TOOLS

**In mijn laatste project werd een model gebruikt waaruit op verschillende momenten allerlei waarden moesten worden uitgelezen. Het bleek erg handig te zijn om te kunnen zeggen: "Ik wil uit de object tree het object met id=X hebben en daarvan het attribuut A". Ik had dus een querytaal nodig om de object tree uit te vragen. Even googlen bracht mij bij de Jakarta Commons library JXPath.**

# Java object query's met JXPath

## Jakarta Commons component is krachtige tool

JXPath is een Commons component waarmee Java-objectstructuren kunnen worden bevraagd met de XML XPath query taal. XPath is gespecialiseerd in het aflopen van hiërarchische elementstructuren en past daarom goed op een Java object tree. Ik heb JXPath op vele plaatsen in het project ingezet en het is een erg krachtige tool gebleken. Het is echter ook een wat minder bekende Commons-component en daardoor is er maar weinig documentatie over te vinden op het internet, zeker over de meer geavanceerde mogelijkheden. Mijn eigen projectervaringen heb ik omgezet in een uitgebreide tutorial die te vinden is op mijn website (zie kader 'Verdere informatie'). Dit artikel is een verkorte versie daarvan.

### Voorbeeldmodel

Uiteraard is voor de voorbeelden in dit artikel een voorbeeldapplicatie nodig: een *bedrijf* met *afdelingen* met *werknemers*. Het classmodel ziet eruit als in afbeelding 1. Aanvullende code is te vinden op de Java Magazine website. In tabel 1 wordt een kort overzicht gegeven van de 'vulling' van de object tree in dit artikel:

| Company | Department | Employee (name, jobtitle, age) |
|---|---|---|
| Acme Inc. | Sales | Johnny, Sales rep, 45 |
| | | Sarah, Sales rep, 33 |
| | | Magda, Office assistant, 27 |
| | Accounting | Steve, Head Controller, 51 |
| | | Peter, Assistant Controller, 31 |
| | | Susan, Office assistant, 27 |

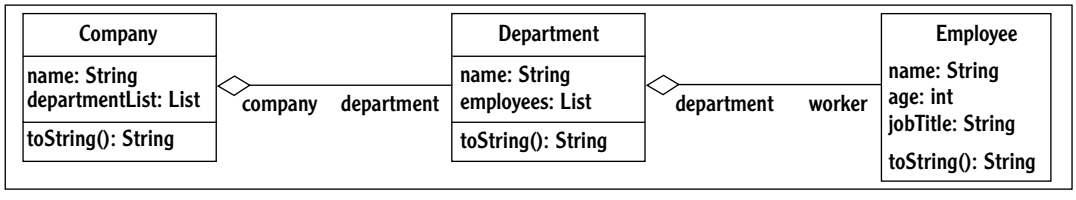
Tabel 1. Een kort overzicht van de 'vulling' van de object tree.

### Eenvoudige JXPath query's

De meest eenvoudige query is die om een enkel object uit de object tree op te halen. Om bijvoorbeeld een company-object op te halen met JXPath voer je de volgende code uit:

```
JXPathContext context = JXPathContext.
newContext(company);
        Company c = (Company)context.getVa-
lue(".");
```

De eerste regel toont het aanmaken van een context. Deze context is het startpunt voor alle XPath-query's in de object tree (vergelijk de 'root-node' in een XML document). De tweede regel voert de query uit. Omdat het startpunt van de context het `company` object is en we het `company`




Afbeelding 1. Classmodel van de voorbeeldapplicatie

**Bart van Riel**
is werkzaam bij Capgemini.

object willen ophalen, bestaat de query uit de current-element-selector '.'.

## Gebruik van predicates

Een `employee` is een kind-object van een `department`. Om de `employee` op te halen met de naam 'Johnny' gebruik je de volgende code (`company` is nog steeds het startpunt van de context)

```
Employee emp = (Employee)context.getValue("/departmentList/employees[name='Johnny']");
```

Lees dit als: 'Zoek vanaf het begin in alle `departments` naar de `employee` waarvan het name attribuut de waarde '`Johnny`' heeft'. Het voorgaande codevoorbeeld is een voorbeeld van het gebruik van een 'predicate' om op bepaalde waarden te zoeken. Het gebruik is te vergelijken met de WHERE clause in SQL. Het is ook mogelijk om predicates met elkaar te combineren:

```
Employee emp = (Employee)context.getValue(
    "/departmentList/employees[name='Susan' and age=27]");
```

Het is vaak niet praktisch om volledig uitgeschreven query's toe te passen. Vaak moet dezelfde opzoekactie verscheidene malen worden uitgevoerd, steeds voor verschillende waarden. JXPath biedt daarom de mogelijkheid om *variabelen* te gebruiken in XPath-query's. Het bovenstaande voorbeeld zou er met gebruik van variabelen als volgt uitzien:

```
 context.getVariables().declareVariable("name", "Susan");
context.getVariables().declareVariable("age", new Integer(27));
Employee emp = (Employee)context.getValue(
    "/departmentList/employees[name=$name and age=$age]");
```

