

# **Web Forms and SPARQL Queries**

**Bob DuCharme**  
**Semantic Tech and Business Conference**  
**Washington D.C.**

**November 30, 2011**





# Introductions

- Presentation and all its URLs:  
<http://snee.com/semtech/2011/dc/>
- Me: SGML and XML at Moody's, LexisNexis, Innodata Isogen, TopQuadrant
- Weblog: <http://www.snee.com/bobdc.blog>
- Twitter: @bobdc



# SPARQL

- **SPARQL Protocol and RDF Query Language**
- “Trying to use the Semantic Web without SPARQL is like trying to use a relational database without SQL” – Tim Berners-Lee
- **SPARQL Endpoint:** a (web) service that accepts a SPARQL query and returns the result set



# Learning SPARQL: The Book

learningsparql.com

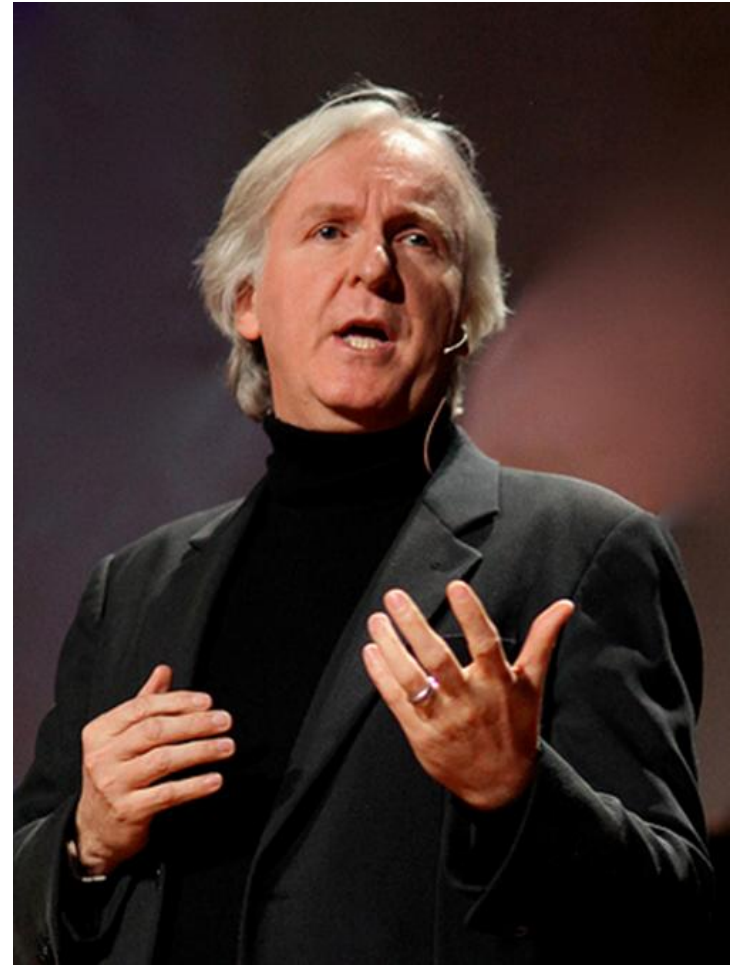
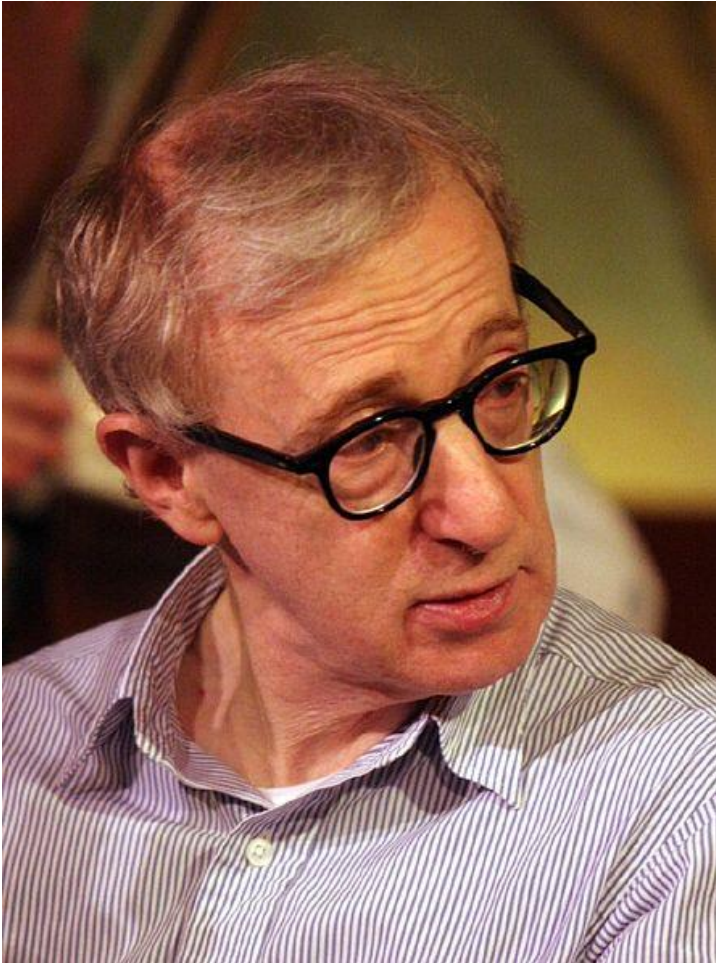
@learningsparql

