Einfache und effiziente Interaktivität des Benutzers mit dem User Interface ist das höchste Ziel. Drag & Drop ist ein effektiver Ansatz, um diese Anforderungen zu erfüllen. In diesem Kapitel erfahren Sie, wie Sie Drag & Drop in Web Dynpro ABAP einsetzen können.

5 Drag & Drop für UI-Elemente

Interaktionsform

Mithilfe der Drag-&-Drop-Funktion in Web Dynpro ABAP können Sie den Benutzern intuitive und einfache Interaktionsformen anbieten. Unter *Drag & Drop* versteht man jene Benutzerinteraktion, die ein UI-Element oder die Einträge eines UI-Elements – die sogenannte *Quelle* (Source) – auf das gleiche oder ein anderes UI-Element zieht, das als *Ziel* (Target) bezeichnet wird. Zur Durchführung dieser Interaktion dient ein Eingabegerät – z. B. die Maus.

Barrierefreie Nutzung [«]

Beachten Sie, dass die Barrierefreiheit einer Web-Dynpro-Anwendung erfordert, dass die Interaktionen, die mit Drag & Drop durchgeführt werden können, auch barrierefrei sein müssen.

Benutzer

Im Fall der Drag-&-Drop-Operation mithilfe der Maus lässt sich das Vorgehen aus Sicht des Benutzers folgendermaßen beschreiben:

- Der Benutzer klickt ein UI-Element (die Quelle bzw. *Drag Source*) an und hält die Maustaste gedrückt.
- Dem Benutzer wird durch ein Geisterbild (*Ghost Image*) angedeutet, dass er gerade eine Drag-&-Drop-Operation ausführt. Bewegt der Benutzer, immer noch mit gedrückter Maustaste, den Mauszeiger, bewegt sich auch das Geisterbild mit. Dieser Vorgang wird als Ziehen (*Drag*) bezeichnet.
- Bewegt der Benutzer den Mauszeiger auf ein gültiges UI-Element, wird dies optisch z. B. durch eine Umrandung des Ziels (*Drop Target*) angedeutet. Weist der Mauszeiger auf ein ungültiges Ziel hin, wird dies durch ein »Verboten«-Bild visualisiert.

- Falls sich der Benutzer über einem gültigen Ziel befindet und die Maustaste loslässt, wird beim Ziel ein Ereignis ausgelöst und die vom Entwickler beabsichtigte Funktion ausgeführt.

Entwickler

Aus Sicht des Entwicklers sind für die Implementierung einer Drag-&-Drop-Operation einige Schritte zu erledigen. Diese gestalten sich nach den eingesetzten UI-Elementen und den Operationsmodi (siehe Abschnitt 5.9) und lassen sich wie folgt zusammenfassen:

- Für das UI-Element, das als Quelle dienen soll, muss eine `DragSourceInfo` im View Designer definiert werden.
- Die `DragSourceInfo` bietet die Eigenschaft `data` an, mit deren Hilfe Sie Daten übergeben und später bei der Behandlung im Ziel verwenden können.
- Für das UI-Element, das als Ziel dienen soll, muss eine `DropTargetInfo` im View Designer definiert werden.
- In der Eigenschaft `tag` der `DragSourceInfo` bzw. der `DropTargetInfo` definieren gleiche Bezeichner (*Flavor*) die Verbindung von Ziel und Quelle.
- Wird ein Element auf einem Ziel fallengelassen, kann dies das Ereignis `onDrop` auslösen. Der Entwickler ist für die Implementierung der Reaktion auf das Ereignis verantwortlich, also z. B. Sortieren, Löschen etc.

UI-Elemente für Drag & Drop

Die Elemente, die im Web-Dynpro-Kontext die Drag-&-Drop-Funktion unterstützen, sind in Tabelle 5.1 aufgelistet. In der Spalte UI-Elemente sehen Sie die Namen der UI-Elemente und Aggregate, in der Spalte Rolle, ob das UI-Element in einer Drag-&-Drop-Operation die Quelle oder das Ziel darstellt, und in der Spalte Beschreibung eine Erläuterung zu der Operation.

UI-Elemente	Rolle	Beschreibung
`Tree`: Knoten/Blatt	Quelle	Einzelne Knoten und Blätter können bewegt werden.
`Tree`: Knoten/Blatt	Quelle	Mehrere Knoten können selektiert werden, falls sie zumindest einen gemeinsamen Flavor besitzen.

Tabelle 5.1 UI-Elemente mit Drag-&-Drop-Rollen

UI-Elemente	Rolle	Beschreibung
`Tree`: Knoten	Ziel	Knoten können zwischen Kindknoten, an erster und letzter Stelle in einem Elternknoten und als neue Unterknoten unter einem Elternknoten eingefügt werden.
`GridLayout`: Layoutzelle	Quelle	Einzelne Zellen können bewegt werden. Voraussetzung dafür ist ein Griff (Handle), z. B. `Image`, `SectionHeader`, `Caption`. Einschränkung: Es ist nur ein einspaltiges `GridLayout` möglich.
`GridLayout`: Layoutzelle	Ziel	UI-Elemente können zwischen den Zellen abgelegt werden. Einschränkung: Es ist nur ein einspaltiges `GridLayout` möglich.
`MatrixLayout`: Layoutzelle	Quelle	Einzelne Zellen können bewegt werden. Voraussetzung dafür ist ein Griff (Handle), z. B. `Image`, `SectionHeader`, `Caption`. Einschränkung: Es ist nur ein einzeiliges `MatrixLayout` möglich.
`MatrixLayout`: Layoutzelle	Ziel	UI-Elemente können zwischen den Zellen abgelegt werden. Einschränkung: Es ist nur ein einzeiliges `MatrixLayout` möglich.
`Image`	Quelle	Das Bild kann bewegt werden.
`Image`	Ziel	Auf das Bild kann ein anderes UI-Element bewegt werden.
`ItemListBox`	Quelle	Einzelne oder mehrere Einträge können bewegt werden.
`ItemListBox`	Ziel	Einträge können zwischen bestehenden Einträgen sowie an der ersten und letzten Position in der Liste eingefügt werden.
`Table`: Zeilen	Quelle	Einzelne Tabellenzeilen können bewegt werden.
`Table`: Zeile	Ziel	Einträge können zwischen bestehenden Einträgen und an der ersten und letzten Position eingefügt werden. Zudem können Elemente auf eine Tabellenzeile gezogen werden.
`CTable`: Zeilen	Quelle	Einzelne Tabellenzeilen können bewegt werden.

Tabelle 5.1 UI-Elemente mit Drag-&-Drop-Rollen (Forts.)

UI-Elemente	Rolle	Beschreibung
`CTable`: Spalten	Quelle	Einzelne Spalten können bewegt werden.
`CTable`: Zeilen	Ziel	Einträge können zwischen bestehenden Einträgen und an der ersten und letzten Position eingefügt werden. Zudem können Elemente auf eine Tabellenzeile gezogen werden.
`CTable`: Spalten	Ziel	Einträge können zwischen, vor und nach Spalten eingefügt werden. Zudem können Elemente auf eine Tabellenspalte gezogen werden.
`Accordion`	Quelle	Einzelne Items können bewegt werden.
`Accordion`	Ziel	Zwischen Items sowie am Anfang und Ende der Items können Elemente eingefügt werden. Zudem kann ein Element auf ein Item gezogen werden.
`PanelStack`	Quelle	Ein ganzer `PanelStack` kann als ein Objekt gezogen werden.
`PanelStack`	Ziel	Einzelne Panels können auf den Stack gedroppt werden.
`DropTarget`	Ziel	Dabei handelt es sich um einen Wrapper für UI-Elemente, die keine generische Drag-&-Drop-Unterstützung anbieten.
UI-Element-Eigenschaft `isDragHandle`	Quelle	Die UI-Elemente `Image`, `SectionHeader` und `Caption` können als Griff für den Drag verwendet werden.
Ereignis `onDrop`	Auslöser	Das Ereignis `onDrop` löst den Aufruf der Behandlermethode zu einem Drop aus. In der Behandlermethode werden die Reaktionen auf den Drop implementiert.

Tabelle 5.1 UI-Elemente mit Drag-&-Drop-Rollen (Forts.)

Im folgenden Abschnitt besprechen wir die Elemente `DragSourceInfo`, `DropTargetInfo`, `DropTarget` und das Ereignis `onDrop`, die die Grundbausteine der Drag-&-Drop-Mechanismen darstellen. In den weiteren Abschnitten 5.2, »Tree«, bis 5.8, »PanelStack«, diskutieren wir im Detail die UI-Elemente, die für Drag & Drop verwendet werden können. Dabei beschreiben wir jeweils die Rolle des UI-Elements als Quelle, als Ziel und die Behandlung des »Fallenlassens« auf dieses Element anhand von Beispielen. In Abschnitt 5.9, »Operationsmodi«, gehen wir auf die unterschiedlichen Verwendungsvarianten von Drag & Drop ein.

5.1 Allgemeines

Bausteine

Die Implementierung von Drag-&-Drop-Operationen baut auf elementaren Bausteinen auf:

- Die `DragSourceInfo` dient als Information für die Quelle der Drag-&-Drop-Operation.
- Die `DropTargetInfo` sammelt die Informationen über das Ziel der Drag-&-Drop-Operation, wobei für die Sammlung der Zielinformation auch das Drop Target verwendet werden kann.
- Mithilfe des Ereignisses `onDrop` kann ein Aktionsbehandler festgelegt werden, der die Implementierung der Reaktion auf das Fallenlassen eines Elements auf ein anderes Element beinhaltet.

Im Folgenden geben stellen wir Ihnen einige Beispiele, die als Anschauungsmaterial für die Verwendung der Bausteine dienen können.

5.1.1 DragSourceInfo

Quelle

In den UI-Element-Eigenschaften der `DragSourceInfo` (❶ in Abbildung 5.1) werden alle Informationen gesammelt, die zu der Quelle einer Drag-&-Drop-Operation gehören, d. h. einer *DragSource*. Auf der linken Seite wird das UI-Element `ItemListBox` ❷, für das die `DragSourceInfo` definiert wird, in der Designdarstellung gezeigt. Auf der rechten Seite sind die dazugehörenden UI-Elemente in der UI-Elemente-Hierarchie mit den Eigenschaften zur `DragSourceInfo` abgebildet.

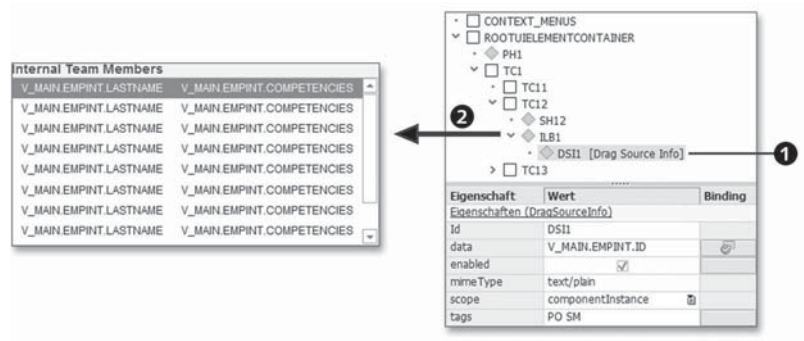

Abbildung 5.1 DragSourceInfo

Eigenschaften

- `Id`
  Der Wert der Eigenschaft `Id` legt die eindeutige Bezeichnung der `DragSourceInfo` fest (siehe Abschnitt 4.1, »Eigenschaften für alle UI-Elemente«). Diese kann in weiterer Folge beim Ziel verwendet werden, um festzustellen, von welcher Quelle (Source) aus die Drag-&-Drop-Operation gestartet wurde.
- `data`
  Mithilfe der Eigenschaft `data` können Sie Daten hinterlegen, die bei der Drag-&-Drop-Operation »transportiert« werden sollen. Diese Daten können vom Ziel (Target) der Drag-&-Drop-Operationen verwendet werden. Zum Beispiel könnte die ID eines Teammitglieds transportiert und im Ziel für die Ermittlung der Abrechnungsergebnisse herangezogen werden.

  Achten Sie darauf, dass keine sicherheitsbedenklichen oder voluminösen Daten übergeben werden, da Drag-&-Drop-Operationen auf dem Client ausgeführt werden. Aus sicherheitstechnischen Gründen bietet es sich an, die Daten nicht direkt (z. B. den Namen eines Teammitglieds), sondern indirekt (z. B. die Personalnummer) als Referenzen zu übertragen. Wie Sie in Abbildung 5.1 erkennen können, haben Sie die Möglichkeit, die Eigenschaft `data` an den Context zu binden. Damit können Sie z. B. zeilenbezogene Daten für Tabellenzeilen hinterlegen.
- `enabled`
  Die Eigenschaft `enabled` steuert, ob die Drag-&-Drop-Operation aus Sicht der Quelle aktiviert ist oder nicht. Diese Eigenschaft kann auch per Data Binding an ein Context-Attribut angebunden werden und ist somit während der Laufzeit aufgrund der Datenänderung im Context änderbar.
- `mimeType`
  Die Eigenschaft `mimeType` definiert die Art der Daten, die an das Ziel übertragen werden. Defaultmäßig wird der Wert `text/plain` vorgegeben. Sie wird im zugrunde liegenden Release nicht vom Web-Dynpro-Framework umgesetzt und ist für zukünftige Verwendungen vorgesehen.
- `scope`
  Die Eigenschaft `scope` legt die Reichweite der `DragSourceInfo` fest. Wird dieser Eigenschaft der Wert `componentInstance` zugewiesen, kann die `DragSourceInfo` nur in der Component verwendet werden, in der sie auch definiert wurde. Wurde der Wert `global` zugewiesen, kann die `DragSourceInfo` über Component-Grenzen hinweg verwendet werden.
- `tags`
  Die Eigenschaft `tags` (manchmal auch als *Flavor* bezeichnet) der `DragSourceInfo` ermöglicht es Ihnen, zu definieren, welche Ziele für diese Quelle relevant sind. Pro Tag können ein oder mehrere Bezeichner definiert werden, je nachdem, wie das Zusammenspiel zwischen UI-Elementen gewünscht ist.

