

WebUtil backporten naar Forms 6i

Forms10g features voor webforms applicaties

WebUtil. Een van de hot features van Forms 10g. Met WebUtil krijgt een webforms applicatie weer volledige controle over de client PC. Zo is het mogelijk om vanuit een webformsapplicatie (binnen de browser) te integreren met software die op de PC is geïnstalleerd (buiten de browser), bijvoorbeeld de Microsoft Office suite. Uiteraard is het ook een uitkomst voor bedrijven die noodgedwongen nog steeds gebruik maken van het Client/Server computing model omdat zij leunen op zaken als OLE. In dit artikel licht Harold Gerritsen de functionaliteit van WebUtil toe, legt hij uit waarom Oracle er voor kiest WebUtil uitsluitend uit te brengen voor Forms9i en Forms10g en geeft hij een stap voor stap instructie hoe WebUtil toch gereed te maken is voor gebruik in Forms6i, het zogenaamde 'backporten'.

Ontwikkelaars die regelmatig een blik werpen op OTN hebben het fenomeen WebUtil waarschijnlijk langzaam zien ontstaan. Eerst was er enkel linkje met een summere begeleidende tekst in de trant van 'Add-on voor Oracle Forms'. Gaandeweg werden er enkele korte whitepapers aan toegevoegd. Toen de marketeers van Oracle echter begonnen te begrijpen wat dit stukje technologie eigenlijk inhield was de beer los. Inmiddels heeft WebUtil op OTN een heuse portal (zie afbeelding 1), waar een schat aan documentatie is te vinden en alle software is te downloaden. Maar het moet ook gezegd worden. Hoewel de kern van de WebUtil software een minimale omvang heeft (past op een floppy disk) is de functionaliteit fenomenaal. Maar voor we dieper in WebUtil duiken eerst even een korte beschouwing van WebUtil in relatie tot Forms releases.

Forms6i, Forms9i, Forms10g

Voor veel klanten begint het Forms-landschap een dicht woud te worden. Het gros van de bedrijven hijgt nog na van een inhaalslag om van niet gesupporte versies te migreren naar – in de meeste gevallen- Forms6i. En Oracle heeft het in z'n marketing-uitingen alleen nog maar over Forms9i en Forms10g. Laten we daarom eerst wat duidelijkheid scheppen. De verschillen tussen Forms9i en Forms10g zijn minimaal.

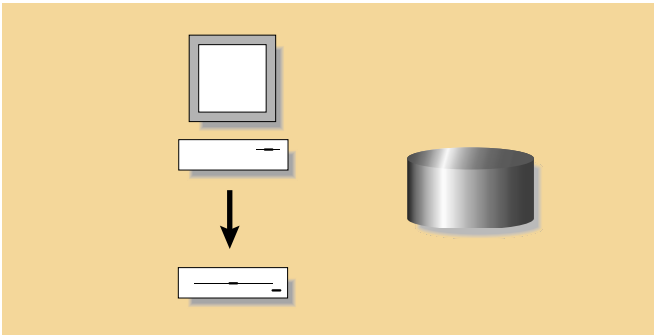
In feite is Forms10g niet meer dan een puntrelease van Forms9i. Het beschikbaar komen van de 10g database server dwingt echter af (althans, op last van de marketingafdeling) dat alle Oracle producten die na de database server verschijnen meelopen in de 10g benaming cq. nummering.

Het verschil tussen Forms6i en Forms9i is een heel ander verhaal. Allereerst is Forms9i uitsluitend te runnen op basis van de Webforms architectuur. Client/Server (C/S) is niet meer mogelijk. Daarnaast is er veel backward-compatibility verwijderd uit Forms9i. Het argument van Oracle hiervoor is natuurlijk dat daar een afgeslankt, snel Formsproduct door is ontstaan. Voor Oracle Development in de US zal de sourcecode van het Formsproduct zeker veel beheersbaarder zijn geworden nu er

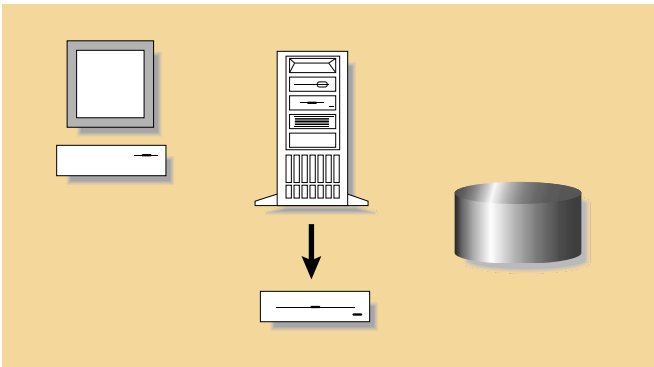


Afbeelding 1. Het WebUtil Center op OTN.

