Ukládání a dotazování nad XML daty, XQuery

Obsah

XML Databáze

Co je to XML databáze

XML databáze je prakticky kolekce XML dokumentů uložených v nějakém úložišti.

Tímto úložištěm může být například

- souborový systém (nebo jiné perzistentní úložiště)
- relační databáze
- objektová databáze
- hierarchická nebo postrelační databáze
- nativní XML databáze

V přeneseném slova smyslu se pojmem XML databáze označují systémy řízení báze dat, které umožňují ukládat kolekce XML dokumentů (podobně jako se pojmem relační databáze nepesně označují systémy řízení báze dat pracující s relačním modelem). Někdy se také význam pojmu XML databáze omezuje pouze na nativní XML databáze.

Proč XML databáze

- XML je univerzální formát pro výměnu a reprezentaci informací.
- Transformace do jiných formátů je snadná.
- Ideální pro ukládání dokumentů.
- Univerzální formát pro datové modelování.
- Snadná a přirozená reprezentace objektů (agregace, dědičnost, asociace) - viz XML Schema.

Typickým případem kdy je vhodné použít XML databázi je situace, kdy pořizujeme nebo získáváme data přímo ve formátu XML a potřebujeme vzniklé dokumenty ukládat.

Kdy nepoužít XML databázi

- Když vyhovuje jiný datový model (většinou relační).
- U aplikací se specifickými požadavky (výkon).
- Když potřebujeme vlastnosti, které zatím běžně dostupné XML databáze neposkytují

XML databáze jsou zatím v plenkách

- Nedosahují robusnosti relačních databázových systémů (transakce, souběžný přístup, škálovatelnost), i když mnohé tyto vlastnosti jsou již nabízeny - viz např. přehled na Wikipedii: Native XML Databases [http://en.wikipedia.org/wiki/XML_database#Native_XML_databases].

- Chybí plná podpora standardů.
- Metody pro indexování a optimalizaci vyhodnocování dotazů se teprve vyvíjí.

XML databáze nejsou a nikdy nebudou následníkem relačních databází; tvoří k nim alternativu vhodnou pro určité typy dat a aplikací.

XML a relační databáze

XML dokumenty je možné ukládat v relačních databázích pomocí

- polí typu TEXT nebo BLOB
- specializovaných schémat pro konkrétní aplikace
- univerzálních schémat bez indexování struktury
- univerzálních schémat s indexováním struktury

Mnoho existujících relačních SŘBD poskytuje většinou proprietární podporu pro ukládání XML (rozšíření SQL).

Při indexování XML dokumentů nalézá uplatnění mnoho algoritmů a technik z relačních SŘBD.

Nativní XML databáze

Existuje řada nativních XML databází, mezi nejznámější patří

- Tamino [http://www.softwareag.com/tamino/News/tamino_41.htm], což je zřejmě nejznámější komerční nativní XML databáze.
- BaseX [http://www.inf.uni-konstanz.de/dbis/basex/index] je volně dostupná nativní XML databáze vyvíjená na platformě Java 6 týmem *Univerzity Konstanz*. Je jednoduchá pro instalaci a vhodná pro interaktivní práci s XML daty a XQuery!
- eXist [http://exist-db.org/], což je open source podporující mimo jiné XQuery [http://www.w3.org/XML/Query], XUpdate [http://xmldb-org.sourceforge.net/xupdate/], XML:DB API [http://xmldb-org.sourceforge.net/xapi/] a XML-RPC. Jde dle mého názoru o nejlepší existující open source databázi. Je šířena pod licencí GNU LGPL [http://www.gnu.org/copyleft/lesser.html].
- Apache Xindice [http://xml.apache.org/xindice/], cože je open source nativní XML databáze od Apache Software Foundation [http://www.apache.org/]. Podporuje XPath [http://www.w3.org/TR/xpath], XUpdate [http://xmldb-org.sourceforge.net/xupdate/], XML:DB API [http://xmldb-org.sourceforge.net/xapi/] a XML-RPC. Není však nadále seriózně vyvíjená a podporovaná.

Indexování XML dokumentů

Efektivní ukládání a vyhledávání

Jak již bylo zmíněno, kolekce XML dokumentů se dají ukládat různým způsobem, které se liší zejména efektivitou různých operací. Ve všech zmíněných případech lze však aplikovat pomocné indexační struktury pro zvýšení jejich efektivity.

Indexování XML dokumentů umožňuje

- efektivní vyhledávání v kolekcích XML dokumentů,
- efektivní provádění XML transformací,
- efektivní aktualizaci dokumentů,
- efektivní navigaci v rámci dokumentu.

Nejčastěji je cílem indexování efektivní vyhodnocování XPath dotazů.

Indexování pro vyhodnocování XPath výrazů

- Indexování textových (hodnotových) informací
  - hodnoty textových uzlů;
  - hodnoty atributů;
  - jména elementů a atributů.
- Indexování strukturálních vztahů (osy XPath)
  - Vyhodnocení relace je na ose/není na ose (např. je uzel *x* potomkem uzlu *y*?);
  - Které uzly leží na dané ose (vrať mi všechny potomky uzlu *x*).