  Falls ein Bezeichner aus der Source mit einem Bezeichner aus dem Target komplett oder in Teilen übereinstimmt, ist das Zusammenspiel möglich. Werden mehrere Bezeichner für ein Tag festgelegt, müssen diese durch Leerzeichen getrennt werden. Die Groß- und Kleinschreibung ist relevant für die Bezeichner. Außerdem dürfen Sie die Zeichen Doppelpunkt (:), Komma (,), Strichpunkt (;), Backslash (\), Slash (/) und Punkt (.) für die Bezeichner nicht verwenden. Ein Beispiel für den Wert eines Tags wäre »PO SM TM«, wobei die Bezeichner der Reihe nach für »Process Owner«, »Scrum Master« und »Team Member« stehen.

Wird eine `DragSourceInfo` auf ein UI-Element aggregiert, das mehrfach instanziiert wird (wie z. B. Zelleditoren pro Zeile in einer Tabelle), und haben Sie eine Eigenschaft der `DragSourceInfo` an ein Attribut der `dataSource` des Aggregats gebunden, wird der Wert des Attributs der Lead-Selection verwendet.

5.1.2 DropTargetInfo

Ziel

Im UI-Element `DropTargetInfo` (❶ in Abbildung 5.2) werden alle Informationen in Form von Eigenschaften gesammelt, die für die Definition des Ziels einer Drag-&-Drop-Operation relevant sind. Auf der linken Seite wird das UI-Element `Tree` ❷ im View Designer dargestellt, und auf der rechten Seite sehen Sie die dazugehörenden UI-Elemente in der UI-Elemente-Hierarchie. Zum `Tree` haben wir zwei `TreeNodeTypes` ❸ und einen `TreeItemType` ❹ für unser Beispiel definiert (siehe Abschnitt 4.5.8, »Tree«).

Eigenschaften

Für den `TreeNodeType` erkennen Sie auf der rechten Seite eine `DropTargetInfo` mit den dazugehörenden Eigenschaften. Die Eigenschaften `Id`, `enabled`, `tags` und `scope` werden analog wie für die `DragSourceInfo` (siehe Abschnitt 5.1.1) auch für die `DropTargetInfo` verwendet.

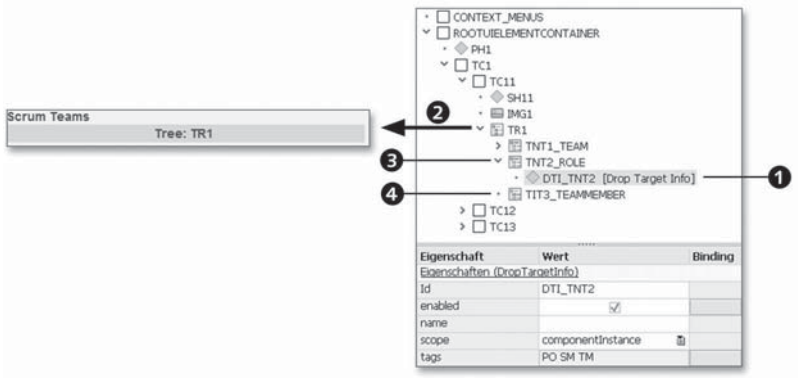

Abbildung 5.2 DropTargetInfo

name Mithilfe der Eigenschaft `name` legen Sie einen Namen fest, der bei der Behandlung der Drag-&-Drop-Operation im Ereignis `onDrop` ausgewertet werden kann (siehe Abschnitt 5.1.4, »Ereignis ›onDrop‹«). Im Speziellen wird diese Eigenschaft für die Behandlung der Drag-&-Drop-Operationen im Kontext von `Table` verwendet (siehe Abschnitt 5.5).

Asterisk in tags Eine besondere Bedeutung hat der Stern (*) in den Bezeichnern, die mithilfe der Eigenschaft `tags` definiert werden, da dieser als Joker verwendet werden kann. Ein Beispiel für den Wert eines Tags wäre »PO SM TEAM*«, wobei die Bezeichner der Reihe nach für »Process Owner«, »Scrum Master« und »TEAM*« mit beliebigem Text stehen. In diesem Beispiel wird vom System ein Mustervergleich mit den Source-Bezeichnern vorgenommen, und würde die Source einen Bezeichner »TEAM_AT« beinhalten, würde dieser dem Muster »TEAM*« im Ziel entsprechen. Mit dem Joker haben Sie demnach die Möglichkeit, Gruppen von Bezeichnern identisch zu behandeln.

5.1.3 DropTarget

Zielkapsel UI-Elemente, die keine `DropTargetInfo` als Unterelement anbieten, können mithilfe des UI-Elements `DropTarget` als Ziel eines Drag-&-Drop-Vorgangs definiert werden. Dabei funktioniert das `DropTarget` wie eine Kapsel um das UI-Element, das als Ziel dienen soll. Im View Designer können Sie nur ein Element unter dem `DropTarget` einfügen, jedoch stellt dies keine Einschränkung dar, da Sie einfach das UI-Element `TransparentContainer` einfügen können, das wiederum mehrere UI-Elemente aufnehmen kann.

DropTarget erzeugen In Abbildung 5.3 sehen Sie ein Beispiel für die Verwendung des UI-Elements `DropTarget`, das exemplarisch für den Einsatz in anderen Verwendungskontexten ist. Im oberen Bereich wird der Drag-Vorgang eines Teammitglieds ❶ von einer `ItemListBox` auf einen `TransparentContainer`, in dem die Details zum Teammitglied dargestellt werden, veranschaulicht. Der `TransparentContainer` beinhaltet ein Bild (Foto des Teammitglieds) und einen `FormattedTextView` (Rolle und Kompetenzen des Teammitglieds) und ist selbst durch ein `DropTarget` gekapselt. Im rechten unteren Bereich wird das Ergebnis nach dem Drop ❷ auf den `TransparentContainer` dargestellt, also nachdem die Behandlermethode für den Zeitpunkt `onDrop` aus der Drag Source abgearbeitet wurde.

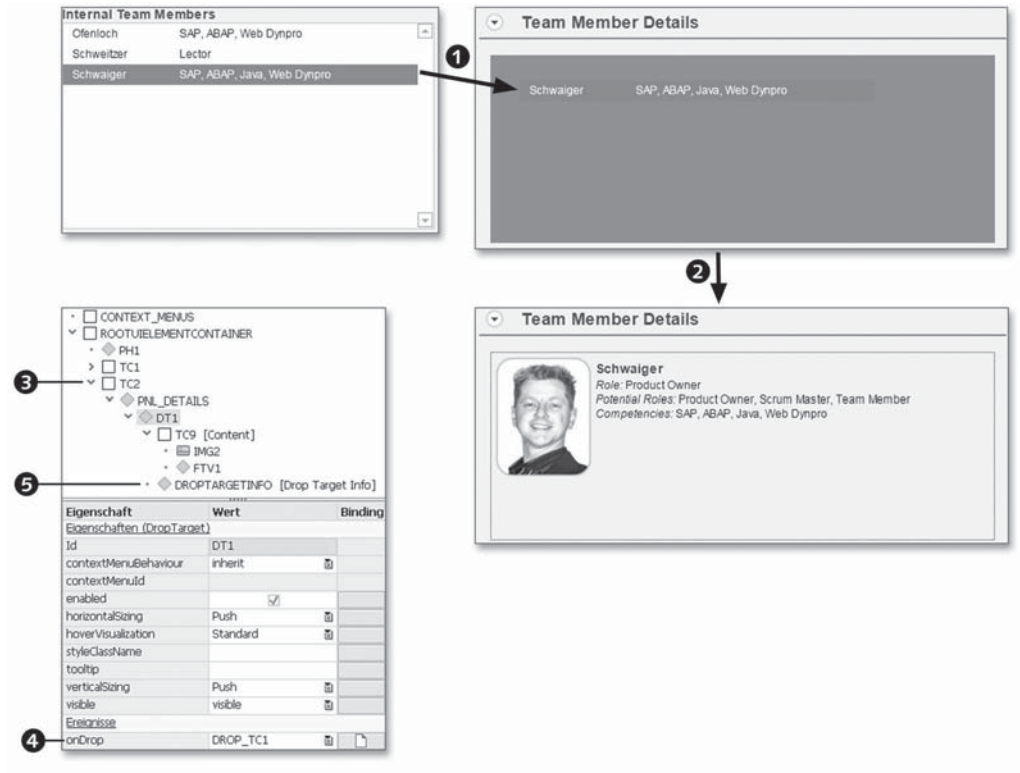

Abbildung 5.3 UI-Element DropTarget

Um ein `DropTarget` in Ihrem Layout zu verwenden, müssen Sie nur wenige Schritte ausführen.

1. Fügen Sie im ersten Schritt das UI-Element `DropTarget` in die UI-Elemente-Hierarchie ein ❸.
2. Definieren Sie eine Aktion für das Ereignis `onDrop` ❹, wie in Abschnitt 5.1.4 beschrieben.
3. Erzeugen Sie für das `DropTarget` eine `DropTargetInfo` ❺ (siehe Abschnitt 5.1.2).
4. Definieren Sie das Unterelement für das `DropTarget`.

Die Arbeit, die jetzt noch zu erledigen ist, betrifft die Implementierung der Behandlermethode für das Ereignis `onDrop`. Diese gestalten Sie je nach der gewünschten Reaktion auf den Drop. In unserem Beispiel wird in der Implementierung das Context-Element für das ausgewählte Teammitglied gelesen, formatiert und dem `Image` und `FormattedTextView` in Form eines Context-Knoten-Elements zur Verfügung gestellt.

[!] **Roundtrip vor der Drop-Behandlung**

Beim Auslesen der Daten aus dem Context-Knoten für die `DropDownListBox` mittels der Lead-Selection muss vor der Drop-Behandlung noch ein Roundtrip durchgeführt werden. Dazu können Sie z. B. eine Aktion für das Ereignis `onLeadSelection` anlegen.

5.1.4 Ereignis »onDrop«

In den UI-Elementen, die als Ziel verwendet werden können, wird das Ereignis `onDrop` angeboten. Damit wird der funktionale Aspekt der Behandlung eines Fallenlassens auf das Ziel implementiert. Dem Ereignis müssen Sie eine Aktion zuordnen, die eine zugeordnete Aktionsbehandler-Methode besitzt. Darin können Sie die gewünschte Reaktion des Drops implementieren, wie z. B. das Einfügen einer neuen Tabellenzeile.

Drop-Daten

Die Daten, die vom Ziel übertragen werden, stehen als Parameter der Aktionsbehandler-Methode für das Ereignis `onDrop` zur Verfügung, wie Sie im Beispiel in Abbildung 5.4 sehen können. Dort werden die Parameter dargestellt, die im Kontext eines `ItemListBox`-Drops übertragen, also an die Aktionsbehandler-Methode für `onDrop` als Parameter übergeben werden:

- Der Parameter `WDEVENT` ❶ repräsentiert wie immer bei einer Aktionsbehandler-Methode das Ereignis. In diesem Objekt sind die

ereignisrelevanten Daten abgelegt, die sich auch in der Schnittstelle der Aktionsbehandler-Methode widerspiegeln.

- Die weiteren Parameter werden in unserem Beispiel explizit über die Schnittstelle der Behandlermethode übergeben. Explizit bedeutet, dass Sie die Parameter entweder manuell in der Schnittstelle der Behandlermethode anlegen oder diese mithilfe des Buttons Parameter aus UI-Ereignis (□) anlegen lassen.

Die Menge und Art der Parameter richten sich nach dem Typ des Drop-Ziels, daher werden wir zu den folgenden Beschreibungen der UI-Elemente immer auch die Parameter in einem Abschnitt »Ereignis onDrop« erläutern.

Die Parameter, die wir in Abbildung 5.4 als Teil der Parameterliste zur Aktionsbehandler-Methode für das Ereignis `onDrop` gezeigt haben, finden sich auch im Objektattribut `PARAMETERS` des Referenzparameters `WDEVENT` wieder. Das bedeutet für Sie, dass Sie auf zwei Arten auf die Daten zugreifen können.

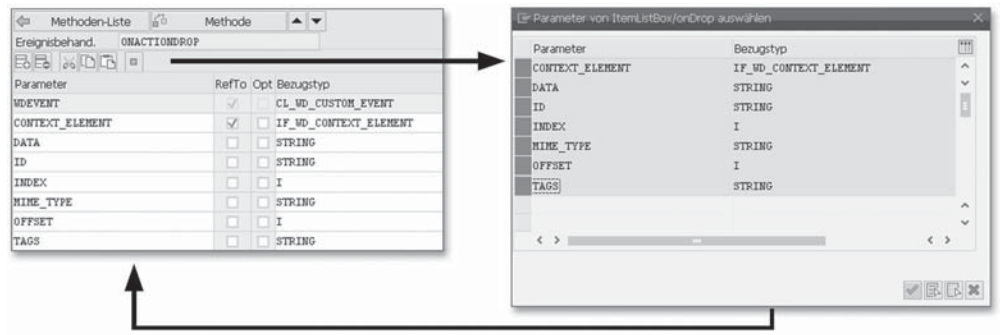

Abbildung 5.4 onDrop-Parameter der ItemListBox

Im Folgenden erläutern wir die für das Drag & Drop relevanten Parameter im Attribut `PARAMETERS` des Ereignisobjekts `WDEVENT`, das Sie in Abbildung 5.5 sehen können.

WDEVENT

Das Ereignisobjekt `WDEVENT` vom Typ `CL_WD_CUSTOM_EVENT` besitzt die Attribute `PARAMETERS` ❶, `NAME` ❷ und `SOURCE_COMPONENT_ID` ❸. Das Attribut `NAME` beinhaltet den Namen des Auslöseereignisses, in unserem Fall `ON_DROP`. Das Attribut `SOURCE_COMPONENT_ID` hält den Namen der Web-Dynpro-Component, von der aus der Drag gestartet wurde. Falls die Drag-&-Drop-Operation in derselben Web-Dynpro-Component ausgeführt wird, ist das Attribut initial.

