Ontwikkelingshulp voor PL/SQL

De stille kracht van Java

In een serie van drie artikelen zal Lucas Jellema een overzicht geven van ideeën, patronen, richtlijnen en tools die PL/SQL ontwikkelaars kunnen overnemen uit de wereld van Java/J2EE. Hij gaat onder meer in op concepten uit de Java programmeertaal die toegepast worden in PL/SQL, maar ook op open source-tools voor onder meer logging, unit-testen, automatiseren van batch-operaties en het genereren van documentatie. Dit artikel is bestemd voor PL/SQL ontwikkelaars; kennis van Java is niet vereist. De source code voor dit artikel kan worden gedownload van http://www.amis.nl/tech_artikelen.php?id=158.

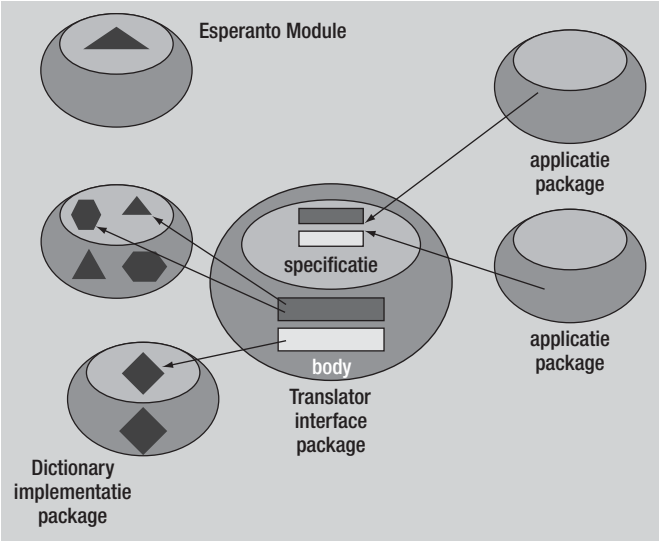
Voor een actueel artikel over Java/J2EE en PL/SQL zou je een titel verwachten als "Van PL/SQL naar Java", maar in dit verhaal is het precies omgekeerd. We gaan kijken wat zo'n tien jaar Java heeft opgeleverd voor PL/SQL- en SQL-ontwikkelaars. Als doorgewinterde PL/SQL-ontwikkelaars zou je wellicht geneigd zijn te denken: wat voor goeds kan er komen van zo'n nieuwerwetse, overgehypte, onvolwassen programmeer-technologie? Maar er is meer dan je wellicht in eerste instantie zou denken, niet in de laatste plaats omdat Java meer is dan een programmeertaal. Het is een platform, een manier van denken, bijna een life-style geworden. En dat heeft heel wat in beweging gebracht, met concrete resultaten, ook voor PL/SQL ontwikkelaars.
We zullen in dit artikel ingaan op enkele concepten en patronen die via Java bekend en populair zijn geworden en ook relevant zijn voor PL/SQL. Daarnaast is rondom Java een aantal open source tools met generieke functionaliteit ontstaan – bijvoorbeeld voor logging, unit-testen en het genereren van documentatie. Recent zijn er vergelijkbare tools beschikbaar gekomen voor de PL/SQL ontwikkel-omgeving. In dit artikel zullen we als voorbeeld Log4PLSQL bekijken, een oplossing voor Logging.

Ontwerpcontract

Een belangrijke plaats in de gesprekken en vooral de literatuur over Java wordt ingenomen door architectuur en generieke ontwerppatronen, de zogenaamde design patterns. Hoewel deze beschouwingen voor mijn gevoel soms wel eens willen doorschieten en de praktijk het volledig aflegt tegen de zuiverheid en dogmatiek van de theorie, kunnen we voor SQL- en PL/SQL-architectuur en -ontwerp wel het een en ander leren uit de Java-aanpak. Allereerst wil ik de schijnwerper richten op misschien wel het meest elegante aspect van Java: de interface. Deze is een hoeksteen voor de Java programmeertaal. De interface wordt vaak aangeduid met de term 'design by contract'. Waar Java doorgaans voornamelijk wordt geassocieerd met object-oriëntatie en overerving door classes van superclasses is de interface – de essentie van het zogenaamde polymorfisme van Java - misschien nog wel belangrijker. Het gaat te ver om hier het concept interface volledig te definiëren en toe te lichten, maar er zijn een paar kernbegrippen die ik wil noemen. Je kunt een interface zien als een soort ontwerp-contract. Een interface definieert methodes en member-variabelen (in PL/SQL termen: procedures, functions en globale variabelen)