Pro indexování textových informací se používají invertované soubory (B+ strom, hešovací tabulky) nebo plnotextové indexy (fulltext). Pro indexování strukturálních vztahů byla vyvinuta řada metod, každá z nich má však nějaká slabá místa. Obvykle jsou tyto metody založeny na nějakém číslovacím schématu.

Vyhodnocování XPath dotazů

Pro vyhodnocení XPath dotazů je nutné

- vyhodnotit všechny predikáty a testy uzlů,
- vyhodnotit všechny strukturální vztahy,
- spojit (*join*) výsledky.

Pořadí operací může výrazně ovlivnit efektivitu zpracování.

Optimalizace XPath výrazů

- Zjednodušování a transformace dotazů
  - s využitím znalosti struktury dokumentu;
  - s využitím znalosti složitosti jednotlivých operací;
  - s využitím statistických informací;
  - eliminiace zbytečných predikátů.
- Výběr vhodného pořadí pro zpracování.
- Výběr vhodné metody pro vyhodnocení.

V současné době jsou bohužel schopnosti optimalizátorů v XML databázích omezené a nepříliš účinné.

Efektivita vyhodnocování XPath výrazů

Různě zapsané XPath/XQuery dotazy se stejným významem se mohou vyhodnotit různě efektivně. Dokonalý optimalizátor dotazů by měl toto eliminovat. Jenomže

- dokonalý optimalizátor (zatím?) neexistuje,
- mnoho implementací ani žádný nemá (XSLT procesory),
- optimizer mnohdy nemá k dispozici všechny informace.

Jak psát efektivní XPath dotazy?

- Pokud to není nezbytně nutné, efektivitu neřešte a dejte přednost přehlednosti.
- Efektivita jednotlivých operací silně závisí na konkrétní implementaci.
- Proto je pro dosažení efektivity vhodné znát způsob vyhodnocování XPath dotazů konkrétním XPath procesorem.
- Alternativou může být experimentální metoda.

Co je efektivnější?

- `/people/person` nebo `//person`
- `//person[@id=1]/name` nebo `//name[parent::*/@id=1]`
- `//person[number(@id)=$a]` nebo `//person[@id=$a]`
- `{ for $p in //person[@id=$a] return $p }` nebo `{ for $p in //person where $p/@id=$a return $p }`

XQuery

Charakteristika

- Jazyk pro specifikaci dotazů k vyhledání a extrakci uzlů XML (elementy, atributy) z dokumentů a konstrukci výstupního XML dokumentu.

Charakteristika (2)

- V současnosti (a zdá se i budoucnosti) je XQuery základním dotazovacím jazykem nad XML dokumenty.
- Definován konsorciem W3C, stane se specifikací - momentálně ve stavu *Last Call Working Draft*, viz http://www.w3.org/XML/Query.
- Založen na XPath 2.0 datovém modelu, operátorech a funkcích
- Podporují ho hlavní producenti databázových strojů (IBM, MS, Oracle a další)

Kde se XQuery použije (a nepoužije)

XQuery se typicky použije např. pro:

- dotazy, kde je složitější extrakční (selekční) část a jednodušší konstrukční část
- v opačném případě je lépe použít XSLT
- nebo dokonce zpracování obecnějším programovacím prostředím - např. manipulaci s (DOM) objektovým stromem dokumentu.

Příklad - zdrojový dokument

Ukázka zdrojového dokumentu, XQuery dotazů nad nimi a jejich výsledku.

Příklad 1. Zdrojový dokument

```
<?xml version="1.0" encoding="Windows-1250"?>
<addressbook>
    <person category="friends">
        <firstname>Petr</firstname>
        <lastname>Novák</lastname>
        <date-of-birth>1969-05-14</date-of-birth>
        <email>novak@myfriends.com</email>
        <characteristics lang="en">Very good friend</characteristics>
    </person>
    <person category="friends">
        <firstname>Jaroslav</firstname>
        <lastname>Nováèek</lastname>
        <date-of-birth>1968-06-14</date-of-birth>
        <email>novacek@myfriends.com</email>
        <characteristics lang="en">Another good friend</characteristics>
    </person>
    <person category="staff">
        <firstname>Jan</firstname>
        <lastname>Horák</lastname>
        <date-of-birth>1970-02-01</date-of-birth>
        <email>horak@mycompany.com</email>
        <characteristics lang="en">Just colleague</characteristics>
    </person>
    <person category="friends">
        <firstname>Erich</firstname>
        <lastname>Polák</lastname>
        <date-of-birth>1980-02-28</date-of-birth>
        <email>erich@myfriends.com</email>
        <characteristics lang="en">Good friend</characteristics>
    </person>
</addressbook>
```

Příklad - jednoduchý dotaz XPath

Ukázka XQuery dotazu nad výše uvedeným zdrojovým dokumentem. Úloha: "extrakce všech příjmení v adresáři".