zoveel legacycode gestript is. Of de klant hier zo blij mee moet zijn is maar de vraag. Naast het aspect van compatibiliteit is op dit moment niet elke klant onder de indruk van de volwassenheid van Forms9i. De reden daarvoor is dat het Formsproduct ingrijpender op de schop is gegaan dan het in eerste oogopslag lijkt. Wat weinigen weten, is dat Oracle het met de 9i productlijn eindelijk heeft aangedurfd de verschillende productlijnen (ontwikkeltools, database server) weer te baseren op dezelfde basiscomponenten. Deze componenten vormen de onderste



Afbeelding 2a. Het aanspreken van het onderliggende platform (Client/Server architectuur).



Afbeelding 2b. Het aanspreken van het onderliggende platform (Webarchitectuur).

laag van alle Oracle producten, en hebben welluidende namen zoals 'Required Support Files'. In het verleden, met het ontstaan van de Oracle8i database, is er een splitsing ontstaan in de versies van deze basiscomponenten. Het formsproduct (Forms6i) werd verder ontwikkeld op basis van de componenten van de (oude) Oracle 8.0 database, en Oracle8i kreeg doorontwikkelde basiscomponenten. Vanuit het oogpunt van software configuration management was deze situatie voor Oracle natuurlijk een draak. Daarom is de 9i versie van Forms weer herschreven om gebruik te maken van dezelfde basiscomponenten, met alle positieve en negatieve gevolgen van dien. De beheersinspanningen van Oracle Development lopen echter pas ingrijpend terug als er geen onderhoud meer uitgevoerd hoeft te worden op Forms6i. En daarom worden kosten nog

moeite gespaard om de gehele Forms-gemeenschap over te krijgen naar Forms9i. Zo is er afgelopen zomer een aanbieding geweest om met korting de applicatieserver aan te kunnen schaffen, onder de voorwaarde dat er direct geupgrade zou worden naar Forms9i. In dit licht is het niet zo vreemd, dat Oracle met betrekking tot WebUtil doet of Forms6i niet meer bestaat of nog amper gebruikt wordt. Maar niets is minder waar. Op dit moment heeft in Nederland slechts een fractie van de bedrijven die met een gesupporte Formsversie werken, release 9i. Kortom, reden genoeg om een versie van WebUtil te willen hebben die geschikt is voor Forms6i.

Het probleem

Nu we even stilgestaan hebben bij de context van WebUtil, is het tijd geworden WebUtil zelf wat gedetailleerder te bekijken. Hiervoor kijken we eerst naar de essentie van het probleem dat WebUtil oplost. Dat wordt geïllustreerd in afbeelding 2a en 2b. In afbeelding 2a is te zien hoe flexibel Forms-applicaties in de Client/Server architectuur zijn waar het aankomt op het aanspreken van het onderliggende besturingssysteem en de daarop draaiende applicaties. Ga maar na, met behulp van standaard softwarecomponenten die bij Forms meegeleverd werden (zoals de Designer/2000 Windows Utilities library, d2kwutil.pll) kan een form bijvoorbeeld lezen en schrijven in de registry van de client PC. Of kan de 'Browse File-dialog' van Windows gestart worden om een item in het Form te vullen met de naam van een file die op de PC staat. Ook kan een C/S Form met de standaard built-in 'Host' een opdracht uitvoeren op het onderliggende besturingssysteem, of met de standaard 'TEXT_IO' functionaliteit bestanden openen, lezen, schrijven en sluiten. Hiervoor maakt het niet uit of die bestanden lokaal op de PC staan, bijvoorbeeld op een Floppy, CD-ROM of harde schijf, of elders, benaderbaar als netwerkdrive. Een laatste feature waarvan voor specialistische toepassingen redelijk veel gebruik is gemaakt, is het aan kunnen roepen van zelfgemaakte zogenaamde Dynamic Linked Libraries (DLL's). Zo kan bijvoorbeeld in de taal C een specifieke functionaliteit worden geprogrammeerd en gecompileerd tot een DLL. In C/S Forms kunnen de routines in zo'n library volledig transparant aangesproken worden met behulp van de standaard 'ORA_FFI' functionaliteit, wat staat voor 'Foreign Function Interface'. Bedrijven die applicaties hebben ontwikkeld die gebruik maken van (delen van) deze functionaliteit werden jaren geleden onaangenaam verrast. Toen Oracle de C/S architectuur begon af te zweren om plaats te maken voor de Webarchitectuur, konden zij niet meelopen in deze ontwikkelingen. Kijken we naar afbeelding 2b, dan zien we waarom. Het goede nieuws was dat de programmatuur die gebruik maakte van genoemde features vrijwel altijd zonder aanpassing gewoon bleef werken zodra deze werd gedraaid als Webforms. Alleen was de werking en het gedrag ervan wèl veranderd. Waar voorheen bijvoor-