Print bound copies,  
PDF, ePUB, Kindle,  
mobi...





# Actors directed by both?





# The SPARQL Query



```
PREFIX m: <http://data.linkedmdb.org/resource/movie/>
SELECT DISTINCT ?actorName WHERE {
```

```
  ?dir1      m:director_name "Woody Allen".
  ?dir2      m:director_name "James Cameron".
  ?dir1film  m:director ?dir1;
             m:actor ?actor.
```

```
  ?dir2film  m:director ?dir2;
             m:actor ?actor.
```

```
  ?actor     m:actor_name ?actorName.
```

```
}
```

# Running query on SNORQL form

The screenshot shows a web browser window titled "SPARQL Explorer for http://www...". The address bar contains the URL: `www.linkedmdb.org/snorql/?query=PREFIX+m%3A+<http%3A%2F%2Fdata.linkedmdb.org%2Fresource%2Fmovie%2F>%0D%0ASELECT+DI`. The main content area is titled "SPARQL Explorer for http://www.linkedmdb.org/sparql".

**SPARQL:**

```

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX oddlinker: <http://data.linkedmdb.org/resource/oddlinker/>
PREFIX map: <file:/C:/d2r-server-0.4/mapping.n3#>
PREFIX db: <http://data.linkedmdb.org/resource/>
PREFIX dbpedia: <http://dbpedia.org/property/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>

PREFIX m: <http://data.linkedmdb.org/resource/movie/>
SELECT DISTINCT ?actorName WHERE {

  ?dir1      m:director_name "Woody Allen".
  ?dir2      m:director_name "James Cameron".
  ?dir1film  m:director ?dir1;
             m:actor ?actor.

  ?dir2film  m:director ?dir2;

```

**Results:**

**SPARQL results:**

actorName
"Leonardo DiCaprio"
"Kathy Bates"
"Sigourney Weaver"

Powered by [D2R Server](#)

# Running query on SNORQL form

SPARQL Explorer for <http://www.linkedmdb.org/sparql>

SPARQL:

```

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX oddlinker: <http://data.linkedmdb.org/resource/oddlinker/>
PREFIX map: <file://C:/d2r-server-0.4/mapping.n3#>
PREFIX db: <http://data.linkedmdb.org/resource/>
PREFIX dbpedia: <http://dbpedia.org/property/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>

PREFIX m: <http://data.linkedmdb.org/resource/movie/>
SELECT DISTINCT ?actorName WHERE {

  ?dir1      m:director_name "Woody Allen".
  ?dir2      m:director_name "James Cameron".
  ?dir1film m:director ?dir1;
            m:actor ?actor.

  ?dir2film m:director ?dir2;

