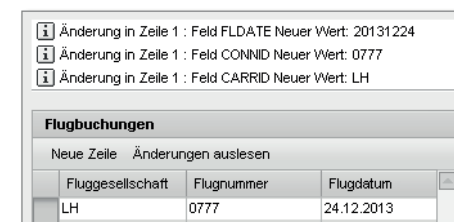
  CONCATENATE 'Änderung in Zeile ' lv_index ': Feld '
    ls_change-attribute_name ' Neuer Wert: ' <ls_new_value>
    INTO lv_text RESPECTING BLANKS.

  lo_msg_mgr->report_message(
    message_text = lv_text
    message_type = if_wd_message_manager=>co_type_info ).
ENDLOOP.

```

Änderungen aus dem Change-Log auslesen und als Nachricht ausgeben

Aktivieren Sie nun die Component, und starten Sie die Anwendung. Fügen Sie eine neue Zeile in die Tabelle ein, und tragen Sie diese Werte ein. Klicken Sie anschließend auf den Toolbar-Button **Änderungen auslesen**, um die Änderungen vom Context-Change-Log abzuholen. Über den Message-Manager werden die attributbezogenen Änderungen (vergleiche mit WHERE-Bedingung `CHANGE_KIND EQ 'A'`) des Change-Logs im Bild ausgegeben.



Ausgabe von Benutzereingaben im Context-Change-Log




Beachten Sie, dass das Context-Change-Log keine programmatischen Änderungen am Context aufzeichnet. Ändern Sie also beispielsweise über die Methode `set_attribute()` an einem Context-Element einen Attributwert, wird diese Änderung nicht automatisch vom Change-Log aufgezeichnet. Möchten Sie programmatische Änderungen am Context aufzeichnen, können Sie hierzu die Methode `add_context_attribute_change()` des Interface `IF_WD_CONTEXT` verwenden.

Tipp 33

Singleton-Eigenschaft verwenden

Sicherlich haben Sie schon einmal vom Singleton-Entwurfsmuster gehört. Dieses stellt die Instanzierung eines einzigen Objekts pro Instanz sicher. Auch Knoten können als Singleton-Knoten eingestellt werden. Doch wieso sollte man nur einen Knoten instanzieren? Die Antwort finden Sie in diesem Tipp.

Jeder Context-Knoten besitzt die Eigenschaft `Singleton`. Nach der Aktivierung der Eigenschaft `Singleton` wird von dem jeweiligen Context-Knoten immer nur eine einzelne Instanz erzeugt, unabhängig von der Anzahl der Context-Elemente des darüberliegenden Knotens. Die Verwendung von Singleton-Knoten ist daher insbesondere bei mehrstufigen Knotenhierarchien mit großen Datenvolumen sinnvoll, da sonst eine vollständige Instanzierung einer umfangreichen Knotenhierarchie entsprechend viel Zeit und ebenso viel Speicherplatz benötigt.

Eigenschaft	Wert	Attribute ...
Knoten		
Knotenname	SFLIGHT	
Dictionary-Struktur	SFLIGHT	
Kardinalität	0..n	
Selection	0..1	
Initialisierung Lead-Selection	<input checked="" type="checkbox"/>	
Singleton	<input checked="" type="checkbox"/>	
Supply-Funktion		

Knoteneigenschaften: Knoten `SFLIGHT` mit aktivem `Singleton`-Kennzeichen

› Und so geht's

Um Ihnen die Verwendung der `Singleton`-Eigenschaft näher zu erläutern, habe ich ein Beispiel mit einer zweistufigen Knotenhierarchie aufgebaut. Die Hierarchie beginnt bei dem Knoten `SCARR`, der auf der gleichnamigen Daten-

banktabelle basiert und dank Kardinalität 0..n beliebig viele Fluggesellschaften beinhalten kann. Der Knoten SCARR liegt direkt unterhalb der Wurzel des Controller-Contexts. Da die Kardinalität jeder Context-Wurzel immer 1..1 ist, sind auch alle direkt unterhalb der Wurzel liegenden Context-Knoten und Context-Elemente per Definition immer Singleton-Knoten, ohne dass Sie die Eigenschaft Singleton explizit aktivieren müssen.

Eine Hierarchiestufe unterhalb von SCARR liegt der Knoten SFLIGHT. Dieser beinhaltet alle Flugverbindungen der jeweils darüberliegenden Fluggesellschaft. Für jede Fluggesellschaft im Knoten SCARR existiert ein eigenes Context-Element, das wiederum seine eigene Knoteninstanz des Sub-Knotens SFLIGHT anlegt. Befinden sich in Knoten SCARR also 100 Fluggesellschaften, werden hierfür 100 SCARR-Context-Elemente und 100 Knoteninstanzen des Sub-Knotens SFLIGHT angelegt. Der Knoten SFLIGHT wiederum beinhaltet so viele Context-Elemente, wie die jeweilige Fluggesellschaft an Flugverbindungen hat. Hat jede der 100 Fluggesellschaften im Jahr 1.000 Flugverbindungen, würden für die vollständige Abbildung der gesamten Hierarchie insgesamt 100.000 Context-Elemente benötigt.