Abbildung 5.5 WDEVENT-Objektattribute

Das Attribut `PARAMETERS` repräsentiert eine interne Tabelle mit den folgenden Einträgen:

- `ID`
  Der Eintrag `ID` steht für die ID des UI-Elements, das das `onDrop`-Ereignis ausgelöst hat, d. h. für das Ziel der Drag-&-Drop-Operation.
- `CONTEXT_ELEMENT`
  Der Eintrag `CONTEXT_ELEMENT` repräsentiert das Context-Element aus der Quelle, die zu der Drag-&-Drop-Operation gehört.
- `DATA`, `MIME_TYPE`, `TAGS`
  Diese drei Einträge liefern die Daten, die in der `DragSourceInfo` der Quelle definiert wurden (siehe Abschnitt 5.1.1).
- `INDEX`
  Der Eintrag `INDEX` bietet die Information über die Drop-Position im betroffenen Ziel. Zum Beispiel hat der `INDEX` den Wert 2 in einer `ItemListBox`, weil der Benutzer den Drop vor dem zweiten Eintrag ausgeführt hat.
- `OFFSET`
  Der Eintrag `OFFSET` liefert die Positionsinformation der Drop-Position relativ zum Eintrag `INDEX` oder anderen Bezugsgrößen. Zum Beispiel hat `OFFSET` den Wert -1 in einer `ItemListBox`, da der Benutzer den Drop vor dem zweiten Eintrag ausgeführt hat.

5.2 Tree

Das UI-Element `Tree` bietet vielfältige Möglichkeiten und Unterstützungen für die Drag-&-Drop-Operationen. Wie Sie diese im Detail umsetzen können, erfahren Sie in diesem Abschnitt.

DragSourceInfo

Knoten oder Blatt

Falls Sie einen Knoten oder ein Blatt eines Baums als Drag Source verwenden möchten, müssen Sie zu dem Knoten (`TreeNodeType`) oder Blatt (`TreeItemType`) ein Unterelement vom Typ `DragSourceInfo` im View Designer definieren.

In Abbildung 5.6 sehen Sie eine eingefügte `DragSourceInfo` für ein Blatt ❶ und einen Knoten ❷. Im unteren Bereich der Abbildung erkennen Sie die optische Darstellung des Drag-Vorgangs ❸.

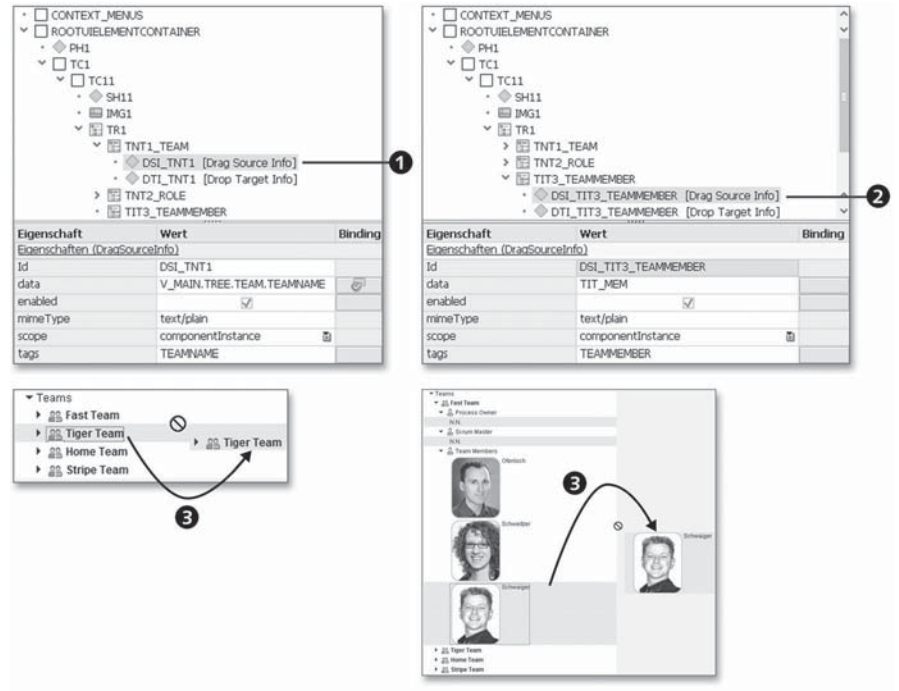

Abbildung 5.6 Tree DragSourceInfo

DropTargetInfo

Wird der `Tree` als Drop Target verwendet, stehen die folgenden Einfügepositionen zur Verfügung:

- zwischen existierenden Blättern
- als erstes oder letztes Blatt
- auf ein existierendes Blatt bzw. auf einen existierenden Knoten

Beispiel In Abbildung 5.7 sehen Sie ein Drop-Beispiel. Im unteren Bereich der Abbildung erkennen Sie auf der linken Seite die Visualisierung der Suche nach der Einfügeposition durch den Benutzer ❶. Dieser hat sich für das Einfügen an der zweiten Position entschieden.

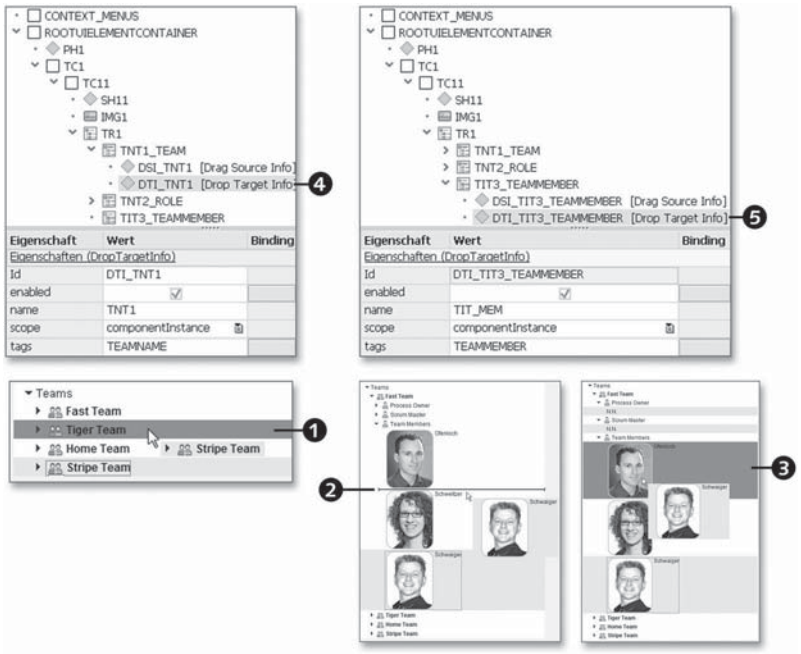

Abbildung 5.7 Tree DropTargetInfo

In der Abbildung rechts unten sehen Sie zwei Varianten für das Fallenlassen auf den `TreeItemType`: einerseits zwischen den Blättern ❷ mit Visualisierung der Einfügeposition und andererseits auf einem anderen Blatt ❸. Lässt der Benutzer das Element an der von ihm gewünschten Stelle fallen, wird das Ereignis `onDrop` des `TreeNodeType` ❹ bzw. `TreeItemType` ❺ ausgelöst und kann durch die Implementierung behandelt werden. Für den Drop zwischen den Blättern

müssen Sie noch eine `DropTargetInfo` für den Knoten `TNT2_ROLE` anlegen und ein Tag definieren, das mit dem Tag `TEAMMEMBER` vom `TreeItemType TIT3_TEAMMEMBER` zusammenpasst, z. B. `TEAM*`.

Ereignis »onDrop«

Parameter Das Ereignisobjekt `WDEVENT`, das an die Behandlermethode übergeben wird, liefert für die Implementierung die folgenden Informationen:

- `ID`: Das Element `ID` beschreibt die ID des Drop Targets.
- `CONTEXT_ELEMENT`: Das Element liefert eine Referenz auf ein Context-Element.
  - Falls auf ein Baumblatt oder einen Baumknoten gedroppt wird, ist dies das Context-Element, das die Datenbasis des Blattes/Knotens bildet.
  - Falls zwischen Blättern/Knoten gedroppt wird, ist dies das Context-Element, das für das Blatt/den Knoten nach der Drop-Position steht.
  - Falls am Ende von Blättern/Knoten gedroppt wird, ist dies das Context-Element, das für das letzte Blatt/den letzten Knoten steht.
- `DATA`: Das Element `DATA` beinhaltet die Daten aus der Drag Source.
- `MIME_TYPE`: Das Element `MIME_TYPE` enthält den MIME-Typ aus der Drag Source.
- `TAGS`: Das Element `TAGS` beinhaltet die Tags aus der Drag Source.
- `OFFSET`: Über `OFFSET` wird die Elementposition im Knoten des Baums bestimmt.
  - Der Parameter hat den Wert -1, falls in einer Liste von Blättern/Knoten gedroppt wird.
  - Er hat den Wert 1, falls am Ende einer Liste von Blättern/Knoten gedroppt wird.
  - Er hat den Wert 0, falls direkt auf ein Blatt/einen Knoten gedroppt wird.
- `PATH`: Das Element `PATH` enthält den Pfad zum `CONTEXT_ELEMENT` im Context.

Für die Implementierung der Reaktion auf das Fallenlassen von Elementen bieten sich unter anderem die Methoden `move_*` aus dem Interface `IF_WD_CONTEXT_NODE` dazu an, Elemente zu einem Context-Knoten zu verschieben.

5.3 GridLayout/MatrixLayout

Layout

Im `GridLayout` und `MatrixLayout` (siehe Abschnitt 3.3, »Layouts«) können einzelne Zellen dieses Layouts verschoben bzw. Objekte zwischen Zellen eingefügt werden. Beachten Sie dabei, dass Drag & Drop nur bei einspaltigem `GridLayout` bzw. einzeiligem `MatrixLayout` möglich ist.

DragSource

Damit eine Drag-Operation durchgeführt werden kann, muss für eine Zelle ein Griff (Handle) definiert werden. Dieser ermöglicht das »Angreifen« der Zelle. Dabei können ein `SectionHeader`, eine `Caption` oder ein `Image` mit der UI-Element-Eigenschaft `isDragHandle = true` eingesetzt werden.

Beispiel

In Abbildung 5.8 sehen Sie ein Beispielszenario, das es dem Benutzer ermöglicht, die Position einer `ItemListBox` zu verändern. Beim Verschieben der Box wird im User Interface die neue potenzielle Einfügeposition durch eine gestrichelte Linie visualisiert.

In Abbildung 5.9 haben wir das Layout des Szenarios schematisch dargestellt. Der zentrale Container für die Gestaltung ist der `TransparentContainer` TC1, der als Layout den Wert `GridLayout` und die Spaltenanzahl `Layout.colCount = 1` erhält. In TC1 haben wir die drei weiteren `TransparentContainer` TC11, TC12 und TC13 platziert, die dann gezogen werden. Beim Drag-Vorgang können auch Daten mitgegeben werden, diese werden in den Layoutdaten (`LayoutDaten.dragData`) abgelegt. In unserem Beispiel wird der `TransparentContainer` gezogen, daher werden die Drag-Daten in den dortigen Layoutdaten abgelegt. Als Ausprägungen haben wir in unserem Beispiel »TEAMS«, »ITM« (für »Internal Team Members«) und »ETM« (für »External Team Members«) verwendet.

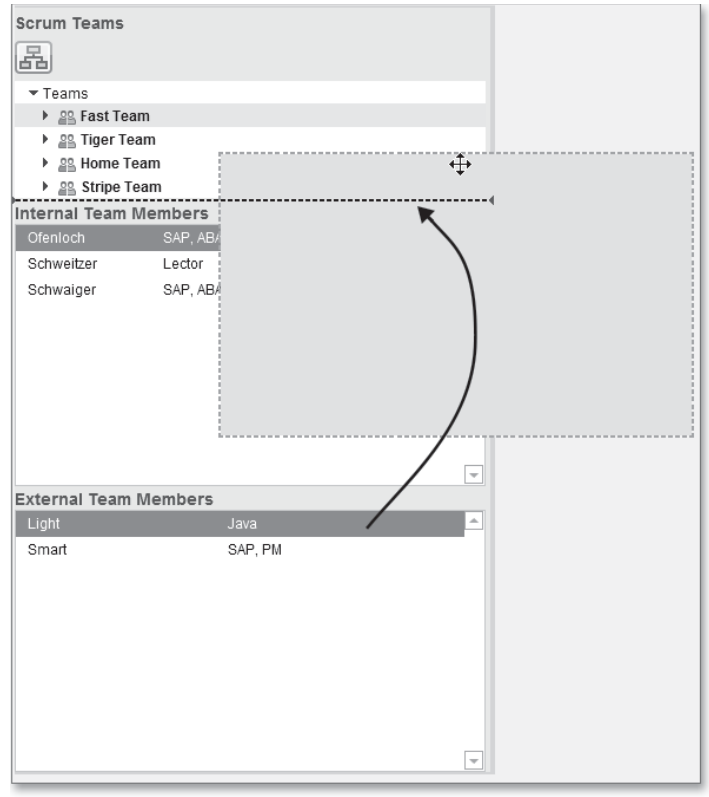

Abbildung 5.8 Drag & Drop einer ItemListBox

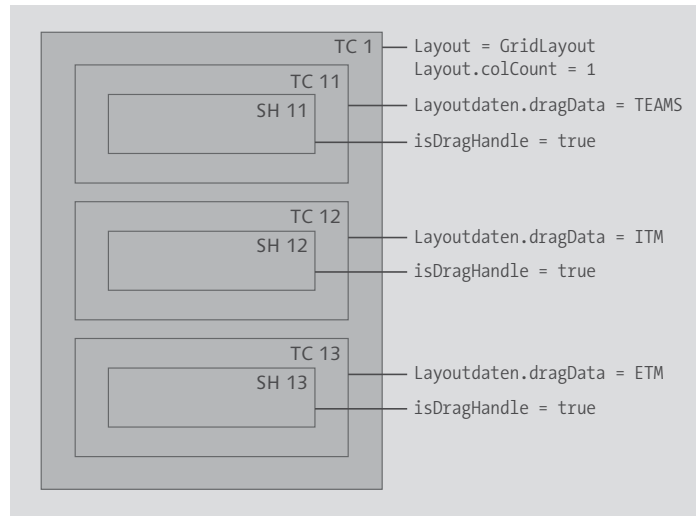

Abbildung 5.9 Drag GridLayout/MatrixLayout

Handle

Für den Griff (Handle) haben wir den `SectionHeader` verwendet (SH11, SH12 und SH13). Hier wurde im `SectionHeader` die UI-Element-Eigenschaft `isDragHandle = true` gesetzt.