Dotaz je v podstatě XPath výrazem - vybere tedy všechny elementy `lastname`.

Příklad 2. XQuery

```
doc('myaddresses.xml')/addressbook/person/lastname
```

Příklad - spuštění v Saxon 9.0j

XSLT procesor Saxon je od verzí 8.x rovněž XQuery procesorem. K vykonání XQuery dotazu je třeba:

- instalovat Saxon, např. 9.0.0.4j ("j" značí javovou implementaci - kromě toho existuje i .NET) rozbalením do adresáře - např. `c:\devel\saxon9-0-0-4j` [http://www.google.com/search?q=c:\devel\saxon9-0-0-4j] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=c:\devel\saxon9-0-0-4j]
- přepnout se do tohoto adresáře: **cd `c:\devel\saxon9-0-0-4j` [http://www.google.com/search?q=c:\devel\saxon9-0-0-4j] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=c:\devel\saxon9-0-0-4j] [http://www.google.com/search?q=cd]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=cd]
- uložit výše uvedený dotaz např. do souboru `lastnames.xq` [http://www.google.com/search?q=lastnames.xq] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=lastnames.xq].
- výše uvedený dokument s "addressbook" uložíme do `myaddresses.xml` [http://www.google.com/search?q=myaddresses.xml] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=myaddresses.xml] ve stejném adresáři.
- z příkazové řádky spustit: **java -classpath saxon9.jar net.sf.saxon.Query -o result.xml lastnames.xq [http://www.google.com/search?q=java -classpath saxon9.jar net.sf.saxon.Query -o result.xml lastnames.xq]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java -classpath saxon9.jar net.sf.saxon.Query -o result.xml lastnames.xq]

Příklad - výsledek

Uvedený XQuery dotaz vrátí do souboru `result.xml` [http://www.google.com/search?q=result.xml] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=result.xml] nad zmíněným dokumentem toto:

Příklad 3. Výsledek aplikace dotazu

```
<lastname>Novák</lastname>
<lastname>Nováček</lastname>
<lastname>Horák</lastname>
<lastname>Polák</lastname>
```

XQuery konstrukce

FLWOR

FLWOR je zkrácené označení pro strukturu XQuery dotazů.

Je to akronym z:

(F)or — Úvodní část dotazu specifikuje cyklus vč. řídicí proměnné, do níž jsou postupně přiřazovány jednotlivé hodnoty vybrané XPath výrazem za klíčovým slovem "in".

(L)et — V této sekci lze provést přiřazení do dalších proměnných použitelných následně.

(W)here — Specifikuje selekční podmínku, tzn. které uzly (hodnoty) vybrané ve "for" budou skutečně použity.

V podmínce lze použít i proměnné vázané v sekci "let".

(O)rder — Takto vybrané uzly (hodnoty) lze výrazem v této sekci uspořádat.

(R)eturn — Co bude vráceno, zkonstruováno ze získaných uzlů (hodnot).

FLWOR - jednoduchý příklad

Selekci uzlů lze specifikovat buďto přímo v XPath výrazu ve "for" nebo až v selekčním "where".

"Vrať datum narození Poláka."

Příklad 4. FLWOR

```
for $person in doc('myaddresses.xml')/addressbook/person
where $person/lastname='Polák'
return $person/date-of-birth
```

Spuštění vrátí:

```
<?xml version="1.0" encoding="UTF-8"?>
<date-of-birth>1980-02-28</date-of-birth>
```

Implementace XQuery

SAXON od verzí 7.x

- instalovat (rozbalit) Saxon od verze 7.x (lze i 8.x, 9.x) rozbalením do libovolného adresáře
- přepnout se do tohoto adresáře a
- z příkazové řádky spustit: **java -classpath saxon9.jar net.sf.saxon.Query -o result.xml queryfile.xq [http://www.google.com/search?q=java -classpath saxon9.jar net.sf.saxon.Query -o result.xml queryfile.xq]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java -classpath saxon9.jar net.sf.saxon.Query -o result.xml queryfile.xq]

- Existuje i stejná verze pro .NET (čili jako .DLL a .EXE soubory)

Nativní XML databáze

Nativní XML databázové systémy často podporují dotazování přes XQuery.

Patří mezi ně např.:

- Berkeley DB XML [http://www.sleepycat.com/products/index.shtml]
- eXist [http://exist.sourceforge.net/]

Rozhraní pro práci s XML databázemi

Rozhraní pro práci s XML databázemi

- XML:DB [http://xmldb.org]
- XUpdate [???] („spící“), resp. nyní XQuery Update Facility [http://www.w3.org/TR/xquery-update-10/] (součást XQuery aktivit)
- XML Query API for Java [http://www.jcp.org/aboutJava/communityprocess/edr/jsr225/] (XQJ)

