

## Усовершенствованные вычисления и проверка правильности (Enhanced Calculation and Validation, by Steve Muench)

Стив Мюнч,  
член Oracle ACE

Источник: журнал Oracle Magazine #6 2007,  
<http://www.oracle.com/technology/oramag/oracle/07-nov/o67frame.html>

**Добейтесь большего в инфраструктуре разработки приложений Oracle 11g – не прибегая к программированию.**

В этой колонке я хочу еще раз бросить взгляд на предстоящий в скором будущем выпуск Oracle JDeveloper и Oracle Application Development Framework 11g (Oracle ADF - инфраструктура разработки приложений Oracle) и непосредственно поэкспериментировать с несколькими примерами, предназначенными для повышения производительности разработчика. Ниже будет показано, как просто теперь создавать вычисляемые атрибуты, подтверждать достоверность значений внешнего ключа, ограничивать значения взаимно зависимых атрибутов и определять более сложные правила проверки, не написав ни строки кода Java.

Следует удостовериться, что используется Oracle JDeveloper 11.1.1.0 Technology Preview (или более поздний выпуск), который бесплатно доступен на сайте Oracle Technology Network (<http://otn.oracle.com/products/jdev/1>). Кроме того, загрузите рабочую область стартера из файла (<http://otn.oracle.com/oramag/oracle/07-nov/o67frame.zip>)

После извлечения контента из файла o67frame.zip откройте в Oracle JDeveloper рабочую область FrameworksNovDec2007.jws. Заметим, что рабочая область стартера определяет уже знакомый набор объектов-сущностей Emp и Dept, объект отображаемого элемента EmpView и модуль прикладных программ HRModule. Затем сконфигурируйте в рабочей области стартера подключение scott, чтобы указать на базу данных, которая будет использоваться. Для этого раскройте в Application Navigator зону Application Resources, папку Connect и узел Database, чтобы воспроизвести подключение scott. Щелкните правой кнопкой мыши по подключению scott и выберите Properties..., чтобы просмотреть параметры настройки подключения базы данных. Проверьте эти параметры и замените их, в случае необходимости, чтобы они указывали на пользователя базы данных SCOTT, с которым вы будете работать. Кликните по кнопке Test Connection, чтобы убедиться, что можно

успешно соединиться с пользователем SCOTT, и затем - ОК.

Заметим, что если используется бесплатная версия Oracle Database Express Edition, то, возможно, придется создать новую учетную запись SCOTT с привилегиями CONNECT и RESOURCE и выполнить сценарий CreateDeptEmpTables.sql, хранящийся в рабочей области стартера, чтобы создать таблицы EMP и DEPT.

### Упрощение вычисляемых атрибутов

Groovy – это динамический язык на основе стандартов для платформы Java, определенный как инициирование запроса на Java-спецификацию 241 (JSR 241). Он предлагает более простой, чем у Java, синтаксис для многих часто встречающихся задач программирования, органично взаимодействует с любым классом Java, и может быть компилируемым или интерпретируемым “на лету”. В Oracle ADF 11g обеспечивается обширная поддержка языка Groovy, и первый пример в этой колонке демонстрирует, как использовать выражения Groovy для определения вычисляемых атрибутов.

В упоминавшейся выше рабочей области стартера объект-сущность Emp уже имеет определенный переходный атрибут, названный TotalComp. Давайте обновим определение этого атрибута, чтобы он стал равен сумме зарплаты служащего и его комиссионного вознаграждения (которые определены атрибутами Sal и Comm, соответственно). В формуле попробуем учесть тот факт, что Comm и Sal могут быть пустыми.

Находясь в Oracle JDeveloper, дважды кликните по объекту Emp в Application Navigator, чтобы открыть редактор Entity Object Editor. Кликните по Attributes, чтобы перейти на страницу Attributes, и дважды кликните по строке в таблице, содержащей атрибут TotalComp. В диалоговом окне Attribute Editor убедитесь, что в группе переключателей Value Type выбран переключатель Expression. В поле Value введите следующую формулу:

$$(Sal!=null?Sal:0)+(Comm!=null?Comm:0)$$

В этой формуле используется трёхместный оператор, который проверяет булево условие ( $Sal! =null$ ), чтобы вернуть значение  $Sal$ , если это не пустой указатель, и ноль в противном случае. Затем она (формула) выполняет аналогичные вычисления для значения  $Comm$  и возвращает сумму обоих вычисленных значений.