Knotenstruktur zur Designzeit	Daten während der Laufzeit
CONTEXT SCARR (Fluggesellschaft) CARRID CARRNAME SFLIGHT (Flugverbindungen) CONNID FLDATE	CONTEXT SCARR (1) CARRID: LH CARRNAME: Lufthansa SFLIGHT (1.1) CONNID: LH280 FLDATE: 24.12.2013 SFLIGHT (1.2) CONNID: LH4711 FLDATE: 31.12.2013 SCARR (2) CARRID: AA CARRNAME: American Airlines SFLIGHT (2.1) CONNID: ... FLDATE: ...

Zweistufige Knotenhierarchie: Fluggesellschaften und Flugverbindungen

Aktivieren Sie nun in dieser Hierarchie für den SFLIGHT-Knoten die Eigenschaft Singleton, werden auf dieser Hierarchieebene nur noch Context-Elemente des Eltern-Elements mit der Lead Selection instanziiert. Haben Sie also im Knoten SCARR die Fluggesellschaft Lufthansa selektiert, enthält nur der zu Lufthansa gehörende Sub-Knoten SFLIGHT die zugehörigen Context-Elemente mit den Flugverbindungen. Wandert die Lead Selection vom Knoten SCARR zu einer anderen Fluggesellschaft, werden die Context-Elemente des

Lufthansa-SFLIGHT-Knotens freigegeben und neue Elemente im SFLIGHT-Knoten der anderen Fluggesellschaft erzeugt. Durch die Aktivierung der Singleton-Eigenschaft können Sie daher die Anzahl der Context-Elemente eines Sub-Knotens deutlich begrenzen. Die Singleton-Eigenschaft eignet sich besonders für die Verwendung im Zusammenspiel mit mehrstufigen Hierarchien und Supply-Funktionen, die bei Änderung einer Lead Selection auf oberer Ebene gezielt die Daten der zugehörigen Sub-Knoten auslesen.

Inhalt

Einleitung	11
TEIL 1 Tipps zum Einstieg	15
Tipp 1 Testanwendungen einsetzen	16
Tipp 2 Druckdialog implementieren	20
Tipp 3 Eigene Kontextmenüs entwickeln	23
Tipp 4 Gängige Namenskonventionen verwenden	29
Tipp 5 Nachrichten mit Context-Bezug erzeugen	33
Tipp 6 Generierte Konstanten, Datentypen und Methoden verwenden	38
Tipp 7 Pop-up-Fenster erzeugen	41
Tipp 8 Dateien downloaden	45
Tipp 9 SAP-GUI-Transaktionen aus Web-Dynpro-Anwendungen starten	49
Tipp 10 Hilfen einbinden und verwenden	53
Tipp 11 Internationalisierung von Texten	58
Tipp 12 Barrierefreie Anwendungen entwickeln	62
Tipp 13 CSS-Maßeinheiten verwenden	65
TEIL 2 Architektur und Komponenten von Web-Dynpro-Anwendungen	67
Tipp 14 EmptyViews verwenden	68
Tipp 15 Daten zwischen Anwendungen übertragen	71
Tipp 16 UI-Elemente dynamisch umhängen	77
Trick 17 Daten mithilfe des WorkProtect-Modus vor Verlust schützen	80
Tipp 18 Select-Options verwenden	83
Tipp 19 Wiederverwendbare Components erstellen	87
Tipp 20 Component-Interfaces verwenden	89
Tipp 21 POWER-Listen verwenden	93
Tipp 22 Eingaben in Pflichtfeldern prüfen	98
Tipp 23 Drag & Drop verwenden	101
Tipp 24 Ereignisse asynchron empfangen	106

TEIL 3	Web-Dynpro-Context verwenden	111
Tipp 25	Supply-Funktionen einsetzen	112
Tipp 26	Context-Attributeigenschaften verwenden	115
Tipp 27	Context-Knoten zur Laufzeit anlegen	118
Tipp 28	Rekursionsknoten anlegen	121
Tipp 29	Context nicht als Datenablage verwenden	125
Tipp 30	Mapping zwischen Components anlegen	127
Tipp 31	Range-Context-Knoten verwenden	131
Tipp 32	Context-Change-Log verwenden	135
Tipp 33	Singleton-Eigenschaft verwenden	139
TEIL 4	User-Interface-Elemente einbauen	143
Tipp 34	Tastatursteuerung mit Hotkeys	144
Tipp 35	Mit der Tastatur schneller durch Anwendungen navigieren	147
Tipp 36	Automatische Vorschlagswerte unterhalb von Eingabefeldern anzeigen	149
Tipp 37	Texteingabefelder formatieren	151