> ***Java is meer dan een programmeertaal: het is een manier van denken, bijna een life-style geworden***

die door iedere class (een package is de logische tegenhanger in PL/SQL) die de interface implementeert, moeten worden aangeboden. Deze interface ontkoppelt ontwikkelaars: sommigen kunnen code gaan ontwikkelen die de interface implementeert, terwijl tegelijkertijd anderen code kunnen schrijven die gebruik maakt van de functionaliteit, die door de interface gespecificeerd is, ook al is de implementatie nog niet beschikbaar. Als de beide (groepen) ontwikkelaars klaar zijn kan hun code worden samengevoegd en zou deze een werkende applicatie moeten opleveren, ook al hebben de ontwikkelaars nooit direct met elkaar samengewerkt. Het gebruik van een interface



Figuur 1. De interface package bevat in de specification alle procedures en functions en eventueel globale constanten uit het ontwerp-contract

betekent ook dat als we een betere implementatie verkrijgen van de gewenste functionaliteit we deze zonder meer in de applicatie kunnen in-pluggen gesteld natuurlijk dat die betere implementatie de interface implementeert.

Translator

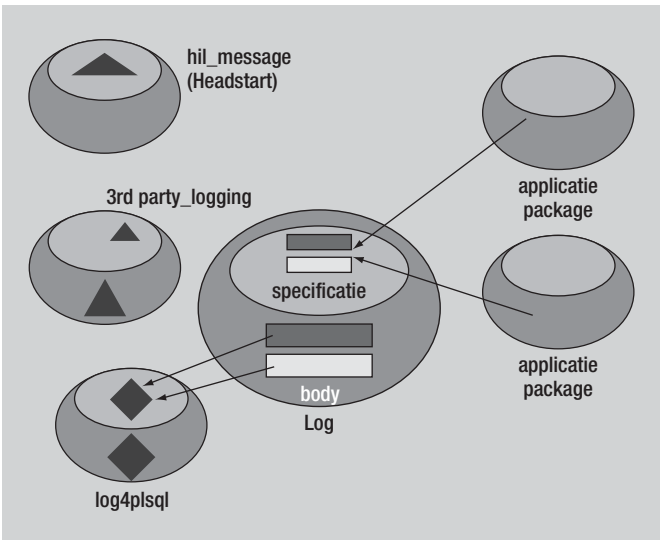
Een voorbeeld: we beschrijven een interface Translator, een pakketje functionaliteit voor het vertalen van Strings tussen verschillende talen. Deze interface specificeert de volgende methodes:

- *getTranslation()* - vertaal een woord van de brontaal in de doeltaal;
- *getTranslations()* - vertaal een array van woorden uit de brontaal in de doeltaal;

Ontwikkelaars kunnen hun code gaan schrijven in de wetenschap dat deze methodes uiteindelijk beschikbaar kunnen zijn. Anderen dragen zorg voor de uitwerking en implementatie van de interface. Misschien dat we ooit ook wel een betere implementatie van de betreffende functionaliteit kunnen vinden op internet; deze kunnen we dan in de interface verpakken en door de applicatie laten gebruiken, zonder dat we aan de applicatie iets hoeven te veranderen.
Voor de ontwikkeling van PL/SQL applicaties kunnen we net zo goed gebruikmaken van het interface-concept. We kunnen dat doen op de volgende manier (zie ook Figuur 1): creëer allereerst een interface package. Dit is een package met in de specificatie alle procedures en functions en eventueel globale constanten die onderdeel uitmaken van het ontwerp-contract. De body van deze interface package bevat uiteraard de implementatie van al deze functies en procedures, maar alles wat ze doen is de aanroep doorgeven aan de daadwerkelijke implementatie van de functionaliteit. In Figuur 2 zien we hoe de interface package specificatie twee procedures specificeert die in de body met hulp van aanroepen van procedures in twee andere packages worden gerealiseerd. Dit kan eventueel transformatie van parameters en meerdere aanroepen van verschillende procedures en functions inhouden.

