Een typisch kenmerk van web 2.0 applicaties is het gebruik van Drag and Drop (DnD). Voorbeelden hiervan zijn te vinden in de Windows verkenner, diverse mailprogramma's, maar ook web-winkels waar je producten in je winkelmandje kunt slepen.

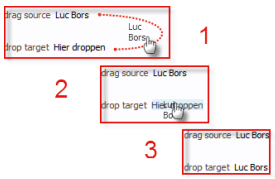
ADF@WORK

Introductie Drag and Drop

ADF biedt ontwikkelaars ook de mogelijkheid om Drag and Drop (DnD) te implementeren. Het concept van DnD is vrij eenvoudig. Je hebt een dropTarget eventueel met een listener om op de dropEvent te reageren. Daarnaast heb je een dragSource die de gegevens bevat die naar de target kunnen worden gesleept. Aan de hand van een paar voorbeelden heb je dit concept snel onder de knie.
De meest eenvoudige vorm van DnD maakt geen gebruik van code. Met een combinatie van attributeDragSource en attributeDropTarget kun je heel eenvoudig DnD toevoegen aan je applicatie om de waarde van attributen te draggen en droppen.

```
<af:inputText value ="Luc Bors">
<af:attributeDragSource attribute="value"/>
</af:inputText>

<af:outputText value="Hier droppen">
<af:attributeDropTarget attribute="value"/>
</af:outputText>
```



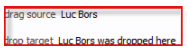
De werkelijkheid is meestal niet zó eenvoudig.

De dropListener

Een DnD actie zal vaak worden gebruikt in combinatie met een dropListener. Deze dropListener kun je gebruiken om manipulatie en extra acties uit te voeren naar aanleiding van een DnD actie. De dropListener heeft een DropEvent als argument. Op basis daarvan heb je alle informatie om te bepalen wat er 'zojuist' is gebeurd: wat was de dropTarget, wat was de DragSource, wat is er eigenlijk gedropt. De dropListener geeft als resultaat de uit te voeren actie terug. Deze actie kan één van de waarden NONE, MOVE, COPY of LINK bevatten.

```
public DnDAction handleDrop(DropEvent drEv) {
try {
   DataFlavor<String> df =
   DataFlavor.getDataFlavor(String.class);
   String drVal =
     drEv.getTransferable().getData(df);
   if(drVal==null) {return DnDAction.NONE;}
   else
     {drVal = drVal +" was dropped here ";
      ValueExpression val = createVex(drVal);
        drEv.getDropComponent().setValueExpression("value",val);}
  return DnDAction.COPY;
}
catch(Exception ex) {........}
}
```

Het gebruik van de dropListener leidt tot een ander resultaat, omdat de waarde van de dropTarget wordt gewijzigd.



Geavanceerde mogelijkheden

Naast attributen, zoals hiervoor beschreven, kun je ook objecten, collecties en componenten gebruiken in een DnD actie. Het principe is hetzelfde, alleen de code in de dropListener kan in sommige gevallen wat complexer worden. Daarnaast moet je in de dropTarget gebruik maken van de <af:dataFlavor/> component om aan te geven wat het type is van de gedropte data..

```
<af:dropTarget
    dropListener="#{DnDBean.handleDrop}">
       <af:dataFlavor
       flavorClass="java.lang.Object[]"/>  </af:dropTarget>
```

Native support

Er zijn componenten die ingebouwde ondersteuning voor DnD hebben. Een uitstekend voorbeeld hiervan is de <af:panelDashboard/> component. Deze component functioneert automatisch als dropTarget. Alle componenten in de panelDashboard (over het algemeen panelBox componenten) moeten de <af:componentDragSource/> als child hebben en dan werkt de DnD direct.

```
<af:panelBox id="pb6" text="Box 6">
   <af:componentDragSource/>
</af:panelBox>
```

Tenslotte

Dit artikel beschrijft slechts een deel van de mogelijkheden en geeft een handvat voor het werken met DnD. Het gebruik van DnD kan een applicatie gebruiksvriendelijker maken, maar zorg er in ieder geval voor dat gebruikers een alternatief hebben voor DnD zoals bijvoorbeeld een knop.

Extra Resources:

- *DnD in panelDashboard: http://technology.amis.nl/blog/5781/adf-11g-paneldashboard-configure-your-own-dashboard*
- *DnD in trees: http://technology.amis.nl/blog/3302/dropping-trees*

Luc Bors (luc.bors@amis.nl) werkt als technisch specialist/ architect en is ADF Expertise Lead bij AMIS Services.