Tipp 38	Randlose Anwendungen bauen	155
Tipp 39	Anwendungen mithilfe von PageLayout in Bereiche einteilen	157
Tipp 40	Seiten mithilfe des Splitter-Elements aufteilen	160
Tipp 41	Das richtige Layout auswählen	163
Tipp 42	UI-Elemente und Layouts dynamisch generieren	168
Tipp 43	Eigenen HTML-Code und Java-Skripte integrieren	173
Tipp 44	Interaktiven Kartendienst Visual Business verwenden	178
Tipp 45	Seiten mit dem Page Builder erstellen	182
Tipp 46	Multi-Value-Paste in Eingabefeldern	186
TEIL 5	Mit Tabellen arbeiten	191
Tipp 47	Client-Tabellen verwenden	192
Tipp 48	Tabellen dynamisch erzeugen	195
Tipp 49	Tabellen- und Spaltenbreite optimieren	198
Tipp 50	Optimale Spaltenbreite berechnen	200
Tipp 51	Mehrere Zell-Editoren in einer Spalte verwenden	203
Tipp 52	Mehrere Zell-Editoren in einer Zelle verwenden	208
Tipp 53	Mit ALV-Tabellen arbeiten	211
Tipp 54	Leere Zeilen in ALV-Tabellen eingabebereit schalten	215

TEIL 6	Im Editor arbeiten	219
Tipp 55	Quellcode mit dem Code Wizard generieren	220
Tipp 56	Benutzeroberflächen mit dem Code Wizard generieren	223
Tipp 57	Drag & Drop im Window-Editor	228
Tipp 58	Window-Editor-Ansicht wechseln	230
Tipp 59	Parameter von Ereignisbehandlern abgleichen	232
Tipp 60	Root-Element austauschen	235
Tipp 61	Mit Layout-Templates arbeiten	238
Tipp 62	Auf den quelltextbasierten Editor umschalten	241
Tipp 63	Controller-Versionen erzeugen und vergleichen	244
Tipp 64	Web Dynpro in Eclipse entwickeln	247
Tipp 65	Context-Editor-Ansicht wechseln	251
TEIL 7	Web-Dynpro-Anwendungen analysieren	253
Tipp 66	Debugging über das Kontextmenü	254
Tipp 67	Web-Dynpro-Debugger verwenden	257
Tipp 68	Components mit der technischen Hilfe analysieren	260
Tipp 69	Helper-Shortcuts verwenden	263
Tipp 70	Aktionen mit dem Trace-Tool aufzeichnen	265
Tipp 71	Delta-Rendering-Analyse durchführen	269
Tipp 72	Memory-Snapshot anlegen	273
Tipp 73	Performance analysieren	276
Tipp 74	Mit eCATT testen	279
Tipp 75	ABAP Unit Tests in Web Dynpro verwenden	284
TEIL 8	Anwendungen anpassen, konfigurieren und erweitern	289
Tipp 76	Web-Dynpro-Anpassungsebenen verstehen	290
Tipp 77	Delta-Handling von Customizing und Personalisierung richtig anwenden	294
Tipp 78	Components mithilfe von Enhancements erweitern	297
Tipp 79	Systemweite Konfigurationen durchführen	301
Tipp 80	Mandantenweite Anpassungen über Customizing vornehmen	307
Tipp 81	Anwendungen und Components personalisieren	311
Tipp 82	Datumsfelder über Component-Defined-Personalisierung dynamisch füllen	313

Tipp 83	Personalisierung für Endbenutzer deaktivieren	319
Tipp 84	Konfigurationen, Customizing und Personalisierungen analysieren	322
TEIL 9	Administration	325
Tipp 85	Anwendungen ohne Benutzeranmeldung starten	326
Tipp 86	Eigene Fehlerseiten definieren	330
Tipp 87	Sitzungs-Timeout-Zeit erhöhen	333
Tipp 88	HTTP-Komprimierung aktivieren	336
Tipp 89	Globale Web-Dynpro-Einstellungen vornehmen	338
Tipp 90	Anwendungsparameter und deren Funktionsweise verstehen	341
Tipp 91	Web-Dynpro-UIs absichern	350
Tipp 92	Eigene Design-Themes erstellen	353
Tipp 93	Firmenlogo in existierende Anwendungen einbauen	357
Tipp 94	Anwendungen über den Administrationsservice absichern	361
TEIL 10	Mit dem Floorplan Manager arbeiten	365
Tipp 95	Floorplan Manager verwenden	366
Tipp 96	Einfache FPM-Anwendung erstellen	370
Tipp 97	Konfigurationen über den Expertenmodus aufrufen	376