Esperanto

Tijdens de ontwikkelfase kan de interface package dummy-implementaties bevatten van de procedures en functions, die statische waarden teruggeven. Hiermee kan tegen het package ontwikkeld en getest worden. In latere fasen wordt deze tijdelijke code vervangen door de echte implementatie van de interface. Als we op een later moment een 'alternatieve' implementatie zouden willen inzetten, hoeft uitsluitend de body van het interface package gewijzigd te worden. Er is geen verdere impact op de applicatie.
Terug naar het voorbeeld van de Translator Interface: creëer een package TranslatorInterface. De specification bevat de procedures en functions die we hierboven genoemd hebben. De body zal uiteindelijk gebruikmaken van de Esperanto module die we bij een externe leverancier besteld hebben. Tot die geïmplementeerd wordt, maakt de Translator gebruik van de huidige Dictionary-applicatie. Als Esperanto gebruiksklaar is, wordt de body van de TranslatorInterface package aangepast.
Het gebruik van interface-packages is waardevol voor het ontkoppelen van verschillende delen van een applicatie, vooral wanneer sommige componenten geïmplementeerd kunnen worden door externe modules of wanneer het waarschijnlijk is dat de implementatie in de loop der tijd wijzigingen zal ondergaan. Design by contract is een goede manier om een beter



Figuur 2. De interface package specificatie specificeert twee procedures, die in de body met hulp van aanroepen van procedures in twee andere packages worden gerealiseerd

gestructureerd ontwerp van PL/SQL packages te bereiken, zelfs voordat de concrete ontwikkeling gestart is.

Open source

Het best meetbare effect van de Java life-style op de wereld van PL/SQL vind je door de volgende Google-search te doen: "PL/SQL" "open source". Dat levert een flink aantal nuttige hits op. Nauw verbonden met de opkomst van Java is de zogenaamde 'open source beweging'. Natuurlijk zijn Java en open source niet equivalent maar toch bestaat er een sterk verband. Productief ontwikkelen met Java/J2EE zou niet mogelijk zijn zonder de vele open source projecten die de afgelopen jaren zijn opgedoken. Zo ongeveer sinds 2001 zien we het begin van een open source gemeenschap rondom PL/SQL: ontwikkelaars die zich in kleine, virtuele teams verenigen om via internet samen te werken aan het bedenken, ontwerpen, ontwikkelen en publiceren van gratis hulpmiddelen voor PL/SQL ontwikkelaars. SourceForce (www.sourceforge.net) is de werkplaats voor de meeste van deze projecten. In [1] (zie het kader 'Resources' aan het eind van dit artikel) vind je een goede introductie van het concept open source software – in het kort: gratis, broncode beschikbaar en toestemming om de broncode te wijzigen en te herdistribueren – en op een aantal interessante open source tools, met name voor Java overigens. Andere internetsites met meer voorbeelden van open source projecten rondom PL/SQL en de Oracle Database zijn *http://plnet.org/* en *http://www.oracle.com/technology/tech/open-source/projects.html*, met onder meer database browsers, PL/SQL code editors, alternatieven voor SQL*Plus, een tool voor generatie van Oracle Forms documentatie en zelfs een Email Server. In deze artikelenserie kijken we naar een aantal van de meest algemeen toepasbare open source projecten voor PL/SQL ontwikkelaars. Overigens, je kunt je met een paar muisklikken aansluiten bij een open source project en bijdragen aan development. En met nauwelijks meer moeite kun je op SourceForge zelf een open source project starten – waar anderen zich dan weer bij aan kunnen sluiten. Het geeft ontwikkelaars een kans zichzelf en hun werk te profileren, samen te werken aan vernieuwende technologie en anderen te helpen met nuttige hulpmiddelen.

Logging

Een van de meest bekende open source projecten onder Java ontwikkelaars is (Apache) Jakarta Log4J (zie: http://logging.apache.org/log4j/docs/index.html). Het is een eenvoudig maar krachtig hulpmiddel voor loggen, tracen en tot op zekere hoogte debuggen van Java code. Een fraaie eigenschap van Jakarta Log4J is, dat de configuratie van de logging buiten de applicatie plaatsvindt. In een externe configuratie-file geef je per applicatie-module aan wat het huidige logging-niveau is en waar de logging naar geschreven moet worden. Log-informatie kan naar verschillende uitvoerkanalen worden geschreven, zoals een file, de scherm-console, een e-mail, een database tabel etc. Logging bevat desgewenst een timestamp – datum en tijd – en de call-stack – een indicatie van de programmasectie die de logging produceert.