Rozhraní XML:DB

- Specifikováno konsorciem XML:DB [http://xmldb.org]
- Podobná koncepce jako JDBC [http://java.sun.com/products/jdbc/]
- Rozhraní je specifikováno na poměrně abstraktní úrovni, implementační detaily jsou skryty.
- Základní objekty:
  - `Driver` - podobně jako JDBC Driver - abstrahuje přístup ke konkrétnímu DBS, implementuje rozhraní Database
  - `DatabaseManager` - řídí zavádění a správu jednotlivých ovladačů (Driver) databázových systémů
  - `Collection` - kolekce XML dokumentů v databázi. Konceptuálně srovnatelné s relační tabulkou (či celou databází). Kolekce totiž mohou být libovolně vnořené.
  - Services - rozhraní konkrétních služeb. Bez nich by XML:DB takřka nemělo smysl - teprve služby definují, co databáze „umí“. Typickou službou je např. `XPathQueryService` na vyhledávání dokumentů a jejich částí přes XPath. Další službou je např. `XUpdateQueryService`.
  - `Resource` - zhruba odpovídá JDBC resource. Obecně „nějaký“ zdroj - nemusí být jen XML, ale i binární. Je-li XML, pak např. SAX, DOM, XML text...

Vrstvy XML:DB API

Rozhraní XML:DB je pro pohodlí programátora členěno do úrovní. Vždy si jednu z nich vybereme a využíváme její nabídky:

- XML:DB Core Level 0 - musí implementovat všechny DBS. Obsahuje základní rozhraní pro *kolekce* (collections), *zdroje* (resources), and *služby* (services).

- XML:DB Core Level 1 - navíc obsahuje `XPathQueryService`.

Ukázka XML:DB programu

Příklad 5. Příklad programu využívajícího XML:DB

```
import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;
import org.xmldb.api.*;
public class Query {
    public static void main(String[] args) throws Exception {
        Collection col = null;
        try {
            String driver = null;
            String prefix = null;

            if ( ( args.length == 1 ) && args[0].equals("dbxml") ) {
                driver = "org.dbxml.client.xmldb.DatabaseImpl";
                prefix = "xmldb:dbxml:///db/";

            } else {
                driver = "org.xmldb.api.reference.DatabaseImpl";
                prefix = "xmldb:ref:///";
            }

            Class c = Class.forName(driver);
            Database database = (Database) c.newInstance();

            if ( ! database.getConformanceLevel().equals("1") ) {
                System.out.println("This program requires a Core Level 1 XML:DB "
                System.exit(1);
            }
            DatabaseManager.registerDatabase(database);

            col = DatabaseManager.getCollection(prefix + "addresses");
            String xpath = "/address[@id = 1]";

            XPathQueryService service = (XPathQueryService) col.getService("XPathQ

            ResourceSet resultSet = service.query(xpath);
            ResourceIterator results = resultSet.getIterator();

            while (results.hasMoreResources()) {
                Resource res = results.nextResource();
                System.out.println((String) res.getContent());
            }
        } catch (XMLDBException e) {
            System.err.println("XML:DB Exception occurred " + e.errorCode + " " +
        } finally {
            if (col != null) { col.close(); }
        }
    }
}
```