Tipp 98	Konfigurationen über Deep-Copy kopieren	379
Tipp 99	FPM-Workbench verwenden	382
Tipp 100	FPM-Anwendungen mit dem Application-Creation-Tool erzeugen	384
Der Autor		387
Index		389

Index

A

Abmeldeseite 330, 331
Accessibility 62, 342
ACCESSIBILITY_MODE 64, 342
AccessibilityDescription 63
AccessKeys 147
ACF-Whitelist 179
activateAccessKey 147
Administrationsservice 361
Aktion
 abgleichen 233
 anlegen 21
 aufrufen 24
Aktionstaste 144
Als Template speichern 239
ALV 168, 211
 Änderungsmodus aktivieren 216
 Configuration-Model 214, 216
 get_model_extended() 214
 Masseneditormodus aktivieren 216
 ON_LEAD_SELECT 214
Analyse, mit Debugger 257
Anmeldefehler 330
Anmeldeseite 330
Anmeldeverfahren 328
Anpassung 290
Anpassungshierarchie 292
Anwendung
 administrationsrelevant 361
 CONFIGURE_COMPONENT 302, 321
 CUSTOMIZE_COMPONENT 307, 321
 Namenskonvention 30
 randlos 155
 schützen 345
 Test 16
 WD_ANALYZE_CONFIG_APPL 322
 WD_ANALYZE_CONFIG_COMP 323
 WD_ANALYZE_CONFIG_USER 324
 WD_GLOBAL_SETTING 319
Anwendungsfehler 330
Anwendungshierarchie 377
Anwendungshierarchie-Browser 379
Anwendungskonfiguration 183, 293, 348
 Analyse 322
 anlegen 303
 Component zuweisen 305

Anwendungsparameter 261, 341
 /H 254
 Anwendungskonfiguration 342
 sap-wd-perfMonitor 276
 Sicherheit 351
 WDCONFIGURATIONID 306
 WDDISABLEUSERPERSONALIZATION 320
anwendungsübergreifende
 Systemeinstellung 342
Application-Creation-Tool 384
ARIA 342
Attribut, Namenskonvention 31
Attributeigenschaft 115

B

Barrierefreiheit 62, 342
Barrierefreiheitsmodus 338
Benutzeranmeldung 326
Benutzereinstellung 311, 312
Benutzerparameter 341
Benutzerverwaltung 350
Berechtigung 263
 S_DEVELOP 338
 S_WDR_P13N 338
Breakpoint, externer 257
Breitenangabe 65
Browser, unterstützter 342
Built-In-Anpassung 291
BusinessGraphics 66
Button 47

C

cellDesign 164
cellPadding 164
cellSpacing 164
CHIP 182
 anlegen 185
 Wires 182
CHIP-Katalog 184
CL_ABAP_CHAR_UTILITIES 153
CL_ABAP_UNIT_ASSERT 287

CL_SALV_WD_CONFIG_TABLE 214
CL_VBC_GEOCODER 181
CL_VSI 352
CL_WD_ 169
CL_WD_C_TABLE 196
CL_WD_COMPONENT_ASSISTANCE 59
CL_WD_DYNAMIC_TOOL 196
 check_mandatory_attr_on_view() 100
 create_c_table_from_node() 196
 create_table_from_node() 196
CL_WD_FLOW_DATA 170
CL_WD_HTML_CONTAINER 176
CL_WD_MATRIX_LAYOUT 171
CL_WD_RUNTIME_SERVICES 51
CL_WD_RUNTIME_UTILITIES 47
CL_WD_UTILITIES
 construct_wd_url() 50
 get_otr_text_by_alias() 61
CL_WD_WEB_DYNPRO_TESTER 285
Client-Tabelle 102, 192, 193, 204, 209
Client-Zeit 277
Code Wizard 220, 223, 240
colCount 167
collapseDirection 161
cols 153
colSpan 164, 166
columnSizeMode 199, 200
Component
 aktive 260
 Namenskonvention 29
 wiederverwendbare 87
Component-Defined-Anpassung 291
Component-Interface 88
 implementieren 91
Component-Konfiguration 183, 293
 Namenskonvention 29
Component-Verwendung 40, 212
 aktive 261
Configuration-Controller 296, 315
Context
 Component-übergreifendes Mapping 127
 Mapping 126
 richtige Verwendung 125
 Sicherheit 351
CONTEXT_MENUS 24, 26, 314
Context-Attribut
 enabled 115
 readOnly 115
 required 115
 visible 115

Context-Attributeigenschaft 115
Context-Change-Log 135
 aktivieren 136
 Änderung ergänzen 138
 auslesen 136
Context-Element, Namenskonvention 30
Context-Knoten
 dynamisch anlegen 118
 Interface 88
 Namenskonvention 30
 Range-Knoten 131
 rekursiver 121
 Singleton 139