Het logging-framework kent verschillende log-niveaus: bij iedere log-boodschap die wordt geproduceerd binnen de applicatie geeft de ontwikkelaar aan van welk niveau die boodschap is, bijvoorbeeld *debug, informatie of error*: log.info("de log boodschap ") of log.debug("de debug boodschap ").Tijdens het runnen van de applicatie wordt geconfigureerd wat de drempel is voor log-boodschappen ofwel, wat het minimale niveau is van de daadwerkelijk te loggen boodschappen. Tijdens het ontwikkelen en testen van de applicatie gebruik je een lage logging-drempel – "log alle boodschappen" – terwijl in productie de log-drempel heel hoog wordt gesteld – "log alleen *warnings* en *errors*". Een Franse ontwikkelaar, Guillaume Moulard, heeft vanaf 2001 gewerkt aan een PL/SQL uitvoering van Log4J en noemde het Log4PLSQL (zie kader).

Log4PLSQL karakteristieken

Home Page	http://log4plsql.sourceforge.net/
Bestaat sinds	april 2002
Meeste recente release	3.1.2 (8 februari 2004)
Status	Productie
Onderliggende technologie	PL/SQL, Java
Ontwikkelteam	5
Zip-grootte	773 Kb
Gerelateerde projecten	Log4J (http://logging.apache.org/log4j), Tora (http://www.globecom.se/tora)

Installatie

De installatie van Log4plsql is zeer eenvoudig: een kwestie van minutenwerk. Log4plsql wordt geïnstalleerd in een eigen database-schema. Dit schema bevat een centraal package: PLOG. Dit package bevat procedures zoals debug(), info(), warn(), error(), fatal() om verschillende vormen van logging te schrijven. Er zijn ook functies als isDebugEnabled(), isInfoEnabled() etc; deze functies geven Boolean waarden terug en kunnen als volgt worden gebruikt:

```
BEGIN
  IF  PLOG.isDebugEnabled
  THEN
    PLOG.debug (calculationResult());
  END IF;
END;
```

Adv.
Motiv

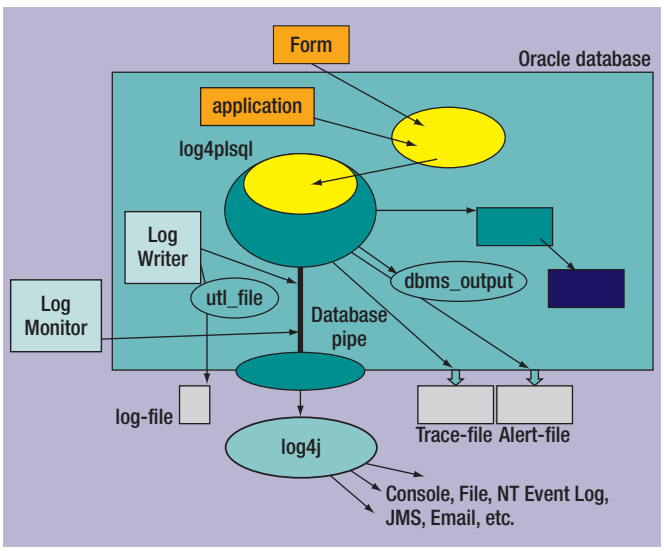
Deze constructie is beter voor de performance in het geval dat de te loggen tekst duur is om af te leiden, bijvoorbeeld wanneer calculationResult() een zware operatie is. Procedure init() wordt gebruikt om de huidige Log4plsql sessie te configureren. Configuratie omvat ondermeer het geldende logging niveau en de actieve logging-kanalen:

```
declare
    pCTX PLOG.LOG_CTX;
 begin
   pCTX := PLOG.init
            ( pLEVEL => PLOG.LDEBUG
            , pLOG4J=> TRUE -- schrijf de logging naar een database
pipe die door een aan Log4J gekoppeld achtergondproces wordt uitgelezen
(of door een ander proces, bijvoorbeeld een PL/SQL package dat via
UTL_FILE een logfile schrijft of een Oracle Forms 'console' die alle
logging toont)
            , pOUT_TRANS    => true -- schrijf logging in een autonome
transactie (en dus vastgelegd in een tabel ook als de hoofd-transactie
niet tot een commit komt)
            , pALERT        => true -- schrijf logging in de Oracle
Alert log-file
            , pTRACE        => true -- schrijf logging naar een Trace
file
            , pDBMS_OUTPUT => true -- schrijf logging naar de standard
output
            , pDBMS_PIPE_NAME => 'loggingPipeAppA -- de naam van de
database pipe waarin voor de huidige sessie de logging moet worden weg-
geschreven
            );
end;
```