```

Results:  XSLT stylesheet URL:

SPARQL results:

actorName
"Leonardo DiCaprio"
"Kathy Bates"
"Sigourney Weaver"

Powered by [D2R Server](#)

**URL: <http://www.linkedmdb.org/snorql/xml-to-html.xsl>**





# XML+XSLT results

SPARQL Query Results

www.linkedmdb.org/sparql?query=PREFIX+owl%3A+<http%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23>%0D%0APREFIX+xsd%3A

### SPARQL Query Results

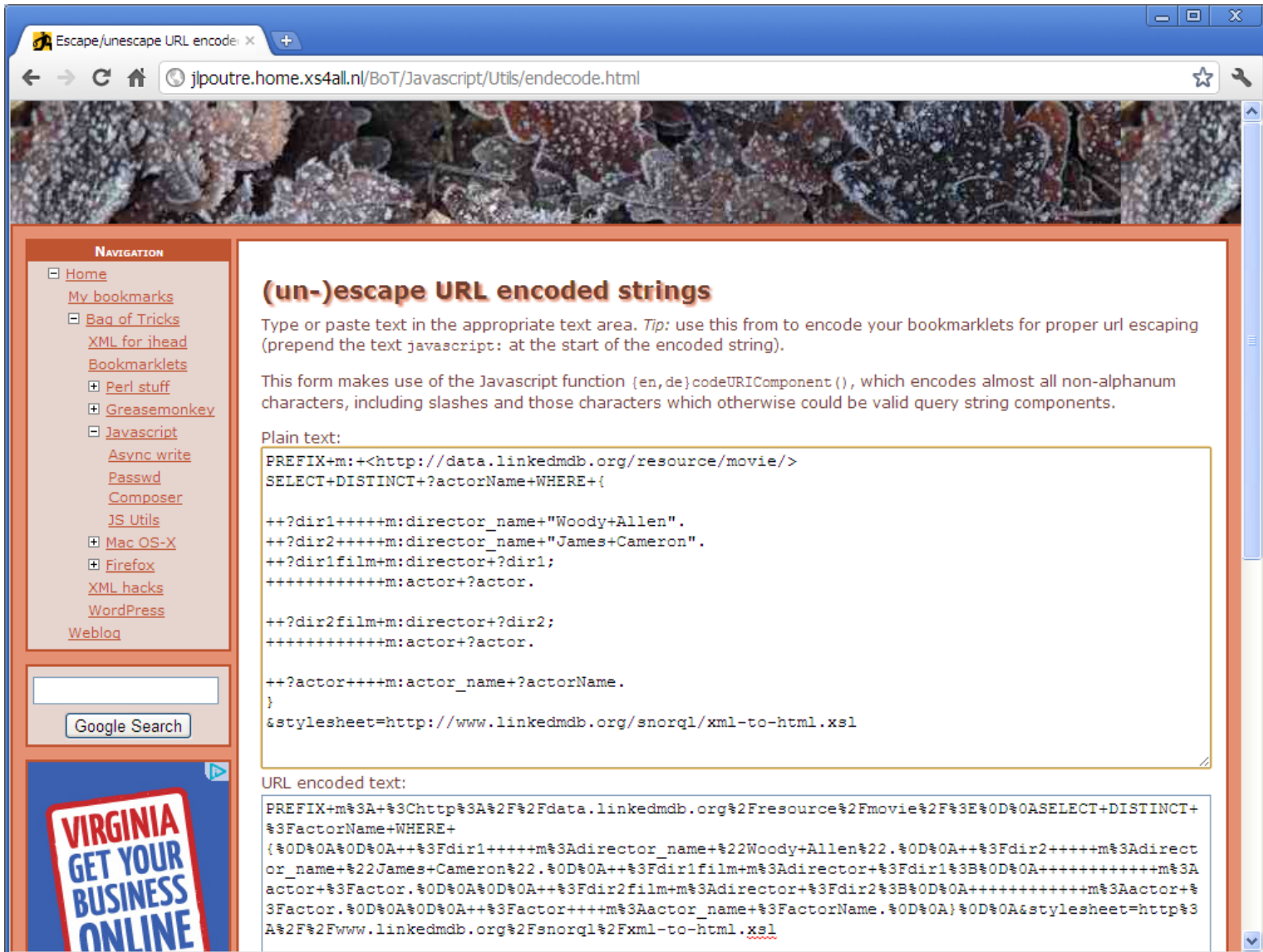
actorName
"Leonardo DiCaprio"
"Kathy Bates"
"Sigourney Weaver"



