Het doorlopen van de Tree:TreeIterator

De interface TreeIterator

Veel van de gewenste functionaliteit wordt gegenereerd tijdens het doorlopen van de nodes. Dat zagen we ook in het vorige hoofdstuk: van een werknemer was alleen de manager vastgelegd in de database: alle overige informatie wordt tijdens het uitvoeren van een query afgeleid (zoals het level, en het pad).

Voor het doorlopen van elementen in een collectie in het algemeen bestaat in Java al de interface java.util.Iterator.
Voor het doorlopen van boomstructuren breiden we daarop uit.

Maak de interface TreeIterator als een subinterface van Iterator.
Ook hierin wordt met een generic type werkt: een TreeIterator werkt met een subtype van Node. Het begin van de code komt er dan als volgt uit te zien:

public interface TreeIterator**<E extends Node<E>>** extends Iterator**<E>**{...}

Wat staat hier precies? Bij TreeIterator introduceren we het type E met de eigenschappen daarvan. E moet een subtype van Node zijn, en omdat Node zelf ook met een generic type werkt, komt <E> ook achter Node te staan. Datzelfde type E is ook het type waar de superinterface Iterator mee werkt (namelijk het type dat next() ophaalt). Dus: extends Iterator**<E>**

In de interface TreeIterator hebben we in ieder geval de volgende methoden nodig:

- int level()
- void startWith(E node)
- boolean isLeaf()

Later zullen we deze interface nog verder uitbreiden, met name met een path() methode. Deze is wat ingewikkelder: er is informatie nodig uit een sub-type van Node om het pad op te kunnen bouwen. We komen hier later op terug.