Чтобы закончить назначение атрибута, перейдите в Attribute Editor на страницу Dependencies, выберите атрибут  $Sal$  в списке Available и кликните по кнопке Add (стрелка вправо), чтобы переместить его в список Selected. Выполните те же самые шаги, чтобы также добавить к списку Selected атрибут  $Comm$ . Наконец, кликните по ОК, чтобы сохранить изменения. Чтобы протестировать изменения, кликните правой кнопкой мыши по модулю прикладных программ HRModule в Application Navigator и выберите из появившегося меню Run. Когда появится диалоговое окно Business Components Browser - Connect, кликните по Connect. Дважды кликните по экземпляру объекта отображаемого элемента Employees и измените комиссионное вознаграждение и/или значение зарплаты в любой строке, чтобы отметить, что актуальность полного вознаграждения обеспечивается всегда.

### Проверка достоверности внешних ключей

В предыдущей посвященной инфраструктурам колонке (сентябрь/октябрь 2007 г.) шла речь о том, как определить декларативные списки значений (LOV) для атрибута объекта отображаемого элемента, чтобы помочь пользователям, выбирающим поисковые значения для существующих внешних ключей. Имейте в виду, что хотя эти LOV удобны для конечных пользователей, они не являются заменой настоящей проверки правильности внешнего ключа на уровне объекта-сущности. Например, некоторые компоненты UI, скажем, текстовые поля со всплывающими LOV, позволяют пользователю непосредственно вводить значение внешнего ключа. Кроме того, в сервис-ориентированной архитектуре внешние ключи в объектах-сущностях могут быть программно установлены другим приложением при помощи интерфейса Web-сервисов. К счастью, новое правило проверки Key Exists облегчает верификацию атрибутов внешнего ключа, быстро делая ту работу, которая раньше была обычной рутинной работой программирования.

В следующем примере правило верификации Key Exists добавляется к объекту-сущности Emp в Model project. На странице General редактора Entity Object Editor кликните кнопкой Add Validation Rule по зеленому знаку “плюс” справа от заголовка страницы Validation Rules (на мониторах меньшего размера вам, возможно, придется пролистать вниз, чтобы увидеть этот раздел). В диалоговом окне Add Validation Rule выберите Key Exists

Validator из списка Rule Types. Находясь в закладке Rule Definition, выберите WorksInDeptAssoc из списка Association Name. Этот выбор обозначает ассоциацию “один ко многим” между Dept и объектом-сущностью Emp, которая представляет подлежащее верификации отношение внешнего ключа. Затем в закладке Failure Handling введите в поле Message Text сообщение об ошибке Department not exist. И, наконец, кликните по ОК, чтобы определить новое правило проверки.

Выполните снова HRModule и измените значение идентификатора отдела для существующего служащего на любое недопустимое двузначное число (скажем, 99). Когда вы зафиксируете транзакцию или просто переместитесь к другой строке, будет возбуждена исключительная ситуация с заказанным вами сообщением об ошибке.

### Ограничение зависимых значений

Другой распространенный вид проверки правильности (верификации) заключается в сравнении значений двух атрибутов одной и той же строки. В следующем примере вводится принудительное исполнение правила, в котором говорится, что комиссионное вознаграждение служащего должно быть меньше, чем его зарплата. Это правило применимо только в том случае, когда и комиссионное вознаграждение, и зарплата имеют непустое (non-null) значение, и оно должно заново вычисляться после каждого изменения комиссионного вознаграждения или зарплаты. Усовершенствованное правило верификации Compare в Oracle JDeveloper 11g делает реализацию этой проверки очень простой.