DropTarget

Wird das `GridLayout` bzw. `MatrixLayout` als Drop Target verwendet, stehen die folgenden möglichen Einfügepositionen zur Verfügung:

- Im `GridLayout` und `MatrixLayout` können einzelne Zellen dieses Layouts verschoben werden.
- Im `GridLayout` und `MatrixLayout` können Objekte zwischen den Zellen eingefügt werden.

Ereignis »onDrop«

Das Ereignis `onDrop` werden Sie für `GridLayout` und `MatrixLayout` vergeblich im View Designer suchen. Der einfache Grund dafür ist, dass dieses dynamisch gesetzt werden muss. Jedoch ist es in der dynamischen Programmierung beim Setzen des Ereignisses erforderlich, den Namen einer Aktion an die Setter-Methode (`set_on_drop`) zu übergeben.

Aktion und Behandler

Legen Sie als Vorbereitung für die dynamische Programmierung mithilfe der Registerkarte Aktionen eine Aktion an. Dadurch wird automatisch eine Behandlermethode erzeugt, die für die Behandlung des `onDrop`-Ereignisses herangezogen wird. In Listing 5.1 zeigen wir Ihnen ein Beispiel für die Implementierung der Behandlermethode.

```
METHOD onactiondrop_tc.
* Hier wird nur das Event-Objekt in den Attributen abgelegt.
* Die eigentliche Behandlung erfolgt in WDDOMODIFYVIEW
  IF wdevent IS BOUND.
    wd_this->gd_index = wdevent->get_string( 'INDEX' ).
    wd_this->gd_offset = wdevent->get_string( 'OFFSET' ).
    wd_this->gd_data = wdevent->get_string( 'DATA' ).
    wd_this->gd_tags = wdevent->get_string( 'TAGS' ).
    wd_this->gd_id = wdevent->get_string( 'ID' ).
  ENDIF.
ENDMETHOD.
```

Listing 5.1 Behandlermethode für GridLayout/MatrixLayout

WDEVENT puffern

Im Gegensatz zu den anderen Drag-&-Drop-Realisierungen müssen Sie bei `GridLayout` und `MatrixLayout` die Daten aus dem Ereignisobjekt `WDEVENT` zwischenspeichern, da diese zu einem späteren Zeitpunkt in der Methode `wddomodifyview` benötigt werden. Wir haben im Beispiel in Listing 5.1 die Attribute des View-Controllers verwendet. Die Bedeutung der dabei verwendeten Parameter `ID`, `MIME_TYPE` und `TAGS` haben wir zum Ereignis `Tree.onDrop` in Abschnitt 5.2, »Tree«, besprochen. Die übrigen Parameter haben die folgende Bedeutung:

- `CONTEXT_ELEMENT`: Der Parameter `CONTEXT_ELEMENT` ist immer initial.
- `DATA`: Der Parameter `DATA` repräsentiert die Daten aus der Layoutdaten-Eigenschaft `dragData` des gezogenen Elements.
- `OFFSET`: Der Parameter `OFFSET` hält die Elementposition in den Zellen des `GridLayout` oder `MatrixLayout`. Der Parameter hat den Wert -1, falls der Wert des `INDEX` als Einfügeposition verwendet werden soll. Er hat den Wert 1, falls am Ende der Zellen gedroppt wird.
- `INDEX`: Der Parameter `INDEX` liefert den Wert der Referenzposition, die in Kombination mit dem `OFFSET` die tatsächliche Position ergibt.

Dynamische Programmierung

Beispiel

Im Vergleich zu den anderen Drag-&-Drop-UI-Elementen ist bei der Verwendung von `GridLayout` und `MatrixLayout` ein hohes Maß an dynamischer Programmierung (siehe Kapitel 6, »Dynamische Programmierung«) erforderlich. In Listing 5.2 sehen Sie ein Beispiel für eine solche Implementierung.

```
METHOD wddomodifyview.
  DATA:
* Der TransparentContainer für die D&D-Operation
      lr_tc TYPE REF TO cl_wd_transparent_container,
* Das GridLayout im TransparentContainer
      lr_grid_layout TYPE REF TO cl_wd_grid_layout,
* Die GridLayout-Daten für dragData
      lr_grid_layout_data TYPE REF TO cl_wd_grid_data,
* Eine neue DragSourceInfo für den TransparentContainer
       lr_drag_source_info TYPE REF TO cl_wd_drag_source_
       info,
```

```
* Eine neue DropTargetInfo für den TransparentContainer
       lr_drop_target_info TYPE REF TO cl_wd_drop_target_
       info.
* Beim ersten Mal
  IF first_time = abap_true.
************************************************
* TC1
************************************************
* TransparentContainer ermitteln für Drag & Drop
    lr_tc ?= view->get_element(
        id      = 'TC1' ).
* Layout zum TransparentContainer ermitteln
    lr_grid_layout ?= lr_tc->get_layout( ).
* DropTargetInfo instanziieren
    CALL METHOD cl_wd_drop_target_info=>new_drop_target_info
      EXPORTING
        enabled = abap_true
        id      = 'DTI_TC1'
        name    = 'TC1'
        tags    = 'TC'
        view    = view
      RECEIVING
        control = lr_drop_target_info.
* DropTarget zum Layout ergänzen
    CALL METHOD lr_grid_layout->set_drop_target_info
      EXPORTING
        the_drop_target_info = lr_drop_target_info.
* DragSourceInfo instanziieren
    CALL METHOD cl_wd_drag_source_info=>new_drag_source_info
      EXPORTING
        data    = 'TC1'
        enabled = abap_true
        id      = 'DSI_TC1'
        tags    = 'TC'
        view    = view
      RECEIVING
        control = lr_drag_source_info.
* DragSource zum Layout ergänzen
    CALL METHOD lr_grid_layout->set_drag_source_info
      EXPORTING
        the_drag_source_info = lr_drag_source_info.
* Drop-Aktion setzen
* Muss vorher in den Aktionen definiert werden!
    CALL METHOD lr_grid_layout->set_on_drop
      EXPORTING
        value = 'DROP_TC'.  ENDIF.
************************************************
```



```
* Behandlung des Drops auf die TransparentContainer
  DATA: lt_children TYPE
cl_wd_uielement_container=>tt_uielement,
        lr_child LIKE LINE OF lt_children,
        ld_drag_data TYPE string,
        ld_target_index TYPE i,
        ld_source_index TYPE i.
* Ist ein onDrop aufgetreten?
  IF wd_this->gd_index IS NOT INITIAL.
* TransparentContainer ermitteln
    lr_tc ?= view->get_element(
     id      = 'TC1' ).
* Kinder
    lt_children = lr_tc->get_children( ).
* Finde das UI-Element, von dem gedraggt wurde
    LOOP AT lt_children INTO lr_child.
      ld_source_index = sy-tabix.
* Hole die Layoutdaten vom Kind
      lr_grid_layout_data ?= lr_child->get_layout_data( ).
* Hole von der UI-Eigenschaft dragData den Wert
      ld_drag_data = lr_grid_layout_data->get_drag_data( ).
* Vergleiche den Wert mit den Daten von onDrop
      IF ld_drag_data = wd_this->gd_data.
* Gefunden!
        EXIT.
      ENDIF.
    ENDLOOP.
* Neue Position ermitteln
    IF wd_this->gd_offset = -1.
      ld_target_index = wd_this->gd_index.
* Am Ende
    ELSE.
      ld_target_index = wd_this->gd_index + 1.
    ENDIF.
* Durch das Löschen verschiebt sich der Index
    IF ld_source_index < ld_target_index.
      ld_target_index = ld_target_index - 1.
    ENDIF.
* Aus der alten Position löschen und in die neue einfügen
    IF ld_source_index <> ld_target_index.
* Löschen
      lr_tc->remove_child( index = ld_source_index ).
* Einfügen
      lr_tc->add_child(
        the_child = lr_child
        index = ld_target_index ).
* Fertig
```

```
      CLEAR wd_this->gd_index.
    ENDIF. "Ungleiche Position?
  ENDIF. "D&D?
ENDMETHOD.
```

Listing 5.2 Drag & Drop GridLayout/MatrixLayout

Container-Layout ermitteln

Die Implementierung startet mit der Ermittlung der Referenz auf den `TransparentContainer` (es können auch andere Container sein, wie z. B. eine Gruppe), für den Drag & Drop realisiert wird, d. h., in den Elemente eingefügt werden sollen oder dessen Kindelemente Sie per Drag & Drop verschieben möchten. Über diese Referenz wird eine Referenz auf das `GridLayout` (`MatrixLayout`) bezogen.

DropTargetInfo, DragSourceInfo, onDrop

Nach der Erzeugung eines `DropTargetInfo`- und `DragSourceInfo`-Objekts werden beide Objekte dem Layout zugewiesen. Darüber hinaus wird das `onDrop`-Ereignis mit einer Aktion gekoppelt, die Sie zuvor definiert haben.

Die genannten Schritte werden nur einmal in der Lebenszeit des View-Controllers ausgeführt. Die nächsten Schritte werden immer dann ausgeführt, wenn der Benutzer eine Drag-&-Drop-Operation im Layout durchführt.

Betroffenes Element

Anschließend ermitteln Sie zu Ihrem Container die Kindelemente, um für jedes Element prüfen zu können, ob es das durch die Drag-&-Drop-Operation betroffene ist. Sie können dazu mithilfe der Layoutdaten der Kindelemente die `dragData` ermitteln. Darüber hinaus sind die `dragData` des fallengelassenen Elements aus der Aktionsbehandler-Methode bekannt. Durch einen einfachen Vergleich auf Gleichheit können Sie feststellen, welches der Kindelemente es war, das gezogen wurde.

Positionieren

Mithilfe von `INDEX` und `OFFSET` berechnen Sie die neue Position für das gezogene Kindelement. Danach löschen Sie dieses Element aus der alten Position und fügen es in die neue Position ein.

5.4 ItemListBox

In einer `ItemListBox` werden Einträge in einer Box aufgelistet. Diese Einträge kann der Benutzer einzeln oder mehrfach markieren und ziehen. Um einen einzelnen Eintrag zu draggen, reicht es, diesen anzuklicken und zu ziehen.

Mehrfachselektion

Möchte ein Benutzer mehrere Einträge ziehen, muss er mithilfe der Tastenkombination aus ⇧ und der linken Maustaste oder mithilfe der Pfeiltasten bzw. Strg und der linken Maustaste oder über die Pfeiltasten den gewünschten Bereich markieren.

DragSourceInfo

In Abbildung 5.10 sehen Sie die `DragSourceInfo`, die es ermöglicht, dass der Benutzer einen oder mehrere Einträge aus einer `ItemListBox` ziehen kann.

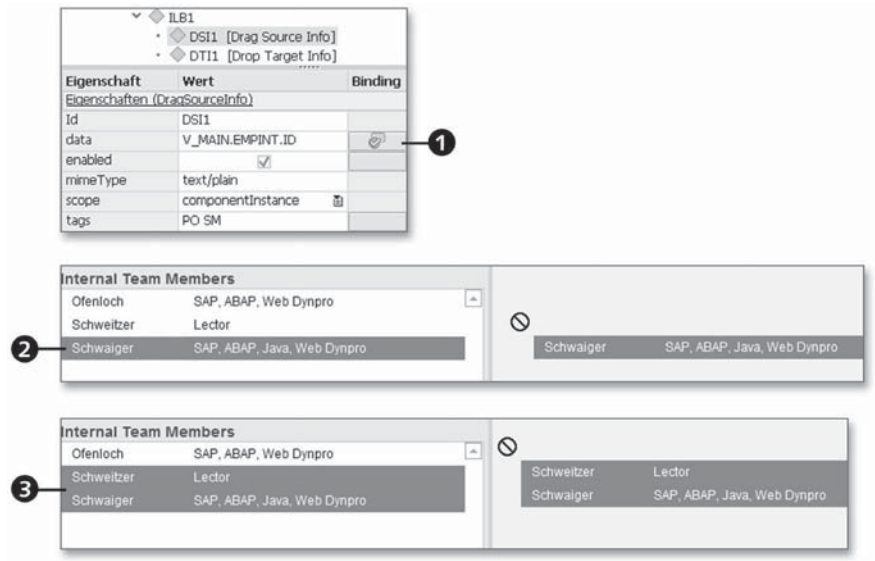

Abbildung 5.10 ItemListBox DragSourceInfo

Beschreibung

Die Eigenschaft `data` ❶ haben wir an den Context gebunden, um von dort die Referenz – hier als ID des Teammitglieds realisiert – auf ein Element der `DragSourceInfo` zuzuweisen. Diese Referenz wird an das Drop Target übergeben. Unter ❷ sehen Sie ein Beispiel für eine Einzelselektion und unter ❸ für eine Mehrfachselektion. Ausschlaggebend für die Fähigkeit der Mehrfachselektion ist die UI-Element-Eigenschaft `multipleSelection = true` im UI-Element `ItemListBox`.

DropTargetInfo

Einfügepositionen

Wird die `ItemListBox` als Drop Target verwendet, stehen die folgenden Einfügepositionen zur Verfügung:

- zwischen existierenden Einträgen
- an erster und letzter Position innerhalb der Liste

In Abbildung 5.11 finden Sie ein Drop-Beispiel. Im rechten Bereich der Abbildung sehen Sie die Visualisierung der Suche nach der Einfügeposition durch den Benutzer. Dieser hat sich für das Einfügen an der vorletzten Position entschieden. Droppt der Benutzer an der von ihm gewünschten Stelle, wird das Ereignis `onDrop` der `ItemListBox` ausgelöst und kann durch die Implementierung behandelt werden.