Afbeelding 3. Het consolewindow van Jinitiator bij gebruik van WebUtil.

beeld een lokale drive werd aangesproken, werd nu de drive van de applicatieserver aangesproken. Op die machine draaide de applicatie immers, ook al was hij wel zichtbaar op de client PC. Voor een aanzienlijk deel van het gebruik van deze features was de situatie wel om te buigen. Door bepaalde zaken van een Client PC te 'sharen', konden deze vanuit de applicatieserver alsnog worden aangesproken. Maar deze constructie bood niet alle gevallen soelaas. En in die gevallen was er vaak geen enkel alternatief. Gevolg: een deel van de klanten van Oracle kan gewoonweg niet upgraden naar Forms9i (alleen Webforms) of Forms6i draaien via het Web.

De oplossing

Ruim een jaar geleden heeft Oracle een projectteam opgezet om het geschetste probleem te adresseren. De opdracht die het team meekreeg was het ontwikkelen van een software-oplossing die de tekortkomingen van de Webconfiguratie ten opzichte van de Client/Serverconfiguratie zouden aanvullen. Anders gezegd: het eindproduct moest ondersteuning gaan bieden voor migratie van C/S naar het Web, en er zouden extra faciliteiten geboden moeten worden voor Webapplicaties. Of, nog concreter, de basale C/S functionaliteit moest weer geboden worden, maar ook de specifieke add-on functionaliteit zoals D2KWUTIL en, last but not least, specifieke Web features, zoals bijvoorbeeld File Transfer. Als we kijken naar de doelstelling van het project, dan kunnen we inmiddels stellen dat die indrukwekkend goed bereikt is. Allereerst biedt WebUtil voor Webforms mogelijkheden voor gebruik van platform-onafhankelijke faciliteiten zoals 'HOST', 'TEXT_IO', 'TOOL_ENV', 'ORA_FFI' en user exits. Maar daarnaast worden ook specifieke Windowsfaciliteiten ondersteunt, zoals 'OLE2', 'FORMS_OLE', 'DDE' en 'D2KWUTIL'. Ondanks het feit dat Java enorm oprukt in de Oracle tooling, is het met name de PL/SQL gemeenschap die werd geconfronteerd met het geschetste probleem. Daarom is WebUtil dusdanig opgezet dat elke PL/SQL ontwikkelaar die de API's van de bestaande functionaliteit kent, zonder aanvullende studie de WebUtil oplossing kan gebruiken. En inderdaad. Zo simpel is het geworden. Voor

elke API is er een broertje gemaakt met voorvoegsel 'CLIENT_'. Door in een aanroep eenvoudigweg de naam van de aan te roepen API aan te passen, vindt de integratieactie niet plaats op de applicationserver, maar op de client PC zelf. Een uitzondering daarop is de ORA_FFI interface. Daarbij heeft de WebUtil variant niet helemaal dezelfde interface en is de omzetting iets ingewikkelder geworden.

De praktijk

Als we kijken wat het inhoudt om een applicatie uit te breiden met WebUtil blijkt dat enorm mee te vallen. Nadat de applicatie met WebUtil geconfigureerd is, is het voor iedereen eigenlijk heel erg transparant. Allereerst de eindgebruikers, die merken helemaal niets. Het enige bewijs voor de eindgebruiker dat er WebUtil onder de motorkap wordt gebruikt, is terug te vinden in het consolewindow van Jinitiator. In afbeelding 3 is te zien dat er een Javalibrary wordt gedownload naar de client (webutil.jar) en dat de WebUtil functionaliteit 'geregistreerd' wordt. Om dat te bereiken, dient de beheerder van de application server WebUtil te configureren. Er moet een aantal WebUtil bestandjes in de juiste directories worden gezet, en de configuratiefile van



Afbeelding 4a. Compilatiefouten in WebUtil PL/SQL library (webutil.pll).