Použití XUpdate v databázích s XML:DB

Příklad 6. Příklad modifikace pomocí XUpdate

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
   <xupdate:update select="/address[@id = 1]/name/last">Herman</xupdate:update>
</xupdate:modifications>
```

Tento XUpdate dotaz nahradí příjmení za „Herman“.

eXist

eXist

`eXist` je podobně jako `Xindice` open-source databáze podporující XML:DB.

Je možné ji provozovat jako:

- samostatně běžící (standalone) server, přístupný soketovým spojením (XML-RPC, HTTP)
- jako in-process (embedded) -server běžící v témže běhu JVM jako aplikace, která jej používá
- jako webová aplikace - .war archív, který se instaluje (deploy) na servletový kontejner (např. Tomcat, Jetty, Bajie...)
- eXist má Jetty server přibalen v instalačním balíku, viz eXist download [???].

eXist: instalace a spuštění

Je možno instalovat na Win NT/2000, Linuxu...

Postupujeme přesně podle instrukcí v eXist Quickstart [http://exist-db.org/quickstart.html].

Doporučuji (odzkoušeno na Win 2000 Pro):

- spustit **java -jar eXist-0.9.1-install.jar [http://www.google.com/search?q=java -jar eXist-0.9.1-install.jar]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java -jar eXist-0.9.1-install.jar]
- řídit se instalačními pokyny, instalovat např. do `\devel\eXist` [http://www.google.com/search?q=\devel\eXist] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=\devel\eXist].
- přepnout se do instalačního adresáře, otevřít Command Shell/Prompt a spustit eXist přes Jetty webový server: **bin\startup.bat [http://www.google.com/search?q=bin\startup.bat]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=bin\startup.bat].