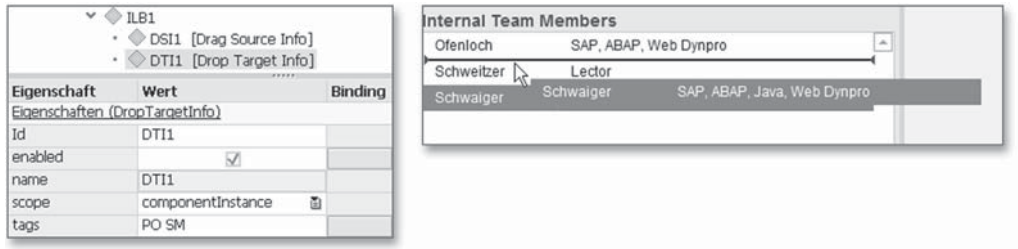

Abbildung 5.11 ItemListBox DropTargetInfo

Ereignis »onDrop«

Parameter

Das Ereignisobjekt `WDEVENT`, das an die Behandlermethode des Ereignisses übergeben wird, liefert für die Implementierung die folgenden Informationen:

- `CONTEXT_ELEMENT`: Der Parameter `CONTEXT_ELEMENT` liefert das Context-Element zur Drag Source.
- `INDEX`: Der Parameter `INDEX` liefert die Basisinformation zur Einfügeposition.
- `OFFSET`: Der Parameter `OFFSET` liefert die Position relativ zum Index.

Darüber hinaus enthält `WDEVENT` die Parameter `ID`, `DATA`, `MIME_TYPE` und `TAGS`, die analog zu den in Abschnitt 5.2, »Tree«, besprochenen Ereignisparametern verwendet werden.

Werte nach Einfügeposition

Falls der Benutzer einen Eintrag an das Ende anhängen möchte, hat der Index den Wert, dem die Anzahl der Einträge entspricht. Außerdem erhält der Offset den Wert 1. Falls der Benutzer einen Eintrag in die Liste einfügen möchte, hat der Index den Wert, der dem Index

des Eintrags nach der Einfügeposition entspricht. Zudem erhält der Offset den Wert –1. Falls der Benutzer einen Eintrag am Anfang in die Liste einfügen möchte, hat der Index den Wert 1 und der Offset den Wert –1.

Muster für Aktionsbehandler

Um die Reaktion auf den Drop des Benutzers in der Aktionsbehandler-Methode zu implementieren, können Sie das folgende Muster verwenden:

1. Ermitteln Sie die Quelle der Drag-&-Drop-Operation. Dafür können Sie das Attribut `DATA` aus dem Event-Objekt `WDEVENT` oder aus den Parametern der Aktionsbehandler-Methode verwenden.
2. Ermitteln Sie die selektierten Einträge auf der `ItemListBox`. Dazu beschaffen Sie sich die Referenz (`IF_WD_CONTEXT_NODE`) auf den Kontextknoten, an den die `ItemListBox` gebunden ist. Mithilfe der Methode `get_selected_elements` der Knotenreferenz ermitteln Sie die selektierten Einträge des Benutzers.
3. Aufgrund der Attribute `INDEX` und `OFFSET` aus dem Event-Objekt `WDEVENT` oder aus den Parametern der Aktionsbehandler-Methode bestimmen Sie die Drop-Position.
4. Realisieren Sie Ihre gewünschte Reaktion auf den Drop.

5.5 Table

Das UI-Element `Table` bietet vielfältige Möglichkeiten und Unterstützungen für die Drag-&-Drop-Operationen.

5.5.1 Nicht hierarchische Tabellen

Modi

Sie können selektierte Zeilen einer `Table` ziehen und Tabellenzeilen, Bilder und andere Quellen (Drag Source) zwischen oder auf bereits existierende Tabellenzeilen fallenlassen. Die zur Verfügung gestellten Modi sind die folgenden:

- **Einfügen**
  Für die Tabelle werden `DragSourceInfo`, `DropTargetInfo` und `onDrop` festgelegt. In diesem Fall kann ein oder können eines oder mehrere Elemente zwischen bestehenden Zeilen eingefügt werden (z. B. von einer `ItemListBox`).

- **Verschieben**
  Für die Tabelle werden `DragSourceInfo`, `DropTargetInfo` und `onDrop` definiert. In diesem Fall können eine oder mehrere Zeilen zwischen bestehenden Zeilen in der `Table` verschoben werden. Sollen mehrere Elemente verschoben werden, muss die Selektionskardinalität im Context angepasst werden.
- **Zeilen-Drop**
  Wählen Sie aus dem Kontextmenü des UI-Elements `Table` den Eintrag `INSERT DROP_ROW_TRG_INF`. Damit wird eine `DropTargetInfo` für den Drop auf eine Zeile eingefügt. Pflegen Sie die Werte zu den Eigenschaften der `DropTargetInfo`. Übernehmen Sie für die Eigenschaft `dropOnRowName` des UI-Elements `Table` den Wert der Eigenschaft `name` der `DropTargetInfo`. Für die Tabelle können Sie mehrere `DropTargetInfo` für den Zeilen-Drop definieren. In diesem Fall verwenden Sie das Data Binding der Eigenschaft `dropOnRowName`, um den unterschiedlichen Zeilen den Namen der `DropTargetInfo` zuzuordnen.

Selection

Für das Verschieben mehrerer Zeilen in der Tabelle müssen Sie im Context die Eigenschaft `Selection` für die `dataSource` der Tabelle mit der Obergrenze `n` setzen, d. h. `0..n` oder `1..n`. Falls Sie Zeilengruppierungen in der Tabelle verwenden (siehe Abschnitt 9.3, »Table«), müssen Sie keine Besonderheiten beachten.

DragSource

Beispiel

Falls Sie eine oder mehrere Zeilen einer Tabelle als Drag Source verwenden möchten, müssen Sie zum `Tree` ein Unterelement vom Typ `DragSourceInfo` im View Designer definieren. In Abschnitt 5.1.1 finden Sie die Definition der `DragSourceInfo`. Abbildung 5.12 zeigt eine eingefügte `DragSourceInfo` für eine Tabelle. Im unteren Bereich sehen Sie die optische Darstellung des Drag-Vorgangs ❶. Wir unterscheiden zwischen einem Drag einzelner ❷ und mehrerer Zeilen ❸. Notwendige Voraussetzung für den Mehrfach-Drag ist, dass die Eigenschaft `Selection` für die `dataSource` der Tabelle mit der Obergrenze `n` gesetzt ist.

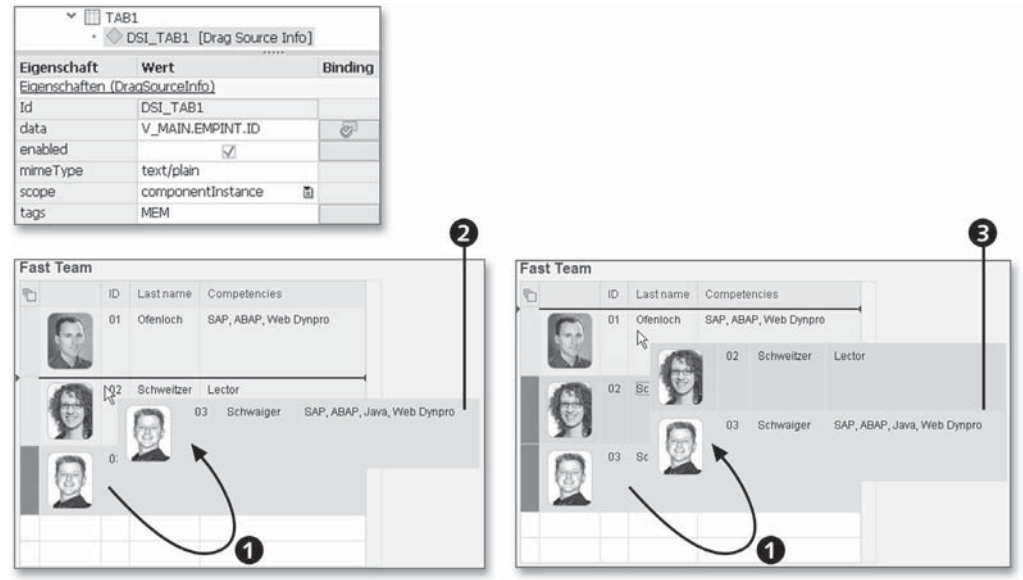

Abbildung 5.12 Table DragSourceInfo

DropTarget

Einfügepositionen

Wird die `Table` als Drop Target verwendet, stehen die folgenden Einfügepositionen zur Verfügung:

- zwischen existierenden Zeilen
- an erster und letzter Position innerhalb der aufgelisteten Zeilen
- auf eine Zeile

Zwischen den Zeilen

In Abbildung 5.13 sehen Sie ein Beispiel für das Fallenlassen zwischen Zeilen. Im oberen Bereich erkennen Sie die Eigenschaften der `DropTargetInfo`, die mit den passenden Werten versehen werden, wie in Abschnitt 5.1.2 beschrieben. In der Abbildung sehen Sie zwei Varianten für das Fallenlassen *zwischen Tabellenzeilen*. Unabhängig davon, ob Sie eine ❶ oder mehrere Zeilen ❷ fallenlassen möchten, ist es ausreichend, eine `DropTargetInfo` für den Vorgang zu definieren. Führt der Benutzer den Drop an der von ihm gewünschten Stelle ❸ aus, wird das Ereignis `onDrop` der `Table` ausgelöst und kann durch die Implementierung behandelt werden.

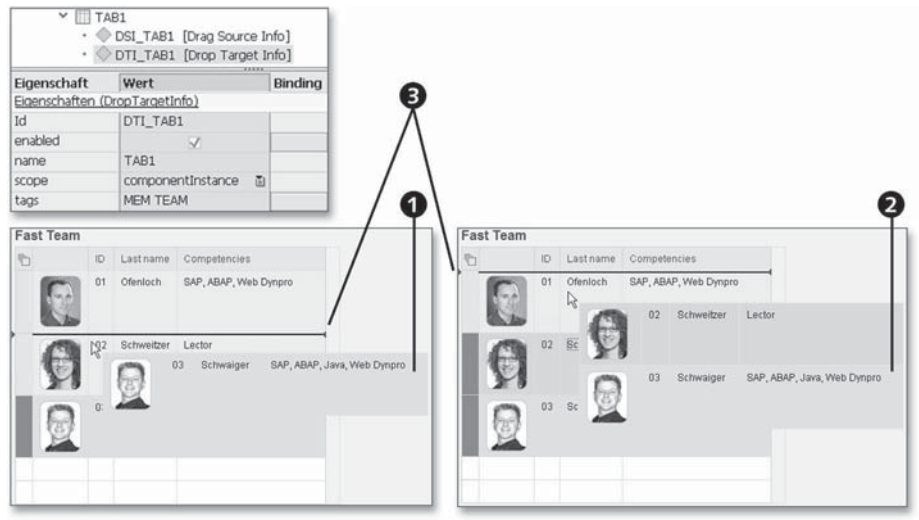

Abbildung 5.13 Table DropTargetInfo

Auf eine Zeile

Möchten Sie das Fallenlassen auf *eine Zeile* erlauben, müssen Sie andere Definitionen für das UI-Element `Table` festlegen (siehe auch Abbildung 5.14):

Abbildung 5.14 Table dropOnRowTargetInfo

1. Wählen Sie aus dem Kontextmenü des UI-Elements `Table` den Eintrag `INSERT DROP_ROW_TRG_INF` (`Drop On Row Target Info` ❶). Damit wird eine `DropTargetInfo` für den Drop auf eine Zeile eingefügt.
2. Pflegen Sie die Werte zu den Eigenschaften der `DropTargetInfo` (siehe Abschnitt 5.1.2). Der Wert der Eigenschaft `name` ❷ wird im nächsten Schritt verwendet.
3. Übernehmen Sie für die Eigenschaft `dropOnRowName` des UI-Elements `Table` den Wert der Eigenschaft `name` der `DropTargetInfo` ❸. Damit haben Sie die Definition abgeschlossen.

Alternativ zur Definition der `DROP_ROW_TRG_INF` (`Drop On Row Target Info`) können Sie auch eine bereits bestehende `DropTargetInfo` verwenden. Gehen Sie analog zum letzten Schritt der vorangegangenen Aufzählung vor.

Für die Tabelle können Sie mehrere `DropTargetInfo` für den Zeilen-Drop definieren. In diesem Fall verwenden Sie das Data Binding der Eigenschaft `dropOnRowName` des UI-Elements `Table`, um den unterschiedlichen Zeilen den Namen der `DropTargetInfo` zuzuordnen.

Ereignis »onDrop«

Parameter

Das Ereignisobjekt `WDEVENT`, das an die Behandlermethode des Ereignisses übergeben wird, liefert neben den bereits in Abschnitt 5.2, »Tree«, besprochenen Parametern `ID`, `DATA`, `MIME_TYPE` und `TAGS` die folgenden Attribute für die Implementierung:

- `CONTEXT_ELEMENT`: Der Parameter `CONTEXT_ELEMENT` ist immer initial.
- `OFFSET`: Der Parameter `OFFSET` hält einen Wert, der relativ zum `ROW_ELEMENT` zu interpretieren ist. Der Offset kann die Werte -1 (davor), 1 (danach, für das Ende der Elemente) und 0 (auf ein Element) annehmen.
- `ROW_ELEMENT`: Der Parameter `ROW_ELEMENT` hält das Context-Element aus der `dataSource` der `Table`, in die etwas eingefügt werden soll.

Implementierungsidee

Die Implementierung der Reaktion auf den Drop beinhaltet im Allgemeinen drei Schritte.

1. **Ermittlung der Datenquelle(n) der Drag-&-Drop-Operation**
   Um die Datenquelle(n) herauszufinden, bestimmen Sie z. B. die Referenz (`IF_WD_CONTEXT_NODE`) auf den Kontextknoten, an den die Quelle gebunden ist. Mithilfe der Methode `get_selected_elements` der Knotenreferenz ermitteln Sie die selektierten Elemente der Quelle.
2. **Ermittlung von Positionen**
   Ermitteln Sie den Index der Einfügeposition(en). Dabei unterstützen Sie die Methode `get_index` aus dem Interface `IF_WD_CONTEXT_ELEMENT` und das `OFFSET`-Attribut aus dem Ereignisobjekt.