Afbeelding 4b. WebUtil object library (webutil.olib) direct na conversie.

de webserver moet worden aangepast. Het voert te ver om deze acties woordelijk te beschrijven. De 'WebUtil Familiarisation Manual' doet dat uitstekend. Aardig is wel om te vermelden dat er een automatische file-deploy functionaliteit in WebUtil zit. Door de naam, locatie en grootte van een bestand te specificeren (bijvoorbeeld een eigen Dynamic Linked Library) wordt deze automatisch gedeployed en gecached naar de client PC, waarna deze door de nieuwe API's kan worden aangesproken. De beheerder moet voorts een package creëren in de database, de package WEBUTIL_DB.

De ontwikkelaars tenslotte, hoeven slechts twee simpele handelingen te verrichten om WebUtil te gebruiken. Allereerst moet de WebUtil PL/SQL Library (webutil.pll) aan het te enablen form worden attached. Merk op dat dit het beste per form kan gebeuren en niet aan een algemeen gebruikte applicationlibrary. Hoewel dit laatste werk zou besparen, legt dit voor de forms die niet actief van WebUtil gebruik maken een onnodig beslag op resources. De tweede actie is dat er een blok en een parameter worden toegevoegd aan het te enablen form. Deze twee objecten moeten worden gesubclassed van de WebUtil Object Library (webutil.olb). Na het doorvoeren van deze twee acties



Afbeelding 4c.
Object library
maintenance form met
WebUtil objecten.



Afbeelding 4d.
Standaardfouten na
conversie: item type
en trigger logica.

zijn de toegevoegde WebUtil API's (beginnend met de naam 'CLIENT_') volledig beschikbaar. Waarschijnlijk zijn de hier beschreven componenten op dit moment nog een black box. Maar voor geïnteresseerden die de backport acties die hierna worden beschreven hebben uitgevoerd, heeft WebUtil waarschijnlijk geen geheimen meer.

Backporten doe je zó

Wellicht dat men in Redwood Shores nog tot inkeer komt en er ten tijde van verschijnen van dit artikel allang een WebUtil voor Forms6i download voorhanden is. Anders moeten we het maar gewoon zelf doen. Wellicht ten overvloede: de navolgende instructie is bedoeld als een stap-voor-stap handleiding voor ontwikkelaars. Lezers die slechts geïnteresseerd zijn in de concepten en mogelijkheden van WebUtil willen dit echt niet meer weten. Eerst nog even een disclaimer: de hierna beschreven manier om de WebUtil software te backporten wordt geboden 'as is'. Het gebruik van de door backporten ontstane software geschiedt op eigen risico. Dat klinkt zwaarder dan het is. Het is een inmiddels bij meerdere klanten beproefde methode, en verschillende Oracle Forms 6i applicaties die op deze manier met WebUtil zijn uitgerust, hebben de productiestatus. En dat WebUtil pas in Forms 10g volledig geïntegreerd is met Oracle Forms is ook geen probleem. In essentie is WebUtil een combinatie van enkele softwarecomponenten die je zelf met de Oracle tooling (Oracle Forms, JDeveloper) gemaakt zou kunnen hebben. Door ze nu te downloaden, te backporten en te bundelen met de rest van de applicatie is er technisch gezien eigenlijk geen verschil met de toekomstige situatie in Forms 10g.

Aanpassen

Nu het backporten zelf: wat hebben we nodig? Allereerst ontkomen we er niet aan om een werkende Oracle Forms9i omgeving te hebben, al hebben we die slechts één minuutje nodig. Een snelle mailwisseling met een bevriende relatie met Forms9i is hiervoor een goed alternatief. Daarnaast hebben we de Forms6i omgeving nodig waarnaar we willen backporten, en last but not least de Forms9i versie van de WebUtil software. Deze wordt op moment van schrijven van dit artikel geleverd als add-on op Forms9i en is te downloaden van de WebUtil Portal^[1]. De navolgende instructies zijn beproefd voor WebUtil versie 1.0.2, download 'webutil_102.zip'. Het backporten betreft twee softwarecomponenten: een PL/SQL Library, webutil.pll, en een Object Library, webutil.olb. Allereerst bewerken we de PL/SQL library. Open daarvoor Oracle Forms9i en converteer de library van het binaire formaat (webutil.pll) in het tekstformaat (webutil.pld). Merk op dat de plaats om dit te doen in Forms 9i een fractie gewijzigd is (File>Convert i.p.v. File>Administration>Convert). Omdat de

[1] Oracle Forms10g WebUtil Portal. <http://otn.oracle.com/products/forms/htdocs/webutil/webutil.htm>



Afbeelding 4e.
Omzetten item type
van Chart in
Bean Area.