- eXist ohlásí, že se spustil. Případné chybové hlášky loggeru ignorovat.

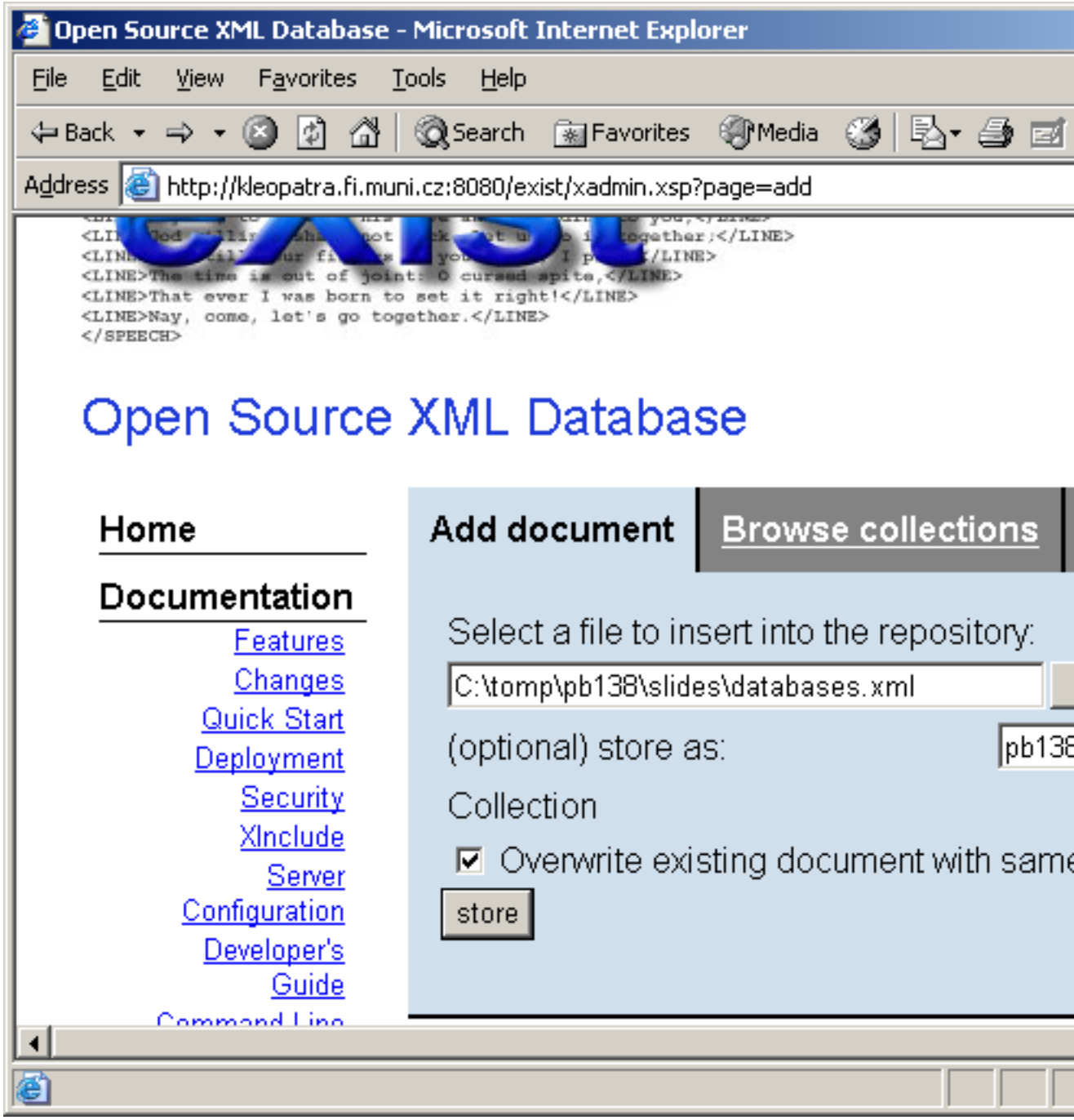
eXist: použití přes webové rozhraní

Pokud se v konfiguracích nic neměnilo, je služba eXist dostupná přes URL podobné tomuto: http://kleopatra.fi.muni.cz:8080/exist/. (tj. port 8080, cesta /exist)

Nyní můžeme vytvořit kolekci, přidat soubor, dotazovat se...

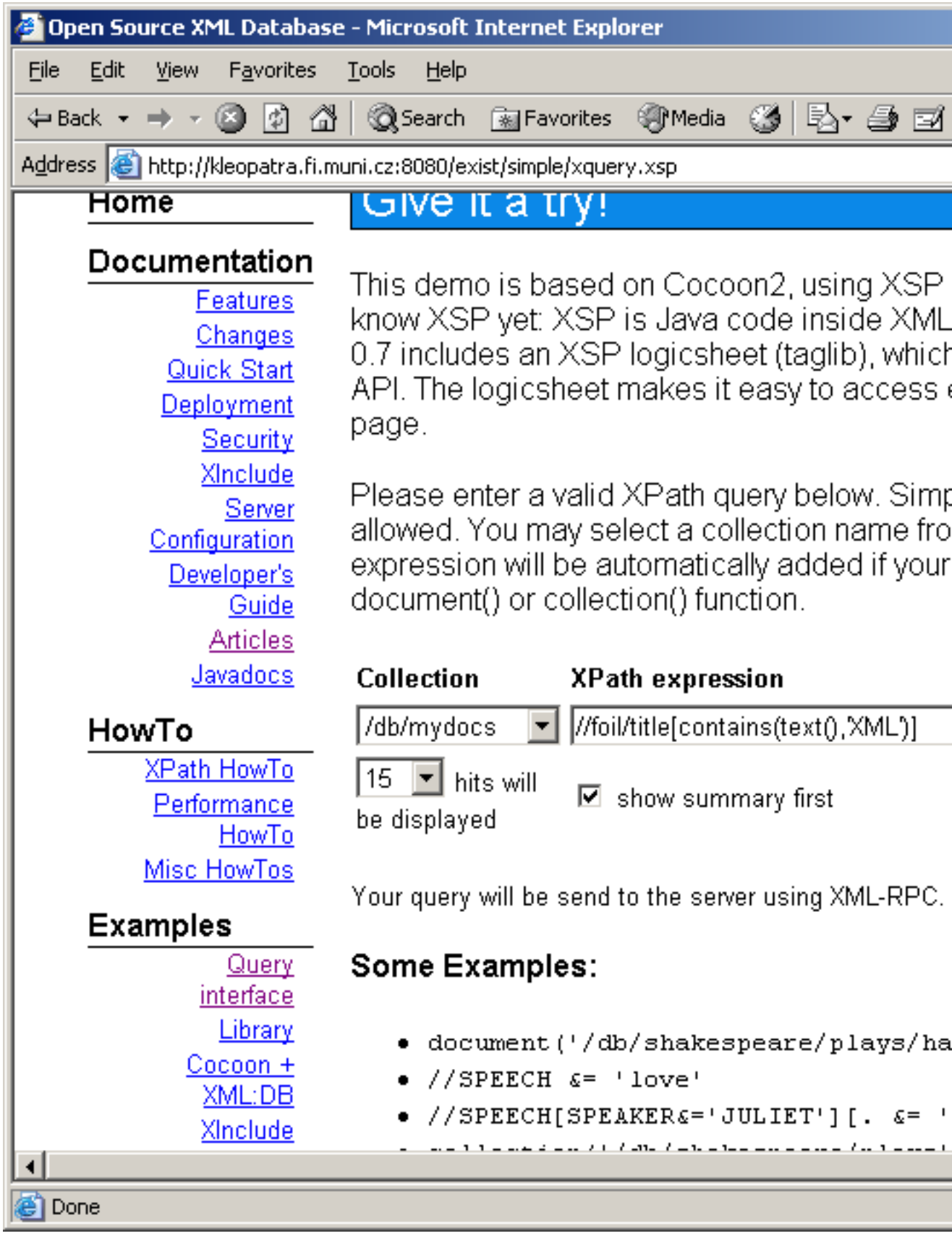
eXist: vložení dokumentu do kolekce

Vytvoříme kolekci `mydocs` a do ní přidáme dokument `databases.xml` [http://www.google.com/search?q=databases.xml] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=databases.xml] (tyto slidy):



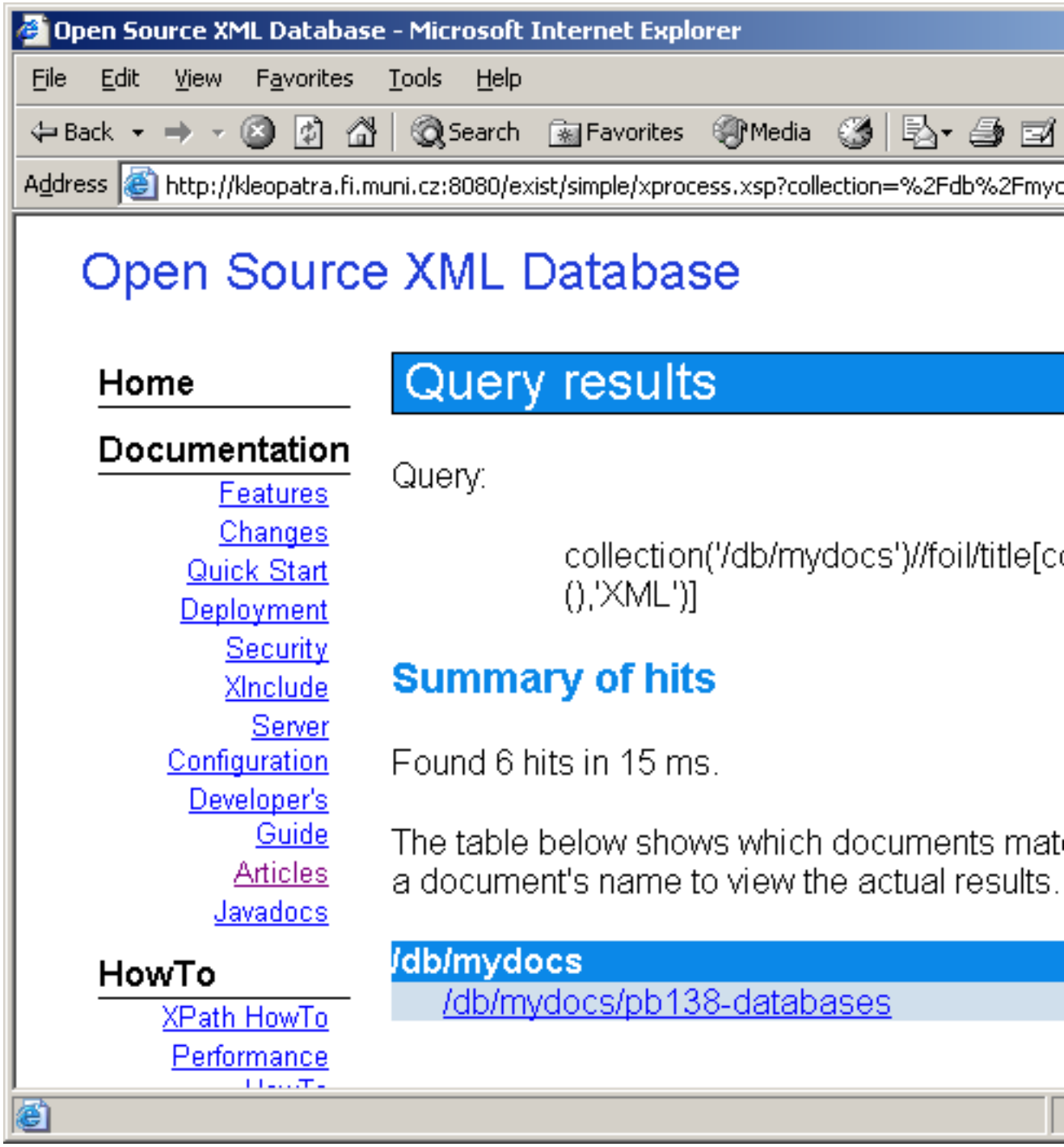
eXist: dotazování - zadání dotazu

Zadáme XPath dotaz, specifikujeme rozsah (ve které koleci hledat), uvedeme, kolik vyhovujících dokumentů v odpovědi vrátit.


Open Source XML Database - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://kleopatra.fi.muni.cz:8080/exist/simple/xquery.xsp
Home
Documentation
Features
Changes
Quick Start
Deployment
Security
XInclude
Server Configuration
Developer's Guide
Articles
Javadocs
HowTo
XPath HowTo
Performance HowTo
Misc HowTos
Examples
Query interface
Library
Cocoon + XML:DB
XInclude
Give it a try!
This demo is based on Cocoon2, using XSP
know XSP yet: XSP is Java code inside XML
0.7 includes an XSP logicsheet (taglib), which
API. The logicsheet makes it easy to access
page.
