

definícií typov či už v zodpovedajúcom DTD (Document Type definition), alebo v schéme.

Vytvorenie pravidiel

Okrem možnosti priameho označenia prvkov na zabezpečenie prekladu atribútom *its:translate* existuje aj možnosť vytvorenia pravidiel. Slúžia na to prvky *its:rules* a *its:translateRule*, napríklad začlenením kódu do dokumentu:

```
<its:rules version="1.0">
  <its:translateRule selector="//div" translate="yes"/>
  <its:translateRule selector="//span" translate="no"/>
</its:rules>
```

predpíšeme vykonať preklad všetkých prvkov *div* a zachovať pôvodný text prvkov *span*. Hodnota atribútu *selector* je výrazom v jazyku *XPath*. Je teda možnosť vytvárať nielen jednoduché predpisy ako v uvedenom príklade, ale vyjadriť aj zložitejšie závislosti.

Okrem možnosti priameho vloženia pravidiel do dokumentu možno vložiť do dokumentu pravidlá z externého súboru. Napríklad ak pravidlá budú zapísané v súbore *pravida.xml*, ich použitie predpíšeme zápisom:

```
<its:rules version="1.0" xlink:href="pravida.xml" xlink:type="simple"/>
```

Treba poznamenať, že pri tvorbe a uplatnení pravidiel možno uvažovať s uplatnením priorít, ako aj s uplatnením dedičnosti. Ich mechanizmus je podobný ako pri definovaní a uplatnení tabuliek kaskádových štýlov.

Záver

Poznanie štandardu a zovšeobecnenia najlepších praktík internacionalizácie môže byť pre tvorcov nových programových produktov a dokumentov prospešné. Odbremení ich od potreby hľadania pravidiel internacionalizácie pre každý vyvíjaný produkt. Umožní im pri viacerých produktoch použiť rovnaké nástroje na lokalizáciu produktu a minimalizovať tak náklady spojené s internacionalizáciou a lokalizáciou.

■ IMRICH BURANSKÝ
iburansky@infoware.sk

Caché 5.1 - Prvé riadky kódu II.

Tentoraz si ukážeme druhý scenár aplikácie z predošlej časti.

Na pripomenutie: Tvorili sme Java projekciu perzistentných tried Caché a nad výslednými proxy Java triedami sme robili operácie projekcie z triedy s aplikačnou logikou, vytvorenou v Caché.

Použijeme teda rovnaké perzistentné triedy ako minule, a to vrátane ich projekcie do Javy. No aplikačnú logiku budeme plne implementovať na strane Javy. Na to musíme najskôr vystaviť triedy účtov do Javy pomocou projekcie, ktorú môžeme umiestniť napríklad do triedy **ucto.demo.IUcet**:

```
Projection Java As @Projection.Java(ROOTDIR =
"C:\Java\nerbeans\mojeucto\src\mojeucto\");
```

Potom skompilujeme celý projekt. Všimnite si, že aj keď v triede **ucto.demo.Denik** nemáme uvedenú deklaráciu projekcie, napriek tomu sa nám vytvorí, a to preto, lebo na ňu odkazuje trieda **Zapis** (ktorá takisto túto deklaráciu neobsahuje) a táto trieda odkazuje prostredníctvom referenčných atribútov na triedu **Analytika**, do ktorej je projekcia vnesená práve z triedy **IUcet**.

V príklade využijeme kód z predošlej časti, iba budeme novo implementovať metódu **zauctuj()**.

Najjednoduchšie bude vytvoriť novú triedu, napr. **Main2**, a do nej skopírovať obsah predchádzajúcej triedy **Main** s výnimkou práve metódy **zauctuj()**.

