In deze rubriek worden specifieke onderdelen van het Oracle Application Development Framework besproken. De bedoeling is dat de lezer daarna zelf met het onderwerp aan de slag kan.

ADF@WORK

JavaScript in ADF Faces

Tijdens Oracle Open World 2009 presenteerde Duncan Mills 'Fusion Design Fundamentals'. Hij opperde '5 vuistregels'. De eerste regel was :'Thou shalt not write JavaScript'. Als iedereen zich letterlijk aan deze regel hield, was dit artikel overbodig. Gelukkig voegde Duncan er 'unless' aan toe. Er zijn situaties waarin het gebruik van JavaScript het laatste redmiddel is. In deze situaties kun je gebruik maken van de JavaScript API uit het ADF Faces Rich Client Framework.

JavaScript in de pagina

JavaScript functies kunnen direct in de pagina worden opgenomen, of in losse JavaScript files die vervolgens worden geladen. In dit artikel wordt uitgegaan van de eerste situatie. Voordeel hiervan is dat dit gecached kan worden, nadeel is echter dat de pagina groot kan worden.
Om een JavaScript functie op te nemen in een ADF Faces pagina kun je het beste gebruik maken van de <af:resource/> component. Deze component zorgt er voor dat JavaScript op een optimale manier wordt geladen. Je kunt deze component overal in de pagina opnemen, maar het is aan te bevelen dat deze component in de header van de pagina wordt geplaatst. Dit heeft twee voordelen. Het zorgt voor betere performance en garandeert dat de JavaScript functies geladen zijn voordat ze ergens in de pagina worden aangeroepen.

```
<af:resource type="javascript">
    function toonPopup(event){
       var bronCall = event.getSource();
       var teTonenPopup = bronCall.findComponent("dePopup");
       teTonenPopup.show();
      }
</af:resource>
```

De functie "toonPopup()" kan nu overal in de pagina worden aangeroepen.

ClientListener en ClientAttribute

De <af:clientListener> component wordt gebruikt om client side events te onderscheppen en op basis daarvan een JavaScript functie aan te roepen.

```
<af:commandLink text="Laat een popup zien"  id="comLink"
                            partialSubmit="true">
    <af:clientListener method="toonPopup"
                                type="action"/>
</af:commandLink>
```

Zodra de commandLink gebruikt wordt, verschijnt de popup. Soms is het nodig om extra informatie in de vorm van argumenten mee te geven aan de JavaScript functie. De <af:clientAttribute> component maakt dit mogelijk.

```
      <af:clientAttribute name="eenAttribuut"
                                    value="Waarde van een Attribuut"/>
```

De waarde in dit voorbeeld is een statische waarde. Je kunt ook verwijzen naar een EL expressie. Een component kan meer dan 1 <af:clientAttribute> bevatten. De waarde van dit attribuut kan nu in JavaScript door middel van getProperty() worden opgevraagd.

```
            var waarde = bronCall.getProperty('eenAttribuut');
            alert (waarde);
```

JavaScript in Managed Beans

Er zijn ook situaties waarin het nodig is om een JavaScript functie vanuit een managed bean aan te roepen, bijvoorbeeld als je een popup wilt tonen nadat er op de server een actie succesvol is uitgevoerd. Dit kan met behulp van de ExtendedRenderKitService.

```
FacesContext fc = FacesContext.getCurrentInstance();
        ExtendedRenderKitService erks =
        Service.getRenderKitService(fc,
        ExtendedRenderKitService.class);
        Erks.addScript(fc, "toonPopup();");
```

De addScript() methode zorgt ervoor dat de toonPopup() methode wordt uitgevoerd en dat de popup wordt getoond. Naast de beschreven mogelijkheden bestaat ook nog the <af:serverListener> component om managed bean code aan te roepen vanuit JavaScript.

Tot Slot

Voor situaties waarin intensieve interactie met JavaScript noodzakelijk is, zoals bij een integratie met googlemaps, is de JavaScript API van ADF zeer waardevol. Op de weblog van Frank Nimphius staan aanvullende voorbeelden over het gebruik van javascript in ADF: http://thepeninsulasedge.com/frank_nimphius/adf-faces-rich-client-javascript-programming/.
Maar let op: te pas en te onpas gebruik van JavaScript kan leiden tot extra onderhoud en JavaScript is lastig te debuggen. Vandaar het bestaan van regel 1. De 'tenzij' in regel 1 is geen vrijbrief voor het gebruik van JavaScript. Denk na over alternatieve oplossingen.

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