Please enter a valid XPath query below. Simp
allowed. You may select a collection name fro
expression will be automatically added if your
document() or collection() function.
Collection
XPath expression
/db/mydocs
//foil/title[contains(text(),'XML')]
15 hits will be displayed
show summary first
Your query will be send to the server using XML-RPC.
Some Examples:
document('/db/shakespeare/plays/ha
//SPEECH &= 'love'
//SPEECH[SPEAKER&='JULIET'][. &= '
Done

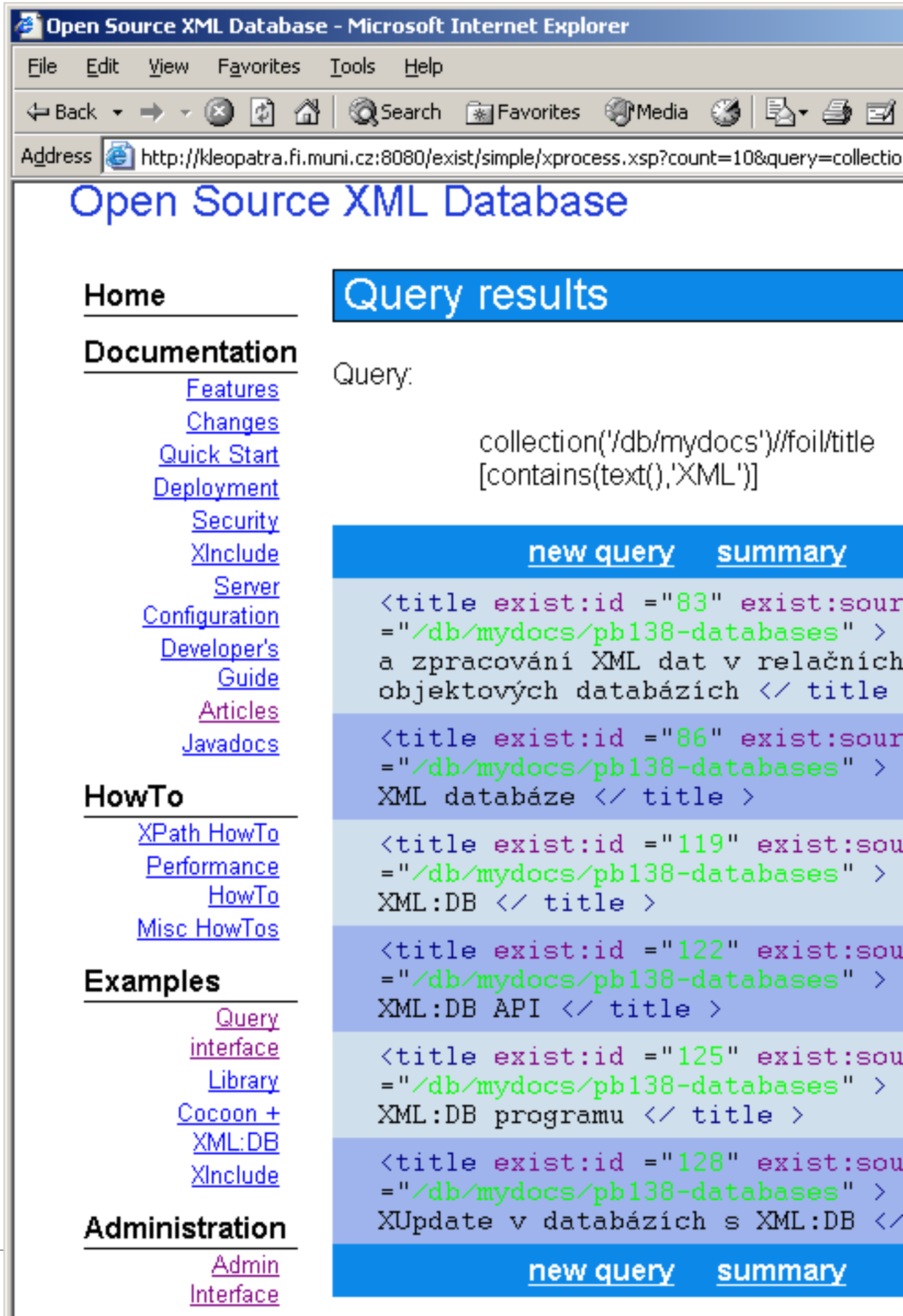
eXist: dotazování - sumarizovaný výsledek dotazu

eXist sdělí, ve kterých kolekcích které dokumenty vyhovují dotazu:



eXist: dotazování - prohlížení jednotlivých výsledků dotazu

eXist pro každý vyhovující dokument vrátí uzly, které dotazu vyhovují


Open Source XML Database - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://kleopatra.fi.muni.cz:8080/exist/simple/xprocess.xsp?count=10&query=collectio
Open Source XML Database
Home
Documentation
Features
Changes
Quick Start
Deployment
Security
XInclude
Server Configuration
Developer's Guide
Articles
Javadocs
HowTo
XPath HowTo
Performance HowTo
Misc HowTos
Examples
Query interface
Library
Cocoon + XML:DB
XInclude
Administration
Admin Interface
Query results
Query:
collection('/db/mydocs')//foil/title
[contains(text(),'XML')]
new query summary
<title exist:id ="83" exist:sour
="/db/mydocs/pb138-databases" >
a zpracování XML dat v relačních
objektových databázích </ title
<title exist:id ="86" exist:sour
="/db/mydocs/pb138-databases" >
XML databáze </ title >