contextMenuBehaviour 25, 27
contextMenuId 24, 25
ContextualPanel 66
Controller, Selbstreferenz 39
CSS-Datei 175
CSV-Datei 45
CTable 102, 192, 193, 204, 209
CTableMultiEditorCell 209
CTableStandardCell 205
CTableSymbolCell 203, 205
Custom-Controller
 anlegen 315
 Namenskonvention 30
Customizing 291, 307, 358
 während der Laufzeit 308
Customizing-Editor 307, 358

D

Datei, downloaden 45
Datentyp, generierter 38
Datentypprüfung 351
Debugger 254
 einschalten 254
 über das Kontextmenü 255
 Web-Dynpro-Tool 257
Deep-Copy-Modus 379
DEFAULT_HOST 363
dekoratives Element 357
Delta-Rendering 264, 269
 Funktionsweise 269
design 66, 153
Design-Guideline 347
Dirty Flag 269
Drag & Drop 101, 228

Drag Source 101
Drop Target 101
DropTargetInfo 102
Druckansicht 20
dynamische Programmierung 168

E

eCATT 279, 284, 344, 345
 Aufzeichnung 280
 Skript-Editor 281
eCATT-Editor 280
Editor 241
Eigenschaft
 contextMenuBehaviour 315
 contextMenuId 315
 explanation 344
 keyToSelect 315
 onSelect 317
 selectedKey 315
Eingabehistorie 150, 338
 ein-/ausschalten 343
Einstellung, anwendungsübergreifende 338
element_<Knotenname> 39
elements_<Knotenname> 39
EMPTYVIEW 70
EmptyView 68
enabled 116
End2End-Zeit 276
Enhancement Framework 290
Enhancement → Erweiterung
Ereignis, Parameter 232
Ereignisbehandler 100
 Namenskonvention 30
 Parameter 232
Erweiterung 290, 297, 357
 anlegen 360
Excel-Export 211
Exit-Plug, Sicherheit 351
Expertenmodus 376
Explanation 55

F

F1-Hilfe 53, 260
F4-Hilfe 53
Farbe ändern 355

Favoritensymbol 344
Feeder-Klasse 94, 368, 371
 Parameter 374
Fehlerseite 330, 331
 eigene 332
FileDownload 47
FileUpload 48, 352
fire__plg() 40
FIRST_TIME 169
FlashIsland 62
Floorplan 366, 385
Floorplan Manager (FPM) 101, 265, 293,
 295, 366
FlowLayout 163, 169
flush() 372
FormattedTextEdit 74
FormData 167
FormHeadData 166
FormLayout 166, 224
FormLayoutAdvanced 167
FormTopData 166
Formular, generieren 224
fpm_cfg_bo_model_act 384
fpm_cfg_hierarchy_browser 379
FPM_CONFIG_EXPERT 376
FPM_FORM_UIBB_GL2 369
FPM_GAF_COMPONENT 369
FPM_LIST_UIBB 369
FPM_OVP_COMPONENT 369, 372
FPM-Anwendung anlegen 384
FPM-Expertenmodus 376
FPM-Workbench 355, 382

G

GAF 366
Generierung von Formularen 224
Generierung von Quellcode 220
Generierung von Tabellen 225
GeoMap 66, 178
get_context_menu() 25
GridLayout 164
Group 168
GUIBB 367
Guided Activity Floorplan 366

H

handleHotkeys 146
height 65, 180
Help Center 55, 344
 starten 57
Hilfe, Tastenkombination 53
Hilfe-Link 56
Hilfemodus 54
Hilfsklasse 94
Hintergrundbild einfügen 355
Höhenangabe 65
Hotkey 144
HtmlContainer 173
HtmlFragment 173
HTML-Integration 173
HtmlIsland 173
HTTP-Komprimierung 336
HTTP-Port 361
HTTPS 345
HTTPWatch 350

I

ICF 326, 352
 nicht erreichbar 331
 Service deaktivieren 352
 Servicehierarchie 331
ICF-Knoten 362
ICF-Service 333
 HTTP-Komprimierung 336
ICF-Serviceknoten 326
ICM-Profilparameter
 icm/conn_timeout 335
IF_FPM_GUIBB_LIST 371
if_fpm_guibb_list~get_data() 371
if_fpm_guibb_list~get_definition() 371
if_wd_application 21
if_wd_component 21
IF_WD_COMPONENT_USAGE 92
IF_WD_CONTEXT
 enable_context_change_log() 136
IF_WD_CONTEXT_ELEMENT
 set_attribute_property() 117
IF_WD_CONTEXT_NODE_INFO 118, 351
 add_new_child_node() 132
IF_WD_CONTROLLER 316
IF_WD_MESSAGE_AREA 34
IF_WD_MESSAGE_MANAGER 35

IF_WD_PERSONALIZATION 316
IF_WD_PORTAL_INTEGRATION 52, 80
IF_WD_RR_WINDOW 155