На странице General редактора Entity Object Editor для Emp перейдите в раздел Validation Rules и снова щелкните кнопкой Add Validation Rule. В диалоговом окне Add Validation Rule выберите Compare Validator из списка Rule Types. В закладке Rule Definition выберите Comm из списка Attribute и Less Than из списка Operator. Из списка Compare With выберите Entity Attribute и выберите атрибут Sal в размещенном ниже поле Select Entity Attribute. Эти шаги настраивают основное сравнение в правиле верификации. Затем в закладке Validation Execution введите формулу  $Sal! =null \ \&\& \ Comm! =null$  в поле Conditional Execution Expression. Это поле становится причиной того, что правило применяется только в том случае, если истинно указанное в этом поле условие. Обратите внимание, что при вычислении истинности выражения учитывается регистр, так что убедитесь, что напечатано Comm, а не comm. Затем перейдите в раздел Triggering Attributes, выберите Sal из списка Available Attributes и кликните по кнопке Add (стрелка вправо), чтобы переместить Sal в список Selected Attribute, а затем проделайте то же самое для атрибута Comm. Если во время выполнения значение любого из этих атрибутов изменится, значение

правила будет вычислено заново.

Наконец, перейдите в закладку Failure Handling и введите в поле Message Text следующее сообщение об ошибке проверки достоверности:

*The {attr1} of {val} must be less than the {attr2}.*

Убедитесь, что три выражения с сообщениями заключены в фигурные скобки. В находящейся ниже таблице Error Message Expressions кликните по строке для лексемы attr1 и дважды кликните по ячейке Expression в этой строке. Введите выражение source.hints.Comm.label, чтобы сослаться на значение удобной для пользователя метки показа для атрибута Comm исходного объекта-сущности. Точно так же введите выражение source.hints.Sal.label для лексемы сообщения attr2 и введите выражение Comm для лексемы val, чтобы сослаться на значение атрибута Comm. Эти выражения с сообщениями, так же, как и те, которые использовались в приводившемся выше примере с вычисляемыми атрибутами, используют синтаксис языка Groovy. Хотя это очень простые выражения, важно понять, что в случае необходимости разработчики могут использовать всю мощь Groovy. Для завершения кликните по ОК, чтобы определить новое правило.

Снова выполните HRModule, чтобы протестировать правило. Попробуйте ввести для комиссионного вознаграждения существующего служащего значение 13000. Когда вы зафиксируете транзакцию, должно появиться параметризованное сообщение об ошибке “Комиссионное вознаграждение 13 000 должно быть меньше зарплаты”. Если вы измените только значение зарплаты служащего (не изменяя комиссионного вознаграждения), можно проверить, что не будет никакого исключительного состояния, потому что значение Comm – пустое. Наконец, изменяя зарплату служащего до меньшего значения, чем имеющееся комиссионное вознаграждение этого человека, можно проверить, что инициирование атрибутов срабатывает должным образом.

### Написание правил верификации на Groovy

В заключительном примере я создам более сложное правило верификации, используя язык Groovy. Это правило определяет, что если служащий работает в отделе, название которого заканчивается буквой s, то его зарплата должна быть кратной числу 5. Это правило требует условной логики и также обращается к атрибуту связанного объекта-сущности Dept.

На странице General редактора Entity Object Editor для Emp перейдите в раздел Validation Rules и снова кликните кнопкой Add Validation Rule. В диалоговом окне Add Validation Rule кликните по списку Rule Types, прокрутите картинку “до упора” вниз и выберите Expression Validator. В закладке Rule Definition введите следующее выражение:

```
if (Dept?.Dname?.toUpperCase()?.endsWith("S")
&& Sal % 5 != 0) {
    return false;
}
return true;
```

Булево выражение в условном операторе ссылается на свойство Dept подвергаемого верификации текущего объекта Emp, чтобы обратиться к связанному объекту-сущности Dept (если такой существует). Затем оно ссылается на атрибут Dname этого объекта Dept, преобразовывает его в верхний регистр, чтобы выполнить сравнение без учета регистра, и использует метод endsWith() класса String, чтобы проверить, заканчивается ли значение Dname буквой s. Заметим, что вместо написания Dept.Dname.toUpperCase().endsWith(), я заменил нормальный точечный оператор оператором безопасной навигации Groovy “?.”.