# Full Linked MDB query URI

```
http://www.linkedmdb.org/sparql?query=PREFIX+m
%3A+%3Chttp%3A%2F%2Fdata.linkedmdb.org%2Fresou
rce%2Fmovie%2F%3E%0D%0A%0D%0A%0D%0A++%3Fdir1++++m%3Adir
ector_name+%22Woody+Allen%22.%0D%0A++%3Fdir2++
+++m%3Adirector_name+%22James+Cameron%22.%0D%0
A++%3Fdir1film+m%3Adirector+%3Fdir1%3B%0D%0A++
+++++++m%3Aactor+%3Factor.%0D%0A%0D%0A++%3F
dir2film+m%3Adirector+%3Fdir2%3B%0D%0A+++++++
++++m%3Aactor+%3Factor.%0D%0A%0D%0A++%3Factor+
+++m%3Aactor_name+%3FactorName.%0D%0A}%0D%0A&s
tylesheet=http%3A%2F%2Fwww.linkedmdb.org%2Fsno
rql%2Fxml-to-html.xsl
```

# Unescaped version of blue text



Escape/unescape URL encode: x

jlpoutre.home.xs4all.nl/BoT/Javascript/Utils/endecode.html

**NAVIGATION**

- Home
- My bookmarks
- Bag of Tricks
  - XML for ihead
  - Bookmarklets
  - Perl stuff
  - Greasemonkey
- Javascript
  - Asvnc write
  - Passwd
  - Composer
  - JS Utils
  - Mac OS-X
  - Firefox
  - XML hacks
  - WordPress
  - Weblog

Google Search

**VIRGINIA GET YOUR BUSINESS ONLINE**

## (un-)escape URL encoded strings

Type or paste text in the appropriate text area. *Tip:* use this from to encode your bookmarklets for proper url escaping (prepend the text javascript: at the start of the encoded string).

This form makes use of the Javascript function `{en,de}encodeURIComponent()`, which encodes almost all non-alphanum characters, including slashes and those characters which otherwise could be valid query string components.

Plain text:

```

PREFIX+m:<http://data.linkedmdb.org/resource/movie/>
SELECT+DISTINCT+?actorName+WHERE+{
++?dir1++++m:director_name+"Woody+Allen".
++?dir2++++m:director_name+"James+Cameron".
++?dir1film+m:director+?dir1;
+++++m:actor+?actor.

++?dir2film+m:director+?dir2;
+++++m:actor+?actor.

++?actor++++m:actor_name+?actorName.
}
&stylesheet=http://www.linkedmdb.org/snorql/xml-to-html.xsl
    
```