<title exist:id ="119" exist:sou
="/db/mydocs/pb138-databases" >
XML:DB </ title >
<title exist:id ="122" exist:sou
="/db/mydocs/pb138-databases" >
XML:DB API </ title >
<title exist:id ="125" exist:sou
="/db/mydocs/pb138-databases" >
XML:DB programu </ title >
<title exist:id ="128" exist:sou
="/db/mydocs/pb138-databases" >
XUpdate v databázích s XML:DB </
new query summary

XQuery 3.0

Charakteristika

- Celá řada nových rysů, funkcí, ošetření výjimek apod.
- W3C Doporučení ve stádiu WD (prosinec 2010)

Nové rysy jazyka Query 3.0

XQuery se typicky použije např. pro:

- `group by` clause in FLWOR Expressions (3.9.7 Group By Clause).
- tumbling window and sliding window in FLWOR Expressions (3.9.4 Window Clause).
- `count` clause in FLWOR Expressions (3.9.6 Count Clause).
- `allowing empty` in 3.9.2 For Clause, for functionality similar to outer joins in SQL.
- `try/catch` expressions (3.14 Try/Catch Expressions).
- Dynamic function invocation (3.2.2 Dynamic Function Invocation).
- Inline functions (3.1.7 Inline Functions).
- Private functions (4.18 Function Declaration).
- Nondeterministic functions (4.18 Function Declaration)
- Switch expressions (3.12 Switch Expression)
- Computed namespace constructors (3.8.3.7 Computed Namespace Constructors).
- Output declarations (2.2.4 Serialization).
- Annotations (4.15 Annotations). Annotation assertions in function tests.

BaseX

BaseX

- `BaseX` je podobně jako eXist open-source databáze podporující XQuery.
- Jde o open-source projekt dostupný na http://basex.org/.