Этот оператор работает как точечный оператор, делая возможной навигацию от самого объекта к его свойствам. Однако, если стоящее с левой стороны значение является пустым, оператор Groovy не генерирует исключительную ситуацию NullPointerException, а вместо этого возвращает пустое значение. Удобно, что пустое значение в Groovy рассматривается как булево значение FALSE, так что использование оператора ?. может сделать многие выражения более компактными. После этой начальной проверки приведенное выше выражение использует целочисленный оператор деления по модулю (%), чтобы проверить, является ли зарплата числом, кратным 5. Если названия отдела заканчивается буквой s, а зарплата не кратна числу 5, то этот оператор возвращает FALSE, что приводит к неудачному вычислению правила. В противном случае, будет возвращено значение TRUE.

Перейдем в закладку Failure Handling. Заметим, что поскольку правила верификации Groovy могут условно (то есть, при выполнении некоторого условия) генерировать одну или несколько исключительных ситуаций, в этой закладке можно добавить несколько сообщений об ошибках. Выбранное сообщение будет использоваться в том случае, если правило верификации возвращает ложь (FALSE). Чтобы добавить сообщение, кликните по кнопке Add Message (обозначенной зеленым знаком “плюс”). Когда появится диалоговое окно Select Text Resource, кликните по New.... В диалоговом окне Create Text Resource введите в поле Key значение MultipleOfFiveSalForDeptsEndingInS-Message и введите в поле Value следующее сообщение об ошибке:

*If department name ends in S the salary must be multiple of five.*

Затем кликните по Save и Select, и, наконец, кликните по ОК, чтобы определить новое правило верификации.

Снова выполните HRModule и попытайтесь ввести для служащего отдела 30 (SALES) или 40 (OPERATIONS) зарплату 2001. Затем зафиксируйте изменения. Поскольку

названия этих отделов заканчиваются буквой s и зарплата служащего не кратна 5, приложение сгенерирует ошибку верификации.

Хотелось бы надеяться, что эти простые примеры помогут вам понять некоторые из новых декларативных возможностей разработки, которые станут доступными в следующем основном выпуске Oracle JDeveloper и Oracle ADF. Для получения дополнительной информации об этом новом выпуске отсылаем к ресурсам, содержащимся на сайте <http://otn.oracle.com/products/jdev/11>. Если вы захотите узнать больше о языке Groovy, посетите сайт <http://groovy.codehaus.org>.

**Steve Muench** (Стив Мюнч) - менеджер-консультант по продукции для Oracle JDeveloper и член команды Oracle ACE (). За 17 лет сотрудничества с корпорацией Oracle он занимался разработкой и поддержкой инструментальных средств Oracle и технологий XML, а в настоящее время продолжает их пропагандировать. Мюнч был соавтором документа Oracle ADF Developer's Guide for Forms/4GL Developers (Oracle, 2006) и написал книгу Building Oracle XML Applications (O'Reilly Media, 2000). На сайте Oracle Technology Network <http://www.oracle.com/technology> и в своем блоге Dive into ADF blog <http://radio.weblogs.com/0118231> он делится со всеми подсказками и "хитрыми" приемами программирования.

#### Следующие шаги

Читайте еще Frameworks (<http://www.oracle.com/technology/oramag/oracle/frameworks>)

Еще об Oracle JDeveloper 10g и Oracle ADF

<http://otn.oracle.com/products/jdev>

<http://otn.oracle.com/products/jdev/tips>

[/muench/designpatterns](http://otn.oracle.com/products/jdev/tips/muench/designpatterns)

Загрузите

Oracle SOA Suite (<http://www.oracle.com/technology/software/tech/soa>)

Oracle JDeveloper 11g Technology Preview

(<http://www.oracle.com/technology/products/jdev/11>)

starter workspace for this column

(<http://www.oracle.com/technology/oramag/oracle/07-nov/o67frame.zip>)