3. **Manipulation der Context-Elemente der dataSource der Tabelle**
   Das Interface `IF_WD_CONTEXT_NODE` bietet die Methoden `move_*` an, um Elemente auf einen Context-Knoten zu verschieben. Damit können Sie die zu verschiebenden Elemente an die Einfügeposition im Context und somit in der `Table` verschieben.

5.5.2 Hierarchische Tabellen

Modi

In hierarchischen Tabellen, d. h. in Tabellen, die eine Master-Spalte `TreeByKeyTableColumn` oder `TreeByNestingTableColumn` besitzen (siehe Abschnitt 9.3, »Table«), gibt es zwei unterschiedliche Drag-&-Drop-Modi:

- **Einfacher Modus**
  Dieser Modus ist dafür gedacht, einen Drop zwischen Tabellenzeilen ausführen zu können. Falls Sie diesen Modus verwenden möchten, reicht es, für die Tabelle eine `DragSourceInfo` anzulegen (siehe Abschnitt 5.1.1). Beachten Sie, dass Sie die Eigenschaft `name` initial belassen müssen.
- **Komplexer Modus**
  Falls Sie der Eigenschaft `name` der `DropTargetInfo` einen Wert geben, ist nur der Drop zwischen den Zeilen der höchsten Hierarchiestufe (direkt unter der Wurzel) möglich. Für das Fallenlassen zwischen den Zeilen und auf eine Zeile niedrigerer Hierarchiestufen müssen Sie eine oder mehrere `dropOnRowTargetInfo` anlegen und diese über die Eigenschaft `dropOnRowName` des UI-Elements `Table` der Tabelle bzw. den Zeilen der Tabelle zuordnen (siehe Abschnitt 5.5.1, »Nicht hierarchische Tabellen«).

Beispiel

In Abbildung 5.15 sehen Sie Beispiele für den Drag unter Einbeziehung einer hierarchischen Tabelle:

- Im oberen Bild wird aus einer `ItemListBox` ein Eintrag auf den Baum gezogen ❶. Sie sehen die Markierung für die Einfügeposition zwischen den Zeilen.
- Im unteren Bereich des Bildes dient der Baum selbst als Drag-Quelle ❷. Sie sehen aufgrund der markierten Zeile und der Markierung für die Einfügeposition, dass zwischen die Zeilen gezogen wird. Bei der Implementierung der Behandlung des `onDrop`-Ereignisses können Sie sich an Abschnitt 5.1.4 orientieren.

Abbildung 5.15 Hierarchische Tabelle DragSource

5.6 CTable

Drop-Optionen

In der `CTable`, die wir ausführlich in Abschnitt 9.8 besprechen, gibt es verschiedenste Möglichkeiten, Drag & Drop zu verwenden. Verschiedene Elemente können in und auf Zeilen und Spalten sowie Zeilen- und Spaltenzwischenräumen bewegt werden. In Abbildung 5.16 haben wir die Drop-Optionen hervorgehoben.

❶ Ein Drop kann auf Zeilen ausgeführt werden (Eigenschaft `CTable.rowDropInfo`, Kardinalitätsobergrenze `n`).

❷ Ein Drop kann zwischen Zeilen ausgeführt werden (Eigenschaft `CTable.tableBodyDropInfo`, Kardinalitätsobergrenze `1`).

❸ Ein Drop kann zwischen Zeilen ausgeführt werden und übersteuert zeilenweise die Eigenschaft `CTable.tableBodyDropInfo` (Eigenschaft `CTable.rowEdgeDropInfo`, Kardinalitätsobergrenze `n`).

❹ Ein Drop kann auf Spalten ausgeführt werden (Eigenschaft `CTableColumn.columnDropInfo`, Kardinalitätsobergrenze `1`).

❺ Ein Drop kann zwischen Spalten ausgeführt werden (Eigenschaft `CTableColumn.columnHeaderDropInfo`, Kardinalitätsobergrenze `1`).

❻ Ein Drop kann zwischen Spalten ausgeführt werden und übersteuert spaltenweise die Eigenschaft `CTableColumn.columnHeaderDropInfo` (Eigenschaft `CTableColumn.columnEdgeDropInfo`, Kardinalitätsobergrenze `1`).

❼ Ein Drop kann auf Zellen ausgeführt werden (UI-Element `DropTargetCellEditor`).

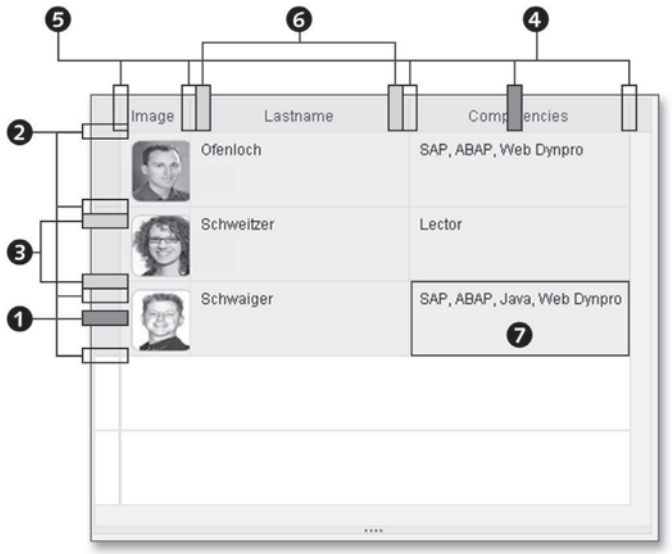

Abbildung 5.16 Drop-Positionen in der CTable

Kardinalitätsobergrenze

Wir haben hinter den Eigenschaftsnamen auch die Obergrenze der Kardinalität für die aggregierbaren `DropTargetInfos` angegeben. Es kann z. B. nur maximal eine `DropTargetInfo` für `tableBodyDropInfo` angegeben werden, jedoch eine beliebige Anzahl für `rowDropInfo`.

DropTargetInfo

DropTargetInfo anlegen

Bei der Anlage und der Verwendung der `DropTargetInfo` gehen Sie im Falle der `CTable` und `CTableColumn` folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf die `CTable`, und wählen Sie im Kontextmenü den Eintrag INSERT DROP_TARGET_INFO, um eine `DropTargetInfo` anzulegen.
2. Vergeben Sie im View-Designer oder per Programmierung eine eindeutige ID für die `DropTargetInfo`.

- *Im View-Designer*: Ordnen Sie der `CTable` bzw. `CTableColumn` eine Eigenschaft für den Drop, z. B. `rowDropInfo` oder `columnDropInfo`, sowie die zuvor vergebene ID der `DropTargetInfo` zu.
- *Per Programmierung*: Binden Sie die Eigenschaft der `CTable` bzw. `CTableColumn` (z. B. `rowEdgeDropInfo` oder `columnEdgeDropInfo`) an ein Attribut des Contexts, das die ID der `DropTargetInfo` beinhaltet.

DragSourceInfo

Für die `CTable` können Sie eine Reihe von Quellen definieren. Die Eigenschaften `rowDragInfo` bzw. `columnDragInfo` beinhalten die IDs der `DragSourceInfo` (siehe Abbildung 5.17).

rowDragInfo

Für Zeilen als Ausgangspunkt des Drags wird die Eigenschaft `rowDragInfo` in der `CTable` verwendet (❶ in Abbildung 5.17). Alle Zeilen haben dieselbe `DragSourceInfo` ❷.

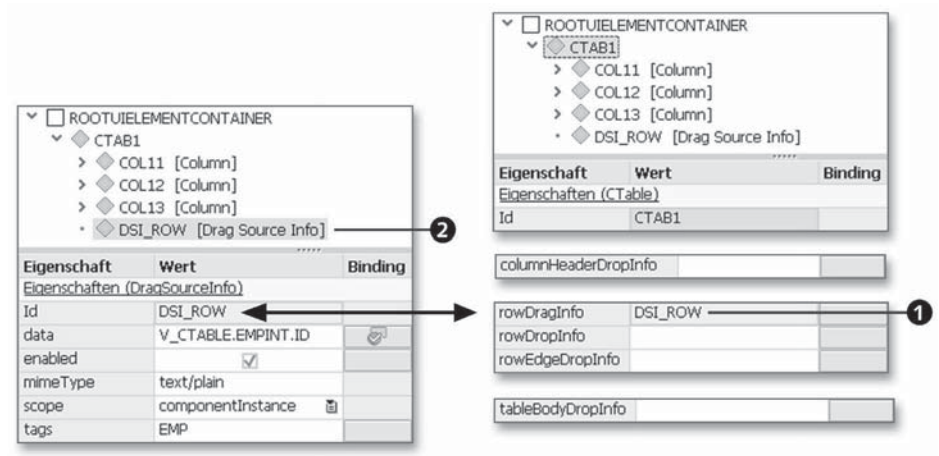

Abbildung 5.17 Verwendung der Eigenschaft rowDragInfo der CTable

columnDragInfo

Für Spalten als Ausgangspunkt des Drags wird die Eigenschaft `columnDragInfo` in der `CTableColumn` verwendet (❶ in Abbildung 5.18). Jede Spalte hat ihre eigene `DragSourceInfo` ❷.

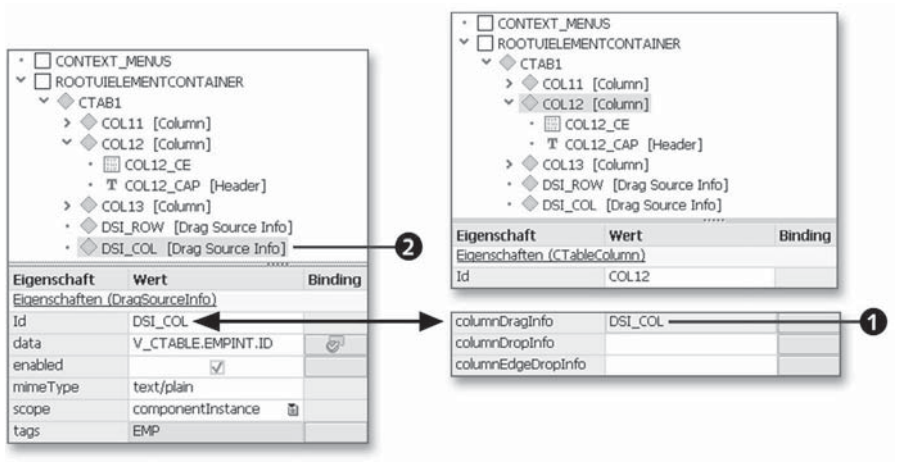

Abbildung 5.18 Verwendung der Eigenschaft columnDragInfo der CTableColumn

DropTarget

In Abbildung 5.16 haben Sie bereits die unterschiedlichen Möglichkeiten gesehen, um den Drop im Zusammenhang mit einer `CTable` zu implementieren. Diese basieren immer darauf, dass Sie eine `DropTargetInfo` zur `CTable` anlegen und diese dann über ihre `DropTargetInfo`-ID der passenden `CTable`- bzw. `CTableColumn`-Eigenschaft zuordnen. Die möglichen Eigenschaften für `CTable` bzw. `CTableColumn` sind in Tabelle 5.2 aufgeführt.

Drop-Zone	Eigenschaft von CTable	Eigenschaft von CTableColumn	Erläuterung
zwischen den Spalten	`columnHeaderDropInfo`		Beschreibt, welche Elemente zwischen allen Spalten gedroppt werden können.
auf einer Spalte		`columnDropInfo`	Beschreibt, welche Elemente auf eine Spalte gedroppt werden können.

Tabelle 5.2 DropTarget-Eigenschaften der CTable bzw. CTableColumn

Drop-Zone	Eigenschaft von CTable	Eigenschaft von CTableColumn	Erläuterung
vor oder nach einer Spalte, aber mit Bezug zur Spalte		`columnEdgeDropInfo`	Beschreibt, welche Elemente vor oder nach einer Spalte, aber innerhalb dieser Spalte gedroppt werden können.
zwischen Zeilen	`tableBodyDropInfo`		Beschreibt, welche Elemente zwischen den Zeilen gedroppt werden können.
auf Zeilen	`rowDropInfo`		Beschreibt, welche Elemente auf eine Zeile gedroppt werden können.
vor oder nach einer Zeile, aber mit Bezug zu einer Tabellenzeile	`rowEdgeDropInfo`		Beschreibt, welche Elemente vor oder nach einer Zeile, aber innerhalb dieser Zeile gedroppt werden können.
Tabellenzellen		`DropTargetCellEditor`	Um ein Element mit `DragSourceInfo` in eine Tabellenzelle zu droppen, verwenden Sie einen `DropTargetCellEditor` als Zelleneditor.

Tabelle 5.2 DropTarget-Eigenschaften der CTable bzw. CTableColumn (Forts.)

rowDropInfo

In Abbildung 5.19 sehen Sie ein Beispiel für die Verwendung der `DropTargetInfo` ❶ für die Eigenschaft `rowDropInfo` der `CTable` ❷. Durch diese Einstellungen wird ein Drop auf Tabellenzeilen, unter der Voraussetzung ermöglicht, dass für das Ereignis `drop` der `CTable` eine Aktion angelegt wurde.