Prv než sa k tomu dostaneme, ešte strávime trochu času výkladom o možných spôsoboch prepojenia klienta Java a servera Caché. V metóde **connectToCache()** z predošlej časti je zostavený objekt **CacheConnection** volaním **CacheDatabase.getDatatase(...)**. Tomuto spôsobu zostavenia spojenia sa hovorí úplná väzba. Existuje aj jej odľahčený variant, volaný podobne: **CacheDatabase.getLightDatabase(...)**

Teraz si rozeberieme rozdiel medzi týmito pripojeniami. Úplná väzba otvára také pripojenie k

serveru Caché, v rámci ktorého sa objekty otvárané na strane klienta zároveň otvárajú i na strane servera Caché. Zo servera sa na klienta dovážajú hodnoty vlastností, ktoré sú vo vhodných okamihoch synchronizované s hodnotami na strane servera Caché.

Na druhej strane pri použití odľahčenej väzby nedochádza k inicializácii objektov na strane servera Caché, ale manipulácia s objektmi prebieha pomocou kódu SQL, ktorý vzniká automaticky pri kompilácii perzistentných tried Caché. To má za následok rýchlejšie vykonanie aktualizácií dát v databáze, pretože sa na strane Caché prístupuje priamo do dátových štruktúr – globálov. Tieto aktualizácie sú však z pohľadu klienta stále objektovými operáciami, stále sa volajú rovnaké metódy vystavených tried Caché. Na rozdiel od úplnej väzby však nemožno volať metódy inštancií, lebo – ako sme už uviedli – inštancie tried Caché sa na serveri nevytvárajú. Odľahčená väzba ponúka vyšší výkon než prípadné použitie SQL prostredníctvom rozhrania JDBC, lebo kód netreba dynamicky zostavovať a je kratší, s menším počtom prístupov do globálov.

V praxi možno z rovnakého klienta otvoriť tak úplné, ako i odľahčené prepojenie a použiť odľahčené prepojenie na priamu manipuláciu s triedami (zmeny, vytváranie nových inštancií, mazanie inštancií...) a úplné spojenie na volanie aplikačnej logiky, prezentovanej metódami inštancií tried Caché.

Na vyskúšanie si skúste upraviť metódu **connectToCache()** v predchádzajúcej časti tak, aby volala **getLightDatabase()**, a spusťte ju. To isté potom na porovnanie urobte s implementáciou metódy **zauctuj()**, uvedenou ďalej v tejto časti.

Ako by mohol kód na zaúčtovanie vyzerať, to si môžete pozrieť na www.infoware.sk v sekcii **Programujeme**.

V kóde si všimnite nasledujúce skutočnosti:
 ■ Každá metóda triedy Caché je vystavená do Javy pod rovnakým názvom, ako je defino-



vaná v Caché, s výnimkou tried začínajúcich sa znakom %. Tento znak je v projekciách metód nahradený podčiarkovníkom. Preto sa napr. objekty na strane Caché otvárajú v Jave niektorým variantom metódy **_open(...)**, zatiaľ čo v Caché sú k dispozícii metódy **%Open()** alebo **%OpenId()**.

■ Každá trieda vystavená z Caché sa otvára ako **RegisteredObject**, to je základná trieda objektov v Caché.

Na to, aby sme dostali správny typ triedy, použijeme casting.

- Pri prístupovaní k vlastnostiam tried Caché vystavených do Javy sa dôsledne používajú prístupové metódy.
- Na ukladanie zmien v objektoch sa volá metóda **_save()**, pre väčšie pohodlie vývojárov existuje aj metóda **save()**. V praxi netreba zisťovať výsledok operácie **save()**, ako je to uvedené v príklade, lebo pokiaľ všetko prebehne v poriadku, je výsledok 1, v opačnom prípade metóda zlyhá a vyhodí výnimku.

V budúcom pokračovaní sa pozrieme na to, ako pracovať so zložitejšími vlastnosťami objektov Caché a ako možno z Caché riadiť spôsob ich vystavenia do Javy.

■ DANIEL KUTÁČ,
Senior Sales Engineer spol. InterSystems B.V.
kutac@intersystems.cz