Navigace v XML dokumentech pomocí XPath

Řešení příkladů

Irena Mlýnková, Martin Nečaský

1. příklad

Vyberte názvy všech projektů.

Řešení

```
//project/info/title/text()
```

2. příklad

Vyberte název projektu s daným označením.

Řešení

```
//project[@oznaceni='im2']/info/title/text()
```

3. příklad

Vyberte vedoucí, kteří mají email.

Řešení

```
//supervisor[email]
```

4. příklad

Vyberte vedoucí, kteří nemají email.

Řešení

```
//supervisor[not(email)]
```

5. příklad

Vyberte přijmení vedoucích, kteří nemají email a mají uvedeno pouze příjmení.

Řešení

```
//supervisor[not(email or firstname)]/surname/text()
```

6. příklad

Vyberte první projekt ve druhé podskupině hlavní skupiny.

Řešení

```
/project-group/project-group[2]/project[1]
```

7. příklad

Vyberte telefony učitelů, kteří nejsou vedoucí žádného projektu ani skupiny. Pro účely dotazu berte jako jednoznačnou identifikaci učitele jeho příjmení.

Je možné napsat XPath výraz řešící tento dotaz v případě, že učitel je identifikován křestním jménem a příjmením dohromady?

Řešení

```
//teacher[not(surname = //supervisor/surname)]/phone
```

V případě identifikace pomocí kombinace hodnot dvou uzlů bychom potřebovali proměnné. Ty jsou až v XPath 2.0 a jejich použití si ukážeme až v rámci jazyka XQuery v dalších cvičeních.

8. příklad

Pro projekt s daným označením vyberte seznam předmětů, v rámci nichž může být projekt rešen (předměty se "sčítají" s těmi definovanými v nadřazených skupinách).

Řešení

Jedno z možných řešení je projít XML dokument shora dolů. Držíme se pouze v předcích hledaného projektu, dokud nedojdeme přímo do něj a bereme název kurzu z těchto uzlů:

```
//*[descendant-or-self::project[@oznaceni='im2']]/info/course/text()
```

Může být pro Vás ale přirozenější přístup zdola nahoru. Najdeme nejdříve hledaný projekt a vezmeme všechny jeho předky včetně projektu samotného. Z takto nalezených uzlů vezmeme názvy kurzů:

```
//project[@oznaceni='im2']/ancestor-or-self::*/info/course/text()
```

Nelze ale říci, které vyjádření dotazu je efektivnější. Spíše jde o osobní volbu, tj. Jak se vám nad daným XML dokumentem lépe přemýšlí. Druhý výraz se zdá efektivnější s ohledem na vyhodnocení. První totiž začíná výběrem všech uzlů v dokumentu. Optimalizátor by však měl zvládnout efektivní vyhodnocení i druhého dotazu (to je ale otázka).

9. příklad

Pro projekt s daným označením vyberte jméno vedoucího projektu. Pokud nemá projekt uveden vedoucího přímo, je jím vedoucí nadřazené skupiny atd.

Řešení

Ze sémantiky XML dokumentu vyplývá, že je třeba najít všechny vedoucí, kteří jsou přiřazeni buď přímo k projektu nebo nadřazených skupin. Potom vezmeme posledního vedoucího (v pořadí uzlů v dokumentu). Dvě možná řešení:

```
(//project[@oznaceni='im1']/ancestor-or-self::*/info/supervisor)[1]/firstname/
text()
```

```
(//*[descendant-or-self::project[@oznaceni='im1']]/info/supervisor)[last()]/
firstname/text()
```

10. příklad

Vyberte názvy projektů, které nemají přiřazeného vedoucího.

Řešení

```
//project[not(ancestor-or-self::*/info/supervisor)]/info/title/text()
```

11. příklad

Vyberte seznam všech různých předmětů. Ruzných znamená bez duplicit!

Řešení

XPath výraz vyjadřující tento dotaz vlastně jen prochází všechny kurzy v XML dokumentu a na výstup dá pouze ty, které ještě nevypsal. To znamená ty, které se ještě v XML dokumentu nevyskytly:¨

```
//course[not(. = preceding::course)]/text()
```

12. příklad

Vyberte název prvního a posledního projektu.

Řešení

Výraz `//project[position()=1 or position()=last()]/info/title` dotaz neřeší. Správně řešení je:

```
/descendant::project[position()=1 or position()=last()]/info/title
```