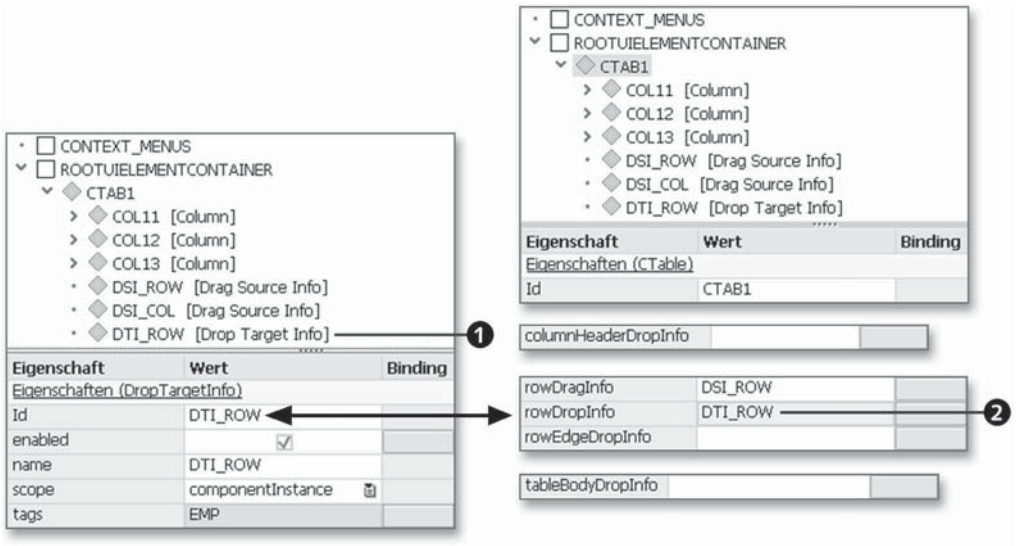

Abbildung 5.19 DropTarget für die Eigenschaft rowDropInfo der CTable

Im Ergebnis ist es möglich, eine Zeile der `CTable` auf eine andere Tabellenzeile zu ziehen. In Abbildung 5.20 sehen Sie den kombinierten Drag-&-Drop-Vorgang.

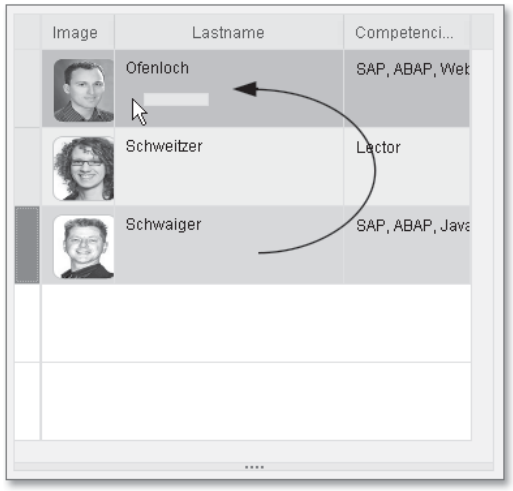

Abbildung 5.20 Drag & Drop auf CTable-Zeilen

tableBodyDropInfo

In Abbildung 5.21 sehen Sie ein Beispiel für die Verwendung der Eigenschaft `tableBodyDropInfo` in einer `CTable` ❶. Durch die Zuordnung der ID einer `DropTargetInfo` ❷ zur Eigenschaft `tableBodyDropInfo` kann der Benutzer den Drop an den Anfang oder das Ende der Tabelle oder zwischen den Zeilen ausführen.

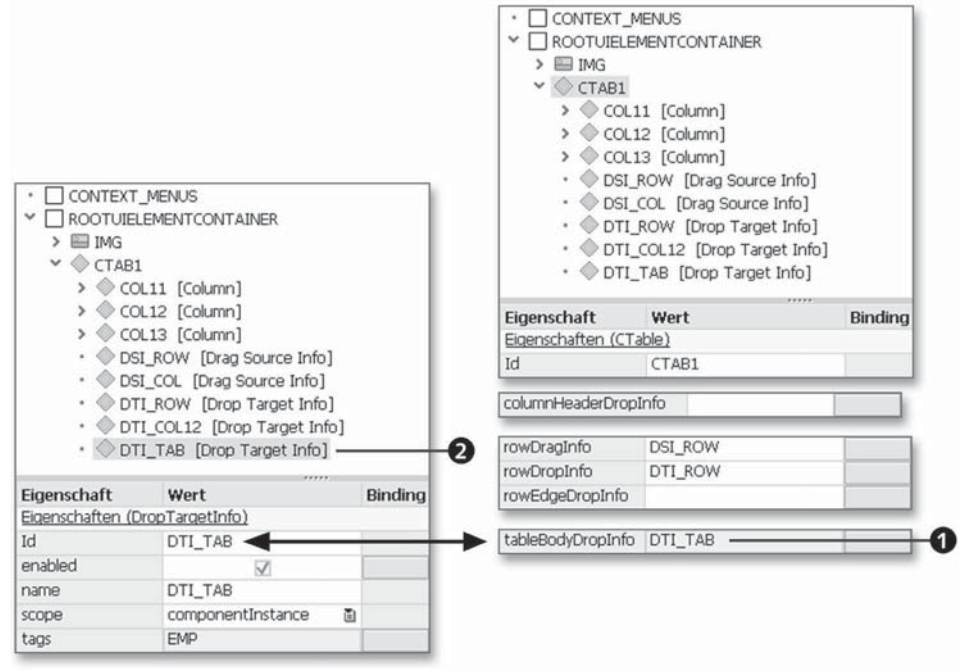

Abbildung 5.21 DropTarget für die Eigenschaft tableBodyDropInfo der CTable

rowEdgeDropInfo

Einzelne Zeilen können die Einstellung der Eigenschaft `tableBodyDropInfo` übersteuern, wenn Sie der `CTable`-Eigenschaft `rowEdgeDropInfo` eine `DropTargetInfo` zuordnen (siehe ❶ in Abbildung 5.22). Dies ermöglicht die individuelle Behandlung eines Drops auf eine bestimmte Tabellenzeile. Dazu müssen Sie ein Data-Binding der Eigenschaft `rowEdgeDropInfo` an ein Attribut vom Typ `string` vornehmen, in das Sie zur Laufzeit den Namen einer `DropTargetInfo` hinterlegen. Die `DropTargetInfo` kann entweder bereits während der Entwicklungszeit oder dynamisch während der Laufzeit erzeugt worden sein.

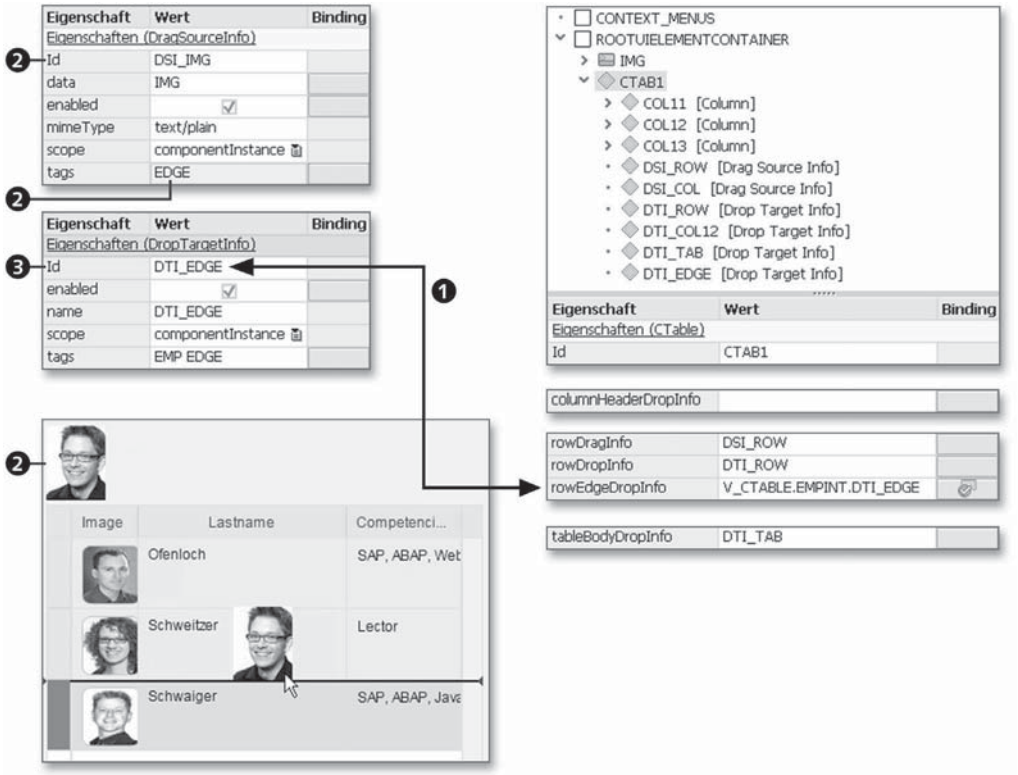

Abbildung 5.22 DropTarget für die Eigenschaft rowEdgeDropInfo der CTable

In unserem Beispiel in Abbildung 5.22 haben wir die `DragSourceInfo` mit der ID `DSI_EDGE` ❷ einem Image mit dem Tag `EDGE` zugeordnet. Anschließend haben wir im View-Designer die `DropTargetInfo` `DTI_EDGE` für eine Zeile angelegt ❸. Im Gegensatz zu den bisherigen Beispielen soll diese Information zeilenbezogen zugeordnet werden und dadurch die Eigenschaft `tableBodyDropInfo` für diese Zeile übersteuern. Haben wir dies für Zeile 2 in der `CTable` definiert, wird im Context im Attribut `DTI_EDGE` die ID der `DropTargetInfo` zu dieser Zeile hinterlegt und mittels Data-Binding an die `CTable`-Eigenschaft `rowEdgeDropInfo` gebunden ❶. Damit ist das individuelle Behandeln von Drop-Ereignissen auf eine bestimmte Zeile realisiert, wie in unserem Beispiel das Ziehen eines Bildes, das außerhalb einer Tabelle liegt, auf eine bestimmte Tabellenzeile, in der die Tags (in unserem Beispiel `EDGE`) übereinstimmen.

Vorrangregeln [«]

Die folgenden Regeln für die Überschreibung von Drag-&-Drop-Eigenschaften sollten Sie beachten:

- Wird für eine Zeile eine spezifische `rowEdgeDropInfo` definiert, wird die Eigenschaft `tableBodyDropInfo` der betreffenden Zeile überschrieben.
- Wird für eine Spalte eine spezifische `columnEdgeDropInfo` definiert, wird die Eigenschaft `columnHeaderDropInfo` der betreffenden Spalte überschrieben.

Einschränkungen [!]

Es gelten die folgenden einschränkenden Bedingungen für die Verwendung von IDs in den UI-Elementen `CTableColumn` und `CTable`:

- `CTableColumn`:
  - Der Wert für die Eigenschaft `columnDropInfo` darf nicht dem Wert für die Eigenschaft `columnEdgeDropInfo` entsprechen.
  - Der Wert für die Attribute `rowDropInfo` und `rowEdgeDropInfo` der Eigenschaft `columnDropInfo` muss ungleich sein.
  - Der Wert für die Attribute für `rowDropInfo` und `rowEdgeDropInfo` der Eigenschaft `columnEdgeDropInfo` muss ungleich sein.
- `CTable`:
  - Der Wert der Eigenschaft `columnHeaderDropInfo` darf nicht dem Wert der Eigenschaft `tableBodyDropInfo` entsprechen.

Ereignis »onDrop«

Um den Drop in einer `CTable` zu behandeln, stehen unterschiedliche Drop-Ereignisse für Zeilen, Spalten und Zellen zur Verfügung.

CTable.onDrop

Das Ereignis `onDrop` von `CTable` wird ausgelöst, wenn ein Element auf eine Tabellenzeile oder zwischen den Tabellenzeilen gedroppt wird. Das Ereignisobjekt `WDEVENT`, das an die Behandlermethode des Ereignisses übergeben wird, liefert neben den bereits in Abschnitt 5.2, »Tree«, besprochenen Parametern `ID`, `MIME_TYPE` und `TAGS` die folgenden Attribute für die Implementierung:

- `CONTEXT_ELEMENT`: Der Parameter `CONTEXT_ELEMENT` ist immer initial.

- `DATA`: Der Parameter `DATA` liefert die Daten aus der Drag Source. Falls `DATA` an ein Context-Attribut der `dataSource` der Tabelle gebunden ist, müssen Sie unter Umständen zuvor das Ereignis `onSelect` der `CTable` verwenden, um die Lead-Selection auf den selektierten Zeilenindex zu setzen.
- `OFFSET`: Der Parameter `OFFSET` hält einen Wert, der relativ zum `ROW_ELEMENT` zu interpretieren ist. Der Offset kann die Werte –1 (vor ein Element), 1 (hinter alle Elemente) und 0 (auf ein Element) annehmen.
- `DIRECTION`: Der Parameter `DIRECTION` erhält den Wert `VERTICAL`, falls Zeilen gedroppt werden, oder `HORIZONTAL`, falls Spalten gedroppt werden.
- `ROW_ELEMENT`: Der Parameter `ROW_ELEMENT` hält das Context-Element aus der `dataSource` der `CTable`, auf das der Drop durchgeführt wurde.

CTableColumn.onDrop

Das Ereignis `onDrop` von `CTableColumn` wird ausgelöst, wenn ein Element auf eine Tabellenspalte oder zwischen die Tabellenspalten gedroppt wird.

DropTargetCellEditor.onDrop

Das Ereignis `onDrop` des UI-Elements `DropTargetCellEditor` wird ausgelöst, wenn ein Element auf eine Zelle gedroppt wird.

5.7 Accordion

Items

In einem `Accordion` werden anklickbare `AccordionItem`- und `MultipleAccordionItem`-Elemente vertikal aufgelistet (siehe Abschnitt 3.2.1, »Accordion«). Diese Items kann der Benutzer im `Accordion` verschieben, d. h. ein Item vor oder nach einem anderen Item platzieren. Darüber hinaus ist es möglich, dass andere UI-Elemente in einer Drag-&-Drop-Operation auf ein Item gezogen werden.

DragSource

Beispiel

Falls Sie ein Item eines `Accordion` als DragSource verwenden möchten, müssen Sie ein Unterelement zum `Accordion` vom Typ `DragSourceInfo` im View Designer definieren (siehe Abschnitt 5.1.1). In Abbildung 5.23 sehen Sie eine eingefügte `DragSourceInfo` für ein `Accordion` ❶. Zudem erkennen Sie unter Punkt ❷ und ❸ die Eigenschaft `dragData` des `AccordionItem` und des UI-Elements `MultipleAccordionItem`, in der Sie Daten für den Drag-Vorgang hinterlegen können.