IF_WD_SELECT_OPTIONS_20 84
IF_WD_VIEW 169
if_wd_view_controller 22
IF_WD_WINDOW_CONTROLLER 34, 156
IF_WD_WINDOW_MANAGER 42
IFRAME 173
IG_ 40
Image, source 360
Info-Objekt 55
InputField 59, 149
 dynamische Erzeugung 171
 Eigenschaft 151
 Formatierung 151
INTERFACECONTROLLER 90
Interface-Controller 88
Interface-View 88
Internationalisierung 58
InvisibleElement 164

J

JavaScript
 Bibliothek 173
 Integration 175

K

Kartendienst 178
Klasse, Shared-Memory-fähig 73
Komponentenmodell 87
Konfiguration 291
 Analyse 322
 anlegen 301
 Anwendung 370
 Built-In 296, 309
 Component 370
 Component-Defined 296
 Delta-Handling 294
 kopieren 379
Konfiguration anlegen/ändern 301, 304, 372
Konfigurationsdaten 320
 explizite 323
 implizite 323

Konfigurations-Editor 296, 302
 Anwendung 304
Konfigurationseigenschaft 319
Konfigurierbare Bereiche anzeigen 378
Konstante, generierte 38
Kontextmenü
 anlegen 23, 26, 314
 Beispiel 23

L

Label 59
 Ausrichtung 346
Launchpad 382
Layout 163
 Eigenschaft 169
layoutContainer 63
Lead-Selection
 Client-Tabelle 194
length 152
Lightspeed-Rendering 344
LinkToAction 47
LinkToURL 50
Logo einbinden 357
lokale Testklasse 285

M

Manipulation (Daten) 350
Mapping 127
 aktualisieren 126
 einfaches 128
 externes 128, 129
 richtige Verwendung 129
Maßeinheit 65
MatrixData 164
MatrixHeadData 164
MatrixLayout 164, 169, 224
Mehr Feldhilfe 53, 77
Meldung, halten 22
Memory 276
Memory Inspector 274
Memory-Snapshot 273
Menu 24
MenuItem 24, 26
 dataSource 25
MenuCheckBox 25
MenuRadioButton 25, 315

MenuSeparator 25
MESSAGE-Anweisung 33
Message-Manager 33
Methode, generierte 38
Methodendefinition 243
MIME-Objekt
 anlegen 175
 importieren 360
Monospace 151
MultiMenuItem 25

N

Nachricht, mit Context-Bezug 33
Namenskonvention 29
Navigation 99
Navigationslink, Drag & Drop 229
newRow 167
noHistory 150

O

onDrop 102, 104
Online Text Repository (OTR) 58
 Alias 60
 Text anlegen 60
 Texte auswählen 60
onResize 161
onSelect 104
orientation 161
Overwrite-Exit 299
OVP-Floorplan 366, 370

P

Padding 155
Page Builder 182
PageHeader 157, 360, 367
PageLayout 157
Panel 16, 77
Parameter, Auswertungsreihenfolge 341
Performance
 Client 336
 Rendering 269
Performanceanalyse 276
Performance-Monitor 264, 274, 276
 Registerkarte 277

Personalisierung 291, 311
 Built-In 311, 313
 Component-Defined 313
 deaktivieren 319, 339, 346
 eigener Personalisierungsdialog 313
Pflichtfeldprüfung 98
Phasenmodell 99
Plug 69
 Namenskonvention 30
Pop-up
 Bestätigungsfenster 43
 erzeugen 41
 öffnen 43
 Standard-Button 43
 technische Hilfe 262
Portal-Navigation
 Sicherheit 351
Post-Exit 299
POWL 93
 Abfrage 96
 Anwendung 95
 Default-Abfrage 96
 Easy POWL 93
 Kategorie 96
 Registrierungstyp 96
 Typ 96
Pre-Exit 299
Primärattribut 295
primäre Eigenschaft 59
print_page() 21
PROCESS_ADM_SERVICES 362

Q

Quellcode
 generieren 220
Query-Klasse 93
Quickview 343
Quirks-Modus 344

R

raise_evt() 40
randlose Anwendung 155
readOnly 115, 116
REASSIGN_ADMSERVICES 364
Registerkarte
 implementierte Interfaces 91

Implementierung 242
 Verwendete Components 91
Rekursionsknoten 121
Response Time 276
Root-Element austauschen 158, 235, 236
ROOTUIELEMENTCONTAINER 158,
 169, 235
RowData 165
rowDesign 165
rowDragInfo 102, 104
RowHeadData 165
RowLayout 165
rows 152, 153
rowSpan 167

S

SALV_WD_TABLE 211
SAP Floorplan Manager 160
SAP GUI
 für HTML 50