## Itereren over collecties

JXPath biedt de mogelijkheid om een Iterator op te vragen over alle gevonden objecten als resultaat van een XPath-query. Het volgende fragment laat zien hoe over alle `departments` kan worden geïtereerd:

```
for(Iterator iter = context.iterate("/departmentList"); iter.hasNext();){
  Department d = (Department)iter.next();
  //...
}
```

Om alle `employees` van alle afdelingen op te halen:

```
for(Iterator iter =
     context.iterate("/departmentList/employees"); iter.hasNext();){
  Employee emp = (Employee)iter.next();
  //...
}
```

Om alle `employees` ouder dan van de afdeling 'Sales' op te halen:

```
for(Iterator iter = context.iterate(
     "/departmentList[name='Sales']/employees[age>30]");
       iter.hasNext();){
     Employee emp = (Employee)iter.next();
     //...
}
```

Een alternatief met parameters:

```
context.getVariables().declareVariable("deptName", "Sales");
context.getVariables().declareVariable("minAge", new Integer(30));
for(Iterator iter =  context.iterate(
     "/departmentList[name=$deptName]/employees[age>$minAge]");
       iter.hasNext();){
    Employee emp = (Employee)iter.next();
    //...
}
```

De laatste twee fragmenten laten zien hoe verscheidene predicates binnen één XPath-query kunnen worden gecombineerd.

## Pointers

Een Pointer is een JXPath-object dat een verwijzing vertegenwoordigt naar de locatie van een object in de object tree. Een Pointer verwijst bijvoorbeeld naar 'de eerste employee van het tweede department'. Ten opzichte van direct uitgevraagde objecten bieden Pointers extra mogelijkheden, zoals het uitvoeren van *relative query's* via *relative contexts*, waarover verderop meer.

## Werken met Pointers

Het verkrijgen van een Pointer naar een object in de object tree is vrijwel identiek aan het direct opvragen van een object uit de tree:

```
JXPathContext context = JXPathContext.newContext(company);
Pointer empPtr =       context.getPointer("/departmentList[name='Sales']/
employees[age>40]");

System.out.println(empPtr);
//output: /departmentList[1]/employees[1]

System.out.println(((Employee)empPtr.getValue()).getName());
//output: Johnny
```

Let op de output van de Pointer, hieruit blijkt dat een Pointer een locatie beschrijft en geen object. Het object dat zich op die locatie in de object tree bevindt, is op te halen met de `getValue()` method van Pointer.

Over Pointers kan ook worden geïtereerd en wel als volgt:

```
for(Iterator iter = context.iteratePointers(
   "/departmentList[name='Sales']/employees[age>30]");
      iter.hasNext();){
   Pointer empPtr = (Pointer)iter.next();
   //...
}
```

## Relative Context en Relative Queries

Omdat een Pointer een verwijzing beschrijft, kan deze als startpunt worden gebruikt om de gehele object tree te bereiken. Een Pointer kan worden gebruikt als 'root object' voor een *relative context*, van waaruit door middel van *relative queries* de gehele object tree kan worden uitgevraagd. Deze flexibiliteit biedt enorme mogelijkheden, zoals de volgende voorbeelden laten zien. Op de volgende wijze wordt een relative context gemaakt:

```
for(Iterator iter =  context.iteratePointers("/departmentList[name='Sales']/employees[age>30]");
   iter.hasNext();){
     Pointer empPtr = (Pointer)iter.next();
     JXPathContext relativeContext = context.getRelativeContext(empPtr);
}
```

Binnen elke iteratie wordt dus een relative context gemaakt met als startpunt een specifieke `employee`. Vervolgens kunnen XPath query's worden uitgevoerd op de gehele object tree, zoals de volgende fragmenten laten zien:

```
//Huidige employee
Employee emp = (Employee)relativeContext.
getValue(".");

//Naam van Employee
String name = (String)relativeContext.
getValue("./name");

//Naam van Department van deze Employee
String deptName = (String)relativeContext.
getValue("../name");

//Naam van Company van deze Employee
String compName = (String)relativeContext.
getValue("../../name");

//Alle direct collega's van deze Employee
for(Iterator empIter = relativeContext.iterate("../
employees"); empIter.hasNext();){
   Employee colleague = (Employee)empIter.next();
}
```

## Conclusie

JXPath is een uitermate handige tool om object trees af te lopen en de objecten in de tree uit te vragen. Omdat de XPath-querytaal wordt gebruikt, is er voldoende referentiemateriaal om zelfs in de meest complexe situaties op efficiënte wijze het applicatiemodel te benaderen. De mogelijkheid van het gebruik van relative contexts maakt het tool nog flexibeler. Dit artikel kan slechts een algemene indruk geven van de toepassingsmogelijkheden van JXPath. Voor een uitgebreidere beschrijving met voorbeelden wordt verwezen naar de codevoorbeelden bij dit artikel en de uitgebreide tutorial op mijn website.

**Verdere informatie**

- De code voor dit artikel is te vinden op de website van Java Magazine.
- Uitgebreide JXPath tutorial op: www.tutorials.tfo-eservices.eu
- Homepage JXPath: jakarta.apache.org/commons/jxpath
- XPath informatie: www.w3schools.com/xpath