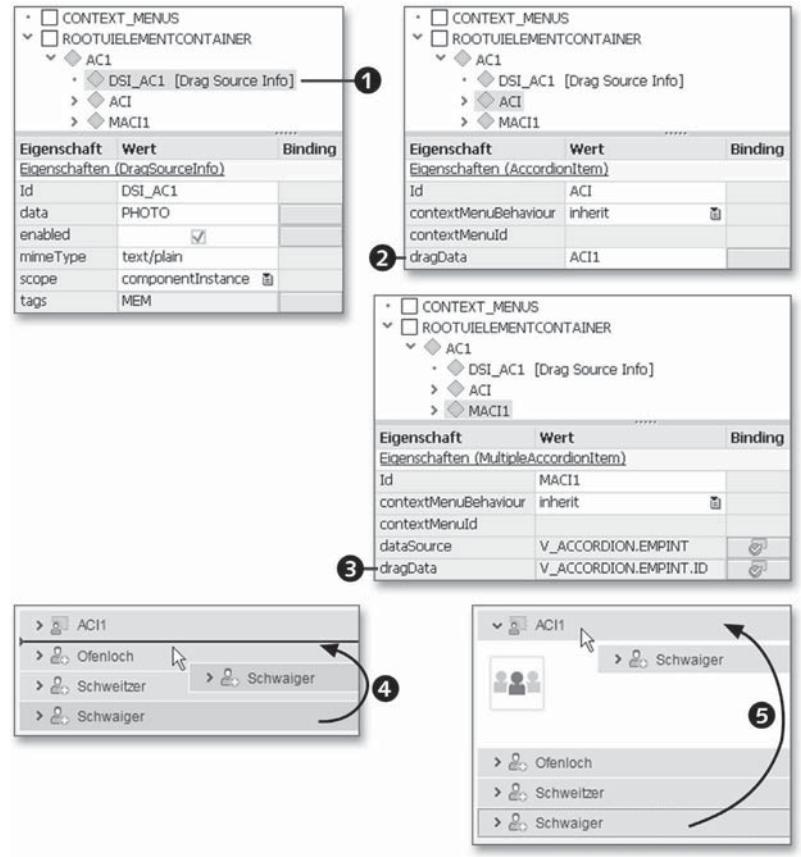

Abbildung 5.23 Accordion – DragSourceInfo

Beschreibung

Im unteren Bereich der Abbildung sehen Sie die optische Darstellung des Drag-Vorgangs. Wir unterscheiden zwischen einem Drag einzelner Items ❹ und einem Drag auf ein Item ❺. Voraussetzung für den Drag auf ein Item ist die Definition einer `DropTargetInfo`.

DropTargetinfo

Wird das `Accordion` als Drop Target verwendet, stehen die folgenden Einfügepositionen zur Verfügung:

- zwischen existierenden Items
- an erster und letzter Position der Items
- auf ein Item

Beispiel

In Abbildung 5.24 finden Sie zwei Drop-Beispiele. Für den Drop zwischen den Items müssen Sie zunächst die `DropTargetInfo` zum `Accordion` definieren ❶. Andererseits benötigen Sie die `DropTargetInfo`-Definition für jene Items, auf denen Sie einen Drop ausführen möchten, in unserem Beispiel das `AccordionItem` und das `MultipleAccordionItem`.

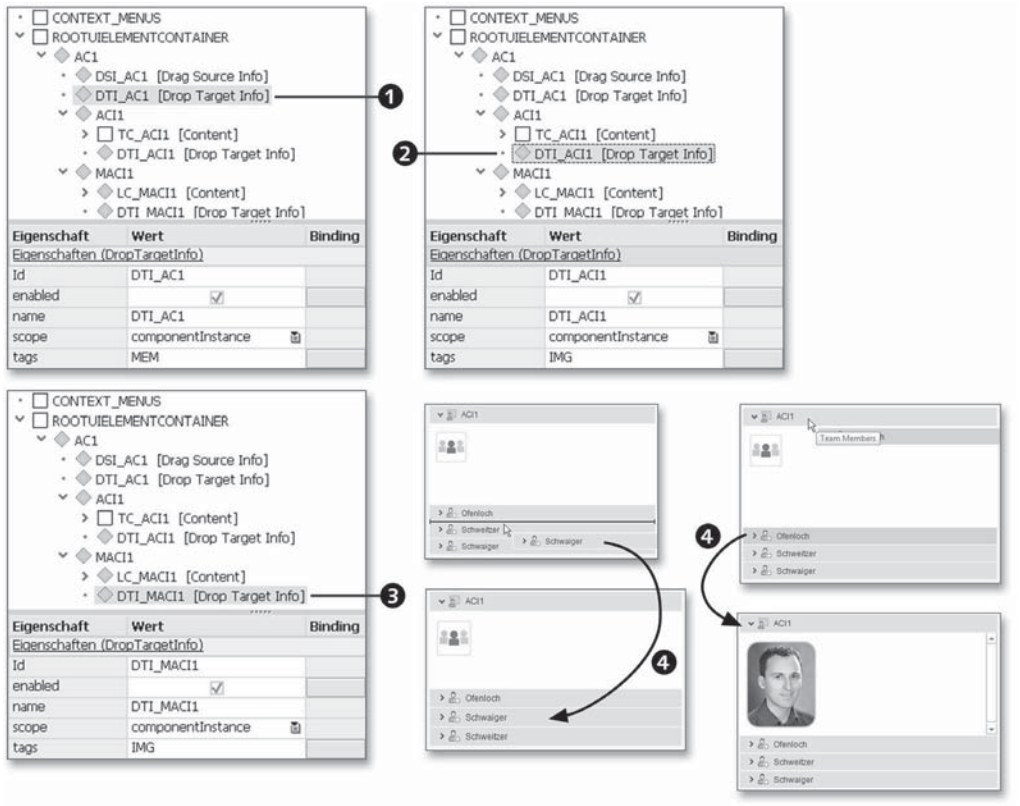

Abbildung 5.24 Accordion, MultipleAccordionItem, AccordionItem DropTargetInfo

Beschreibung

Im unteren Bereich der Abbildung sehen Sie die Visualisierung des Fallenlassens auf eine Einfügeposition durch den Benutzer ❹. Dieser

hat sich im linken Beispiel für das Einfügen zwischen den Items entschieden. Im rechten Beispiel wird ein Drop auf ein Item ausgeführt und bewirkt eine Reaktion, die im Ereignis `onDrop` des `Accordion` implementiert ist.

Mit den Attribuierungen der UI-Elemente ist nur die halbe Strecke zum Ziel genommen. In der Aktionsbehandler-Methode des `onDrop`-Ereignisses müssen Sie die Elemente entsprechend der Aktion des Benutzers programmatisch verschieben, verändern und anpassen. Dazu benötigen Sie die Parameter der Aktionsbehandler-Methode, die durch das Ereignis `onDrop` definiert sind.

Ereignis »onDrop«

Parameter

Das Ereignisobjekt `WDEVENT`, das an die Behandlermethode des Ereignisses übergeben wird, liefert neben den bereits in Abschnitt 5.2, »Tree«, besprochenen Parametern `ID`, `MIME_TYPE` und `TAGS` die folgenden Informationen für die Implementierung:

- `CONTEXT_ELEMENT`: Der Parameter `CONTEXT_ELEMENT` ist immer initial.
- `ITEM`: Der Parameter `ITEM` liefert die Bezeichnung des betroffenen Items aus dem `Accordion`.
- `ITEM_PATH`: Der Parameter `ITEM_PATH` hält den Pfad zum Element im Context, falls das betroffene Item ein `MultipleAccordionItem` ist. Ist das Element ein `AccordionItem`, ist `ITEM_PATH` leer.
- `OFFSET`: Der Parameter `OFFSET` hält die Position relativ zum `ITEM`. Falls `OFFSET` den Wert 1 hat, wurde nach dem (unterhalb des) Item(s) gedroppt, und falls er den Wert -1 hat, wurde vor dem (oberhalb des) Item(s) gedroppt. `OFFSET` nimmt den Wert 0 an, wenn ein Drop auf ein Item durchgeführt wird.

5.8 PanelStack

Ein `PanelStack` umfasst mehrere `Panel`-UI-Elemente, die immer sichtbar sind, geöffnet und kollabiert werden können. Die inaktiven Panels werden entweder als Tabs dargestellt oder als Eintrag in einem Menü. Im `PanelStack` können mithilfe von Drag & Drop ein oder mehrere Panels verschoben werden.

[»] **Voraussetzung für Drag & Drop**

Drag & Drop wird für den gesetzten Anwendungsparameter `WDUIGUIDELIND=GL20` unterstützt. Weitere Informationen zum Setzen der Anwendungsparameter finden Sie in Anhang A, »Anwendungsparameter und URL-Parameter«.

DragSource

Ein Panel im `PanelStack` kann als Drag Source verwendet werden. In Abbildung 5.25 sehen Sie die `DragSourceInfo` des `PanelStack` ❶, über die Sie definieren, dass der Benutzer den Drag für ein Panel ausführen kann.

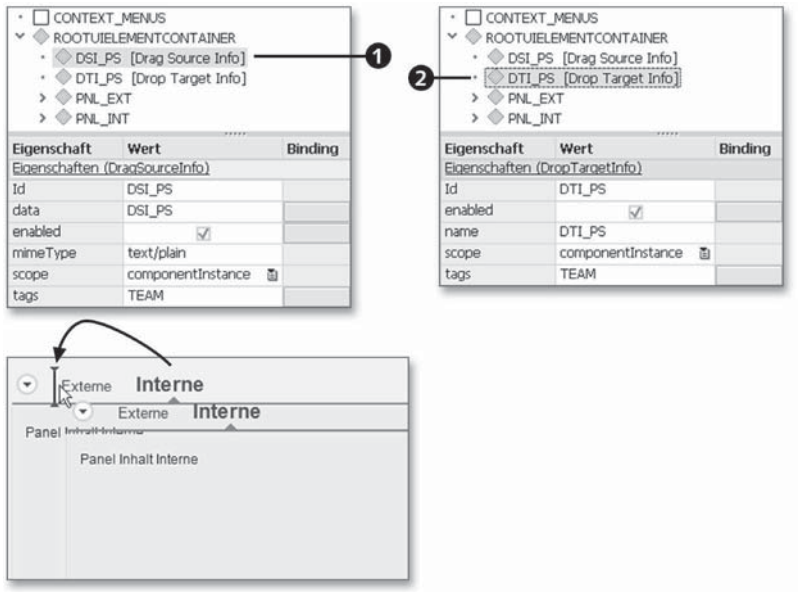

Abbildung 5.25 PanelStack – DragSourceInfo und DropTargetInfo

DropTarget

In der `DropTargetInfo` des `PanelStack` ❷ werden die bekannten Eigenschaften gepflegt. Dabei werden unterschiedliche Einfügepositionen für den Drop im `PanelStack` zur Verfügung gestellt:

- als erstes Panel (im unteren Teil von Abbildung 5.25 dargestellt)
- als letztes Panel
- zwischen den Panels

Ereignis »onDrop«

Das Ereignis `onDrop` des UI-Elements `PanelStack` wird ausgelöst, wenn ein Panel auf dem `PanelStack` losgelassen wird. Das Ereignisobjekt `WDEVENT`, das an die Behandlermethode des Ereignisses übergeben wird, liefert neben den bereits in Abschnitt 5.2, »Tree«, besprochenen Parametern `ID`, `CONTEXT-ELEMENT` und `TAGS` die folgenden Informationen für die Implementierung:

- `DATA`: Der Parameter `DATA` beinhaltet die Daten aus der Drag Source, d. h. die Eigenschaft `data` der `DragSourceInfo`.
- `MIME_TYPE`: Der Parameter `MIME_TYPE` beinhaltet den MIME-Typ aus der Drag Source, wird derzeit aber nicht verwendet.
- `PANEL_ID`: Der Parameter `PANEL_ID` beinhaltet die ID des Panels, auf das, vor das oder hinter das gedroppt wurde.
- `OFFSET`: Der Parameter `OFFSET` liefert die Position des Drops relativ zur `PANEL_ID`. Der Wert –1 bedeutet, dass der Drop über (vor) dem Panel erfolgt, und der Wert +1 bedeutet, dass er unter (hinter) dem Panel erfolgt.

5.9 Operationsmodi

Beim Drag & Drop kristallisieren sich bestimmte Verwendungsmuster des Benutzers heraus. Wir beschreiben diese im Folgenden.

5.9.1 Browse & Collect

Beim Browse & Collect wählt ein Benutzer einen Eintrag bzw. mehrere Einträge aus und zieht diese an eine bestimmte Position auf einem anderen UI-Element. Dadurch werden Daten entweder verschoben oder kopiert. Für alle Zeilen wird die gleiche `DragSourceInfo` verwendet. Überall, wo ein `DropTargetInfo` definiert ist, können Einträge hineinkommen. Browse & Collect steht für die folgenden UI-Elemente zur Verfügung:

- `ItemListBox`
- `Table`

- `CTable`
- `Tree`

5.9.2 Generisches Drag & Drop

Beim generischen Drag & Drop zieht der Benutzer ein UI-Element fort und lässt es auf ein `DropTarget` fallen. Dazu können `Image`, `GridLayout` und `MatrixLayout` verwendet werden. Wie Sie in Abschnitt 5.1.3 erfahren haben, wird in diesem Fall ein `DropTarget` als verschalendes UI-Element verwendet.

5.9.3 Laufzeit-Authoring

Beim Laufzeit-Authoring zieht der Benutzer ein UI-Element an eine andere Stelle in einem `GridLayout` oder `MatrixLayout` (siehe Abschnitt 5.3). Dabei klickt er den Griff des UI-Elements an und zieht diesen an die gewünschte Stelle. Damit ein UI-Element als Griff definiert werden kann, muss es die Eigenschaft `isDragHandle` besitzen. Die folgenden UI-Elemente bieten diese Eigenschaft an:

- `Image`
- `Caption`
- `SectionHeader`
- `PanelStack`

Zu einer Drag-&-Drop-Operation gehört nicht nur die Quelle, sondern auch ein Ziel, das das Ereignis `onDrop` für die Behandlung des Drops anbietet. Für die folgenden UI-Elemente steht das Event `onDrop` zur Verfügung:

- `Tree` (über `AbstractTreeNodeType`)
- `Accordion`
- `DropTarget`
- `GridLayout`
- `MatrixLayout`
- `ItemListBox`
- `Table`
- `CTable`
- `PanelStack`