 Transaktion starten 49
SAP NetWeaver Business Client (NWBC)
 51, 145
SAP-ACCESSIBILITY 63, 342
sap-accessibility-debug 64
SAP-ACCESSIBILITYMODE 342
sap-config-mode 358
SAP-LS-USEANIMATION 347
SAP-THEME 347
sap-theme 356
SAP-WD-CONFIGID 348
SAP-WD-DELTARENDERING 343
SAP-WD-LIGHTSPEED 344
SAP-WD-REMOTEDESKTOP 345
SAP-WD-STABLEIDS 345
SAP-WD-SUPPORTSFULLHEIGHT 349
sap-webdav-themeroot 356
SashPosition 161
SashPositionMode 161
sashType 161
Schrift bearbeiten 355
scrollableColCount 198, 200
ScrollContainer 159
Security Guide 352
Seite konfigurieren 377
Seitenbereich, Aufteilung 157
selectedCellVariant 205, 209
selectionMode 194

Select-Options 83
Selenium Recorder 264
Serveradministration 350
Server-Zeit 277
Session-Timeout 333
 erhöhen 334
Shared Memory 72, 345
Shared Object 72
Shortcut 263
SICF 364
Sicherheit 343
Sicherheitshinweis 350
Sichtbarkeit 311
Side Panel 339
 Breite 348
 einstellen 348
SilverlightIsland 62
Singleton 112, 113, 140
Soforthilfe anzeigen 54
SOTR_VOCABULARY_BASIC 60
Spaltenbreite 199
 dynamisch berechnen 200
Splitter 160
SQL-Attacke 351
Standardmodus 344
Standardwert setzen 312
state 98, 116
staticHtml 174
Stylesheet 174, 339, 347
 erzwingen 347
Suchhilfe 150
Suchmaske → Select-Options
SuggestValues 149
Supply-Funktion 46, 102, 112
 Range-Knoten 132
Supply-Methode, Namenskonvention 30
Swap Root-Element 236
Systemanmeldung 350
Systemdaten anzeigen 261

T

T100 58
Tabelle
 clientseitiges Scrollen 198
 dynamische Höhenbestimmung 194
 eingabebereite Leerzeilen 215
 Filtern 211
 generieren 225

Generierung während der Laufzeit 195
 mehrere Zell-Editoren 203
 sortieren 211
TABLE 192
TableMultiEditorCell 209
tags 103
Tastaturnavigation 147
Tastenkombination 144
technische Feldhilfe 298
technische Hilfe 260, 359
Template 238
Test, automatisierter 279
Test-Component 16
TextEdit 151
Theme 353, 357
 freigeben 355
Theme Designer 353
Theme-ID 356
Timeout-Zeit 333
Tooltip 63, 145
Trace herunterladen 268
Trace-Tool 264, 265, 271
 Delta-Rendering 270
 starten 265
Transaktion
 /UI5/THEME_DESIGNER 353
 FPM_WB 382
 LPD_CUST 382
 POWL_CAT 96
 POWL_COCKPIT 95
 POWL_EASY 94
 POWL_QUERY 96
 POWL_QUERYR 96
 POWL_TYPE 96
 POWL_TYPER 96
 S_MEMORY_INSPECTOR 274
 SECATT 281
 SHMA 72
 SICF 326, 334, 336
 WD_TRACE_TOOL 265
TransparentContainer, dynamische
 Erzeugung 169
Tray 77
Tree 122
 dataSource 123
 rootVisible 123
 TreeItemType 123
 TreeNodeType 123
TreeByNestingTableColumn 122
Trennbalken 160

U

- Übersetzung 58
- UI5 353
- UIBB 367
 - Freestyle 367
 - GUIBB 367
 - hinzufügen 373
 - konfigurieren 374
- UI-Element
 - deaktivieren 346
 - dynamische Erzeugung 169
 - dynamische Generierung 168
 - eigenes 173
 - Namenskonvention 31
 - Sichtbarkeit 115
 - umhängen 77
- UI-Guideline 2.0 83
- unbeabsichtigte Navigation 80
- Unit Test 284, 345
- URL-Parameter 340, 341
 - sap-config-mode 308
 - sap-wd-configId 306
 - Sicherheit 351

V

- ValueComparison 66
- VBC_WDC_GEOMAP_GEN2 180
- Version
 - erzeugen 244
 - vergleichen 246
- Versionsverwaltung 244
- vGutter 164, 165
- View
 - Default-View 70
 - einbetten 228
 - Interface-View anlegen 90
 - Personalisierung 24
 - randloser 155
 - Sichtbarkeit 68
- ViewContainer 68
- ViewContainerUIElement 69
- View-Element anzeigen 261
- virtueller Host 363
- Virus Scan Interface 352
- visible 115, 116
- visibleRowCount 198
- Visual Business 178

