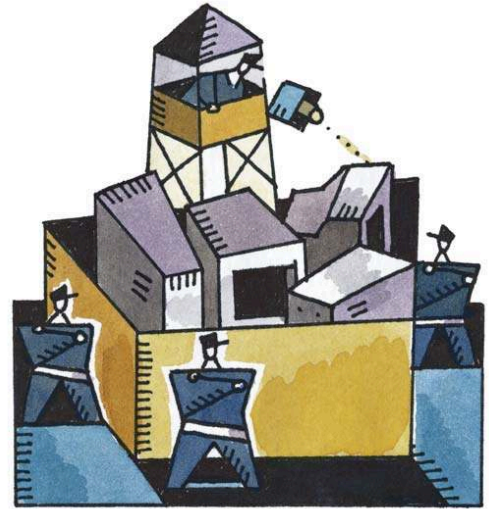


Oracle Application Development Framework 11g 강화된 연산과 검증

저자 Steve Muench



본 기사에서는 곧 출시 예정인 Oracle Jdeveloper와 Oracle Application Development Framework(Oracle ADF) 11g의 개발자 생산성 개선을 위한 몇 가지 예제를 직접 진행해 보도록 하겠다. 연산 속성 생성, 외부 키 값 검증, 상호 의존적 속성 값 제한, 복잡한 검증 규칙 정의가 Java 코드를 작성할 필요 없이 어떻게 쉽게 이루어지는 지를 살펴 보겠다.

Oracle Technology Network(otn.oracle.com/products/jdev/11)에서 무료 다운로드 가능한 Oracle JDeveloper 11.1.1.0(또는 상위 버전) Technology Preview를 사용해야 본 기사의 내용에 따라 진행할 수 있다. 스타터 워크스페이스는 otn.oracle.com/oramag/oracle/07-nov/067frame.zip에서 다운로드 받을 수 있다.

Oracle Application Development Framework 11g, 코드는 없어도
지고 기능은 확대된다.

o67frame.zip 파일의 내용을 추출한 후 Oracle Jdeveloper의 FrameworksNovDec2007.jws 워크스페이스를 열어 시작한다. 스타터 워크스페이스는 친숙한 Emp 및 Dept 엔터티 오브젝트, EmpView 뷰 오브젝트, HRModule 애플리케이션 모듈을 생성한다는 사실에 주의해 본다. 다음으로, 사용할 데이터베이스를 지정해서 스타터 워크스페이스에 scott 연결을 구성한다. 이를 위해서 Application Navigator, Connection 폴더, Database 노드의 Application Resources 구역을 확장하여 scott 연결을 나타낸다. scott 연결의 오른쪽을 클릭해 Properties를 선택하면 데이터베이스 연결 설정을 확인할 수 있다. 이들 설정을 확인한 후 필요한 경우 변경해 작업에 사용할 SCOTT 데이터베이스 사용자를 지정한다. Test Connection 버튼을 누르면 SCOTT 사용자에게 연결되며, OK를 클릭한다.

무료 Oracle Database Express Edition을 사용하고 있는 경우에는 CONNECT와 RESOURCE 권한이 있는 새로운 SCOTT 사용자 계정을 생성해서 스타터 워크스페이스에서 제공하는 CreateDeptEmpTables.sql

스크립트를 통해 DEPT와 EMP 테이블을 생성한다.

연산 속성 간소화

Groovy는 Java Specification Request 241(JSR 241)로 정의된 Java 플랫폼 용 표준 기반 다이내믹 언어다. 많은 공통 프로그래밍 작업에서 Java에 비해 구문이 간단하고, Java 클래스와 원활하게 호환되며, 작동 상태에서 컴파일링 및 해석이 모두 가능하다. Oracle ADF 11g은 Groovy 언어에 대한 광범위한 지원을 제공하며, 본 기사의 첫 번째 예제에서 Groovy 익스프레션을 사용해 어떻게 연산 속성을 정의하는 지를 보여 줄 것이다.

앞에 언급한 스타터 워크스페이스에서 Emp 엔터티 오브젝트는 이미 TotalComp라고 불리는 정의된 임시 속성을 가지고 있다. 이 속성 정의를 업데이트해서 직원의 임금과 커미션의 합이 되도록 한다(Sal과 Comm 속성으로 각각 정의). 공식에서 Comm과 Sal의 값이 null이 될 수도 있다.

Oracle Jdeveloper에서, Application Navigator에 있는 Emp 엔터티를 더블 클릭하면 Entity Object Editor를 열 수 있다. Attributes를 클릭하면 Attributes 페이지로 이동한다. 여기에서 TotalComp 속성에 포

용자 친숙 디스플레이 레이블 값을 레퍼런스한다. 마찬가지로, attr2 메시지 토큰을 위해 source.hints.Sal.Label 익스프레션을 입력한 후 val 토큰의 Comm 익스프레션을 입력해 Comm 속성 값을 레퍼런스한다. 이들 익스프레션 메시지는 앞서의 연산 속성 예제에서와 마찬가지로, Groovy 구문을 사용한다. 이들이 매우 간단한 익스프레션이기는 하지만, 개발자가 필요한 경우에 Groovy의 전체 권한을 활용할 수 있음을 이해할 필요가 있다. OK를 눌러 새로운 규칙을 정의하면 모두 마치게 된다.