Deze configuratie-instellingen worden vastgelegd in een record van het type plog.log_ctx. Dit record moet bij iedere log-actie meegegeven worden; het kan daarom het beste gedurende de sessie vastgehouden worden in een globale package variabele in de body van het logging-interface package. De beste manier om Log4plsql te configureren is door middel van een ON-LOGON trigger gedefinieerd voor het applicatieschema. Daarmee blijft de logging configuratie buiten de applicatie en kan deze eenvoudig aan- of uitgeschakeld worden.
De logging wordt naar de tabel TLOG geschreven. Deze kan zowel rechtstreeks worden benaderd als ook via de view VLOG die een netter geformatteerde uitvoer oplevert. In plaats daarvan kan Logging ook naar de standaard (dbms_)output of alert- of trace-file worden geschreven of via een database pipe beschikbaar worden gesteld aan achtergrondprocessen. Het belangrijkste achtergrondproces, meegeleverd met Log4plsql, is een direct aan log4j gekoppeld proces. Dit proces kan via de normale log4j configuratie file, logging in verschillende formaten schrijven naar ondermeer de console, files, de Windows Event Log en een database. Voor meer informatie over Log4J, zie http://logging.apache.org. Het zou ook de moeite waard kunnen zijn eens te kijken naar het Chainsaw project (http://logging.apache.org/log4j/docs/chainsaw.html), een GUI tool om de logging produces door log4j en log4plsql te analyseren. Tenslotte volgen nog enkele voorbeelden van het gebruik van Log4PLSQL:

```
-- example 1 : directe aanroep
Exec PLOG.info ('informatie');

-- example 2 : in procedure
create or replace procedure TestProc is
    cpt number;
begin
    plog.info('select statement veroorzaakt ORA-01403:No Data Found');
    select id into cpt from tlog where id = -1;
    exception
        when others then
        plog.error;  -- default message is SQLCODE SQLERRM
end;
/

exec TestProc

select * from vlog
/

LOG
------------------------------------------------------------
[18 Aug, 15:06:44.25][INFO][SCOTT][block][informatie]
[18 Aug, 15:07:26.41][INFO][SCOTT][block.SCOTT.TESTPROC][select state-
ment veroorzaakt ORA-01403:No Data Found]
[18 Aug, 15:07:26.43][ERROR][SCOTT][block.SCOTT.TESTPROC][SQLCODE:100
SQLERRM:ORA-01403: No Data Found]
```

Overeenkomstig de discussie over interfaces, zou je niet direct het PLOG package moeten aanroepen in je applicatie code. In plaats daarvan zou je een logging interface package moeten definiëren. De body van dat interface package doet aanroepen



Figuur 3. De logging wordt naar de tabel TLOG geschreven of naar de (dbms_)output, alert- of trace-file,of via een database pipe beschikbaar gesteld

Adv.
Array

naar Log4PLSQL. Logging is een goed voorbeeld van de waarde van het interface concept: veel organisaties maken al gebruik van een – of zelfs meerdere – oplossingen voor PL/SQL logging, bijvoorbeeld het hil_message package dat Oracle levert bij JHeadstart of een eigen frameworkje. Als je voor de logging gebruik zou maken van een interface package waar alle applicatie-logging naar toe gaat, zou het nu een koud kunstje zijn om van Log4PLSQL gebruik te gaan maken. Alleen de body van dat package behoeft dan nog aanpassing. Een van de waardevolle gevolgen van het gebruik van Log4PLSQL is dat het opeens de moeite waard is om logging te gaan produceren in je PL/SQL code: het kost weinig moeite en het resultaat is zeer bevredigend. De toegevoegde waarde van zelfs de meest simpele logging is direct zichtbaar, al was het maar vanwege de call-stack en timestamp die je cadeau krijgt. Om een goed beeld te hebben van de 'flow' door de applicatie is logging van grote waarde, vooral als er problemen optreden. Je zou zelfs simpele profiling en performance meting kunnen doen: log4plsql verzamelt heel veel statistieken over het aantal aanroepen van verschillende programma-onderdelen en de tijd die iedere aanroep kostte.