Afbeelding 4f.
Toevoegen trigger
logica aan dummy item.

library in tekstformaat niets meer bevat dan de programma-code, kunnen we het aldus omgezette bestand één op één inlezen in Forms6i en daar weer omzetten in binair formaat, ditmaal geschikt voor Forms6i. Voor we dat doen moeten we echter een kleine aanpassing doorvoeren. De syntax van PL/SQL in Forms9i kent iets meer mogelijkheden, en daar is in de library gebruik van gemaakt. Om te illustreren welke aanpassing we moeten doorvoeren is in afbeelding 4a een screenshot weergegeven van het met Forms6i compileren van de gebackportede library, zonder vooraf de benodigde aanpassing te hebben gedaan. We zien hier de constructie 'is table of VARCHAR2(256 char)'. Door in de gehele library in dergelijke constructies '<spatie>char' te verwijderen is de library geschikt gemaakt voor Forms6i. Het efficiëntst kan dit geschieden door de library in tekstformaat -vóór deze weer wordt geconverteerd met Forms6i- met een editor te editen (Search and Replace; met beleid natuurlijk, '<spatie>char' komt ook in andere constructies voor). De aldus bewerkte library kan vervolgens worden geconverteerd en gecompileerd met Forms6i, en is daarna gereed voor gebruik. Merk op dat er ook webutil database-objecten moeten zijn, wil de library foutloos compileren.

Library

Het bewerken van de Object Library heeft meer voeten in aarde. De benodigde acties in Forms9i zijn vergelijkbaar met het bewerken van de PL/SQL Library. Open Forms9i en converteer de library van het binaire formaat (webutil.olb) in het tekstformaat (webutil.olt). Open vervolgens Forms6i en converteer de library weer naar het binaire formaat, ditmaal geschikt voor Forms6i. Door het achterwege blijven van errors in deze stappen ben je snel geneigd te denken dat de Object Library hiermee succesvol gebackport is. Mocht je die conclusie trekken dan staat dat garant voor dagen debuggen om WebUtil aan de praat te krijgen. Software, zo ook die van Oracle, is nu eenmaal niet vaak forward compatible. In afbeelding 4b t/m 4h worden de stappen geïllustreerd die nog nodig zijn om de library te corrigeren. Nadat de library geconverteerd is naar Forms6i kan deze daar worden geopend. In de navigatortree van Forms zien we dat de library slechts één tab heeft, met de naam 'MAIN'. Als we op deze tab dubbelklikken, zien we in het rechterwindow dat hier twee objecten in zijn gedefinieerd, met de respectievelijke namen WEBUTILCONFIG en WEBUTIL (zie afbeelding 4b). Eigenschappen van objecten in een object library kunnen we



Afbeelding 4g.
Toevoegen trigger
logica aan overige
items.



Afbeelding 4h.
Slepen van objecten
van maintenance form
naar nieuwe olb.

slechts bekijken en aanpassen in de context van een form. Daarom maken we voor dit doel een leeg form aan, met de naam WEBUTILOLM, wat staat voor WEBUTIL Object Library Maintenanceform. Het is goed programmeergebruik om voor elke olb een bijbehorende olm te hebben. Eerst verwijderen we het default in het form aanwezige WINDOW1. Als het form echt leeg is, slepen we de objecten WEBUTILCONFIG en WEBUTIL uit de object library (het rechterwindow) naar het maintenance form, en wel naar de naam van het form. De dialoog die opkomt met de vraag 'Subclass' of 'Copy' beantwoorden we met 'Copy'. Het maintenance form bevat dan alle objecten uit de objectlibrary (zie afbeelding 4c). Zodra we de objecten wat meer in detail gaan bekijken, blijkt dat er inconsistenties zijn ten opzichte van de definities van de objecten in Forms9i waar we zijn begonnen. Kijken we in afbeelding 4d, dan zien we dat het merendeel van de items in het WEBUTIL blok van het type 'Chart' is, getuige het staafdiagramicoontje voor de itemnamen. Dit is onjuist. Daarnaast blijkt de triggerlogica van de itemtriggers volledig verdwenen te zijn. Let hierbij op: zodra het compilatiescherm met de ontbrekende logica zonder aanpassing wordt gesloten, treedt er onvoorspelbaar gedrag op. De trigger kan verdwijnen of zelfs het hele item.