HRModule을 다시 작동하면 규칙을 테스트할 수 있다. 기존 직원의 커미션으로 13000 값을 입력해 보자. 실행하면 “The Commission of 13,000 must be less than the Salary” 라는 파라미터화 된 오류 메시지가 나타날 것이다. 커미션 없이 직원의 임금 값을 변경하려는 경우에는 Comm 값이 null이기 때문에 예외가 일어나지 않음을 확인할 수 있다. 마지막으로, 직원의 임금을 해당 직원의 기존 커미션보다 낮게 변경해서 트리거 속성이 제대로 작동하는지 확인할 수 있다.

Groovy 검증 규칙 작성

마지막 예에서는 Groovy 언어를 사용해 좀더 복잡한 검증 규칙을 생성해 보겠다. 이 규칙은 어떤 직원이 s자로 끝나는 이름의 부서 소속이라면, 그의 임금이 5의 배수여야 한다는 것을 규정한다. 이 규칙은 조건 논리를 요구하며, 관련 Dept 엔터티 오브젝트의 속성에 접근한다.

Emp의 Entity Object Editor의 General 페이지에서 Validation Rules 항목으로 이동한 후 Add Validation Rule 버튼을 다시 클릭한다. Add Validation Rule 대화상자에서 Rule Types 목록을 클릭해서 아래로 내려 간 후, Expression Validator를 선택한다. Rule Definition 탭에서 다음과 같은 익스프레션을 입력한다.

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

‘if’ 구문의 Boolean 익스프레션이 현재 검증되고 있는 Emp 오브젝트의 Dept 속성을 레퍼런스해 관련 Dept 엔터티 오브젝트에(존재하는 경우) 접근한다. 그 후, Dept 오브젝트의 Dname 속성을 레퍼런스하고, 이를 대문자로 전환해 대소문자 비구분(case-insensitive) 비교를 실시한 후 endsWith()의 String 클래스를 사용해 Dname 값이 s로 끝나는지를 테스트한다. Dept.Dname.toUpperCase().endsWith()를 작성하는 대신에, 일반적인 dot operator를 Groovy의 ‘?’ safe-navigation operator로 대체했다. 이 operator는 dot operator처럼 작동해 하나의 오브젝트에서 해당 오브젝트 속성으로의 이동을 가능케 한다. 그러나 왼쪽 값이 null이면, Groovy operator가 NullPointerException을 버리는 대신에

null로 평가한다. 편리하게도, null이 Boolean 값으로 Groovy에 존재하면, false로 평가되기 때문에 ?. operator를 사용해 많은 익스프레션을 컴팩트하게 만들 수 있다. 이 초기 확인이 끝나면, 상기의 익스프레션이 integer modulo operator(%)를 사용해 임금이 5의 배수인지 여부를 테스트한다. 부서 이름이 s로 끝나고 임금이 5의 배수가 아니면, false를 리턴함으로써, 규칙은 실패한다. 그렇지 않은 경우에는 true를 리턴한다.

다음으로, Failure Handling 탭으로 이동한다. Groovy 검증 규칙은 몇 가지 예외적인 조건을 야기할 수 있기 때문에, 이 탭에 여러 개의 오류 메시지를 추가할 수 있다. 선택하는 메시지는 검증 메시지가 false를 리턴할 때 사용된다. 메시지 추가를 위해서는 Add Message 버튼(녹색 십자 표시)을 클릭한다. Select Text Resource 대화상자가 나타나면, New.... In the Create Text Resource 대화상자에서 New...를 클릭한 후 Key 필드에 MultipleOfFiveSalForDeptsEndingInSMMessage를 입력하고 Value 필드에 다음과 같은 오류 메시지를 입력한다.

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

Save와 Select를 클릭하고, 마지막으로 OK를 누르면 새로운 검증 규칙을 정의하게 된다.

다시 HRModule을 작동해, 30(SALES)이나 40(OPERATIONS) 부서 직원의 2001년 임금을 입력한다. 그리고 나서 변경을 적용한다. 이들 부서 명칭이 s로 끝나고 직원의 임금이 5의 배수가 아니기 때문에 애플리케이션이 검증 오류를 야기한다.

이들 간단한 예제가 차기 Oracle Jdeveloper 및 Oracle ADF에서 제공될 새로운 선언적 개발 기능을 이해하는데 도움이 되었기를 바란다. 신제품 출시에 대한 자세한 내용은 otn.oracle.com/products/jdev/11의 리소스를 확인하기 바란다. Groovy 언어에 대한 자세한 정보는 <http://groovy.codehaus.org>에서 확인할 수 있다. ●

Steve Muench는 1990년부터 오라클에서 근무해 왔으며, 현재는 Oracle Jdeveloper와 Oracle ACE의 제품 컨설팅 매니저로 일하고 있다. Muench는 오라클 톨과 XML 기술을 개발하고 지원해 왔으며, 이들 첨단 기능의 확산에 앞장서고 있다. Oracle ADF Developer's Guide for Forms/4GL Developers(Oracle, 2006)의 공동 저자이며, Building Oracle XML Applications(O'Reilly Media, 2000)을 집필하기도 했다. Muench가 제공하는 정보는 OTN 및 Dive into ADF blog를 참조하자.