- Vorlage 238
- Vorschlagsliste 149
- Vorschlagswert 338, 343

W

- Währungsfeld 345
- wd_context → get_context() 136
- wd_cpifc_ 40
- WD_GLOBAL_PARAMETERS 339
- WD_GLOBAL_SETTING 339
- WD_GLOBAL_SETTINGS 341
- WD_SELECT_OPTIONS_20 83
- wd_this 39
- WDACCESSIBILITY 64, 342
- WDACCESSIBILITYMODE 342
- WDALLOWMULTIPLEACTIONS 343
- WDALLOWQUICKVIEWS 343
- WDALLOWUSERAGENTS 342
- WDALLOWVALUESUGGEST 150, 343
- WDCC_ 29
- WDCONFIGURATIONID 348
- wdctx_<Knotenname> 39
- WDDELTARENDERING 343
- WDDISABLEDYNAMICRESOURCESDN 343
- WDDISABLEUSERPERSONALIZATION 346
- wddobeforeaction() 99
- wddomodifyview() 79, 169
- wddooncontextmenu() 24
- WDENABLEFIELDHISTORY 150, 343
- WDENABLEUIELEMENTSHIDE 346
- WDENABLEXBCMLCLIENT 344
- WDFAVICON 344
- WDHELPCENTERDISPLAY 55, 344
- WDHIDEEXPLANATION 344
- WDHIDEMOREFIELDHELPASDEFAULT 54, 344
- WDLIGHTSPEED 344
- WDPREFERREDRENDERING 344
- WDPROTECTEDAPPLICATION 345
- WDR_ACF_WLIST 179
- WDR_CHIP_CATALOG 184
- WDR_CHIP_PAGE 183, 348
- WDR_SELECT_OPTIONS 83
- WDR_TEST_ 17
- WDR_TEST_HELP 56
- WDREFFIELDBEHAVIOUR 345
- WDREMOTEDESKTOP 345
- WDSHAREDREPOSITORY 345

- WDSIDEPANELCONFIGURATIONID 348
- WDSIDEPANELOPEN 348
- WDSIDEPANELREMOTECONSUMER 348
- WDSIDEPANELREMOTEPRODUCER 348
- WDSIDEPANELRESIZABLE 348
- WDSTYLE_LABELALIGNMENT 346
- WDSTYLE_TOOLBARDESIGN 346
- WDSTYLE-LABELALIGNMENT 346
- WDSTYLE-TOOLBARDESIGN 346
- WDSUPPORTSFULLHEIGHT 349
- WDTHEMEROOT 347
- WDUIGUIDELINE 347
- WDUSEANIMATION 347
- WDUSEEXTERNALSTYLESHEET 347
- Web-Dynpro-Component-Interface 90
- WebGUI 50
- width 65, 180
- Window
 - Namenskonvention 30
 - randloses 155
- Window-Editor 228
 - Ansicht 230

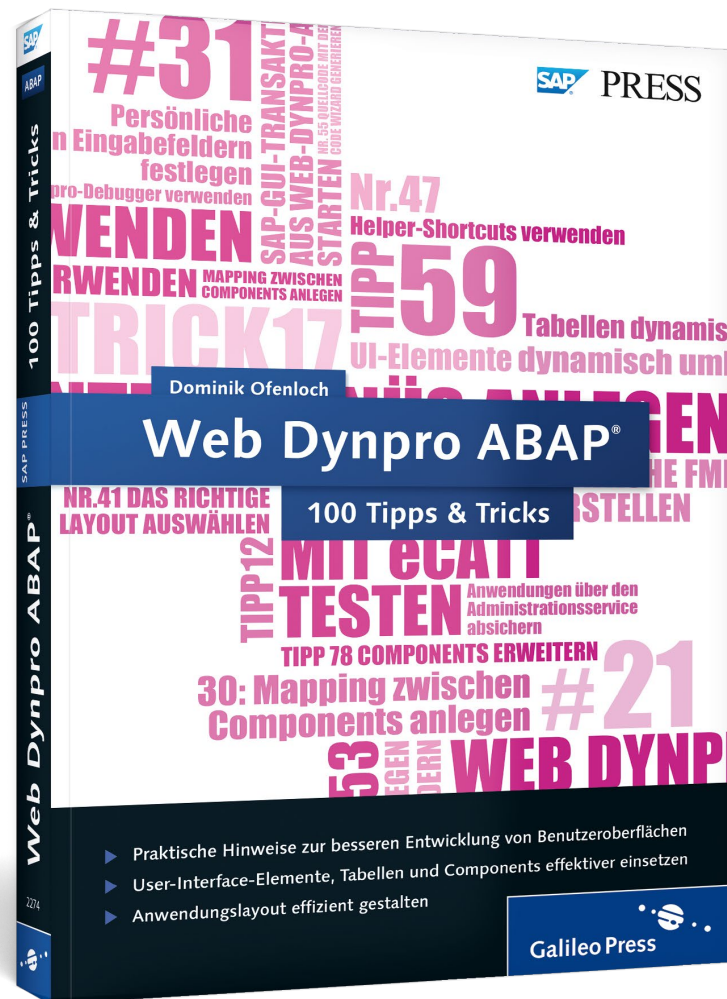
- Java-basierter 231
- Window-Manager 41
- Window-Struktur, Drag & Drop 228
- WorkProtect-Modus 80
 - Variante 81
- wrapping 153
- WTS-Anzeige 345

X

- XML-Anzeige 323

Z

- Zeichenbreite, feste 152
- Zeilenumbruch 153
- Zell-Editor 203
 - mehrere in einer Zelle 208
- Zell-Variante 203
- Zugangstaste 147



Dominik Ofenloch

Web Dynpro ABAP – 100 Tipps & Tricks

EPUB-Format, 397 Seiten*, in Farbe, Dezember 2013
44,90 Euro, ISBN 978-3-8362-3161-9

*auch erhältlich als gedrucktes Buch: 49,90 Euro, ISBN 978-3-8362-2274-7



Dominik Ofenloch studierte an der Dualen Hochschule Mannheim Wirtschaftsinformatik und programmierte bereits während seines Studiums bei der SAP AG in Walldorf verschiedene UI-Technologien. Nach erfolgreichem Studienabschluss im Jahr 2006 begann er seine Karriere bei SAP in der SCM-Entwicklung. Dort entwickelte er für das SAP Transportation Management Web-Dynpro-Benutzeroberflächen. Im Jahr 2009 wechselte er in die SAP-Beratung. Bis Mitte 2013 war er zunächst als CRM- und IS-U-Berater, später als Berater im SAP Transportation Management aktiv. Seit Juli 2013 arbeitet er in der UI-Entwicklung des SAP Floorplan Managers, der auf Web Dynpro basiert. Dominik Ofenloch ist Koautor der SAP PRESS-Bücher Einstieg in Web Dynpro ABAP und Web Dynpro ABAP – Das umfassende Handbuch.

Wir hoffen sehr, dass Ihnen diese Leseprobe gefallen hat. Gerne dürfen Sie diese Leseprobe empfehlen und weitergeben, allerdings nur vollständig mit allen Seiten. Die vorliegende Leseprobe ist in all ihren Teilen urheberrechtlich geschützt. Alle Nutzungs- und Verwertungsrechte liegen beim Autor und beim Verlag.

Teilen Sie Ihre Leseerfahrung mit uns!