Resources

J2EE Open Source Toolkit, Building an Enterprise Platform with Open Source Tools, John T. Bell, Wiley, 2003
ISBN 0-471-44435-9

Voorbeelden van en achtergrondinformatie bij Log4j, AMIS Technology Corner
(http://www.amis.nl/tech_artikelen.php?id=158)

Tot slot

In volgende afleveringen van deze artikelenreeks zullen we nog kijken naar de multi-tier-architectuur, de opzet van meertalige applicatie en het gebruik van SQL binnen de applicatie. Ook zullen we nog een aantal open source tools onder de loep nemen, zoals utPLSQL, een tool voor geautomatiseerd Unittesten, ANT, een tool voor build management en PLDoc, een open source tool voor het genereren van documentatie, vergelijkbaar met JavaDoc. Tenslotte zal ik aandacht besteden aan Oracle's platform voor SQL en PL/SQL ontwikkeling (want zo is het gepositioneerd): Oracle 10g JDeveloper.

Lucas Jellema is technisch consultant bij Amis Services B.V.
(e-mail: jellema@amis.nl)

NIEUWS

Artikelen met praktische informatie, geschreven door en bestemd voor Oracle-professionals vindt u in het Online Archief van Array Publications. Vaktijdschriften als Database Magazine, Software Release en Java Magazine hebben hun artikelenarchief online gezet. Met een heldere zoekstructuur vindt u snel wat u zoekt op www.optimize.nl.

Vervolg van pagina 34.

interne processen op Oracle softwareproducten baseren, van dataverzameling uit het veld tot aan het leveren van eindproducten. De kaartgegevens zullen voortaan in het 10g spatial format geleverd worden. De overeenkomst legt de nadruk op het leveren van online kaarten en het gebruik van kaartgegevens in combinatie met Oracle softwareproducten en diensten, met name op het gebied van CRM, E-business en ERP. Juist voor deze typische bedrijfsapplicaties groeit de behoefte aan locatie-gebaseerde componenten voor het managen van de buitendienst of de toeleverketen. De volumes van geografische gegevens nemen exponentieel toe. Oracle en Tele Atlas zullen gezamenlijk de visuele weergave ervan op een gebruiksvriendelijke manier mogelijk maken.

Microsoft en de overheid: Male of bitch?

Volgens een artikel op Silicom.com heeft Microsoft af en toe problemen met het inschatten van gevoeligheden bij de overheden van buitenlandse mogendheden. Zo geeft een Spaanse, voor Latijns-Amerika bedoelde vertaling van XP bij het kiezen van het geslacht de keuze tussen male en bitch. Waarschijnlijk een grapje van een vertaler dat er niet tijdig uitgehaald is, maar dit was toch ook de Latijns-Amerikaanse macho's te gortig. Een andere fout betrof een spelletje waarin Koranverzen als achtergrondmuziek te horen was. De regering van Saoedi-Arabië was 'not amused'. In een ander spelletje werden moslims getoond die kerken in moskeeën veranderen - ook dat werd uiteraard niet gewaardeerd. Ook in Korea, Koerdistan en China zijn dergelijke blunders gemaakt, althans volgens Microsoft. (*overigens alweer een blunder, Koerdistan is geen onafhankelijke staat - red.*) Het meest onhandig was Microsoft's optreden in India. Het omstreden gebied Kashmir werd op een landkaart getoond alsof het geen deel van India uitmaakte. Dit foutje kostte miljoenen. Volgens een employee van Microsoft zijn sommige medewerkers van het bedrijf weliswaar heel slim, maar hebben ze slechts een zeer vaag beeld van de wereld buiten de V.S. Microsoft ziet zich nu genoodzaakt het personeel extra aardrijkskunde lessen te gaan geven...