Corrigerende actie

Zodra er een leeg compilatiewindow in beeld is dient er direct 'null;' te worden gecodeerd, of direct de juiste code zoals in één van de volgende stappen wordt uitgelegd. Als eerste corrigerende actie moeten we het Item Type veranderen van de items van het WEBUTIL blok waarvan de naam begint met 'WEBUTIL_'. Hiervoor selecteren we de respectievelijke items en wijzigen we de property 'Item Type' van 'Chart' in 'Bean Area' (zie afbeelding 4e). Hiermee geven we aan dat de items worden gebruikt als placeholder voor een Java Bean. Overigens geeft de property 'Implementation Class' aan welk stuk Java code met het item geassocieerd moet worden. De tweede corrigerende actie bestaat uit het opnieuw definiëren van de verdwenen trigger logica. Dit is een kleine wijziging, aangezien er slechts twee varianten zijn van de vast te leggen logica. De When-Button-Pressed trigger van het Dummy item krijgt de volgende code (zie afbeelding 4f):

```
begin
  if :system.cursor_block = 'WEBUTIL'
  then
    next_block;
  end if;
  WebUtil_Core.ShowBeans(false);
end;
```

De overige items van het WEBUTIL blok hebben allen een identieke When-Custom-Item-Event trigger. De code hiervan is (zie afbeelding 4g):

```
Begin
  WebUtil_Core.CustomEventHandler(:system.custom_item_event,
                                :system.custom_item_event_parameters);
end;
```

Nadat deze correcties zijn doorgevoerd is het maintenance form eigenlijk volledig up-to-date. Wat nog rest is om de inhoud van het maintenance form onder te brengen in een echte Object Library. Hiervoor sluiten we –als we dat al niet gedaan hadden– alle webutil Object Libraries in Forms6i en maken we een nieuwe object library aan met de onderscheidende naam WEBUTIL60. Vervolgens creëren we in deze library een librarytab met naam 'MAIN' en label 'WebUtil Objects'. We selecteren de object group 'WEBUTIL' (Let op! Niet het blok zelf maar de object group) en de parameter 'WEBUTILCONFIG' en slepen deze van het linker window naar het rechter –nu nog lege– WebUtil Objects window (zie afbeelding 4h). Save en sluit tenslotte de aldus ontstane object library WEBUTIL60.olb en eventueel de object library maintenance form. In principe is het maintenance form niet meer nodig, tenzij er een aanpassing nodig is in de object library. Dit zou bijvoorbeeld kunnen bij het beschikbaar komen van een nieuwe versie van WebUtil of als voorgaande stappen niet correct zijn uitgevoerd. Hebben we de PL/SQL library en de Object Library echter op de beschreven wijze gebackport, dan kunnen we in Forms6i heus gebruikmaken van de WebUtil functionaliteit.

Resumerend

De functionaliteit van WebUtil is vele malen groter dan in dit artikel uit de doeken is gedaan. Om met alle features van WebUtil vertrouwd te raken is het een must de 'WebUtil Familiarisation Manual' door te nemen. Deze is op OTN terug te vinden op de speciale WebUtil pagina I. Hier staan ook verschillende How-To documenten, on-line demonstraties, en natuurlijk de WebUtil download. Veel klanten zullen het echter jammer vinden dat WebUtil (standaard) niet geschikt is om binnen Forms6i te gebruiken. Maar met de hiervoor beschreven backport instructie is WebUtil gereed te maken voor Forms6i. Door de stappen nauwgezet te volgen is dat nog geen kwartiertje werk. En zo gaat er ook een wereld aan integratiemogelijkheden open voor klanten met Forms6i via het web.

Harold Gerritsen

is principle consultant bij A New Link bv. Hij heeft meer dan dertien jaar ervaring in het adviseren over effectief inzetten van Oracle technologie. E-mail: h.gerritsen@anewlink.nl.