URL encoded text:

```

PREFIX+m%3A+%3Chttp%3A%2F%2Fdata.linkedmdb.org%2Fresource%2Fmovie%2F%3E%0D%0ASELECT+DISTINCT+
%3FactorName+WHERE+
{%0D%0A%0D%0A++%3Fdir1++++m%3Adirector_name+%22Woody+Allen%22.%0D%0A++%3Fdir2++++m%3Adirect
or_name+%22James+Cameron%22.%0D%0A++%3Fdir1film+m%3Adirector+%3Fdir1%3B%0D%0A+++++m%3A
actor+%3Factor.%0D%0A%0D%0A++%3Fdir2film+m%3Adirector+%3Fdir2%3B%0D%0A+++++m%3Aactor+%
3Factor.%0D%0A%0D%0A++%3Factor++++m%3Aactor_name+%3FactorName.%0D%0A}%0D%0A&stylesheet=http%3
A%2F%2Fwww.linkedmdb.org%2Fsnorql%2Fxml-to-html.xsl
    
```



# Revised Linked MDB query URI

```
http://www.linkedmdb.org/sparql?query=PREFIX+m
%3A+%3Chttp%3A%2F%2Fdata.linkedmdb.org%2Fresou
rce%2Fmovie%2F%3E%0D%0A%0D%0A%0D%0A++%3Fdir1++++m%3Adir
ector_name+%22Woody+Allen%22.%0D%0A++%3Fdir2++
+++m%3Adirector_name+%22Robert+Altman%22.%0D%0
A++%3Fdir1film+m%3Adirector+%3Fdir1%3B%0D%0A++
+++++++m%3Aactor+%3Factor.%0D%0A%0D%0A++%3F
dir2film+m%3Adirector+%3Fdir2%3B%0D%0A+++++++
++++m%3Aactor+%3Factor.%0D%0A%0D%0A++%3Factor+
+++m%3Aactor_name+%3FactorName.%0D%0A}%0D%0A&s
tylesheet=http%3A%2F%2Fwww.linkedmdb.org%2Fsn
o
rql%2Fxml-to-html.xsl
```



# Woody Allen/Robert Altman actors

SPARQL Query Results

www.linkedinmdb.org/sparql?query=PREFIX+m%3A+<http%3A%2F%2Fdata.linkedinmdb.org%2Fresource%2Fmovie%2F>%0D%

### SPARQL Query Results

actorName
"Lily Tomlin"
"Shelley Duvall"
"Michael Murphy"
"Bob Balaban"
"Sydney Pollack"
"Helen Hunt"
"Kathy Bates"
"Mia Farrow"
"Robin Williams"
"Steve Buscemi"
"Meryl Streep"
"Mark Rydell"
"John Schuck"
"Julia Roberts"
"Danny Aiello"
"Tracey Ullman"
"Kenneth Branagh"
"Famke Janssen"
"Tim Roth"
"Julie Hagerty"



# Using cURL or wget to retrieve results

```
wget -O actors.xml "big url from earlier slide"
```

```
curl -o actors.xml "url from earlier slide"
```



# Form-based SPARQL app

Find common actors between two directors - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.snee.com/sparqlforms/commonActors.html

Find common actors between two di...

## Find common actors between two directors

Enter the "official" name of each director (check [IMDB](#) if you're not sure) and click "Search" to list actors who have appeared in movies by both directors.

Woody Allen John Waters search

Find: ( Next Previous Highlight all Match case

Done



# HTML source of query form

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Find common actors between two directors</title>
    <link href="simple.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <h1>Find common actors between two directors</h1>
    <form action="twodirectors" method="get">

      <p>Enter the "official" name of each director (check <a
href="http://www.imdb.com">IMDB</a> if you're not sure) and click
"Search" to list actors who have appeared in movies by both
directors.</p>
      <p>
        <input type="text" name="dir1" />
        <input type="text" name="dir2" />
        <input type="submit" value="search" />
      </p>

    </form>
  </body>
</html>
```





# URL this will try to retrieve

`http://www.snee.com/sparqlforms/twodirectors?dir1=Woody+Allen&dir2=John+Waters`

## Can we somehow get this instead?

`http://www.linkedmdb.org/sparql?query=PREFIX+m%3A+%3Chttp%3A%2F%2Fdata.linkedmdb.org%2Fresource%2Fmovie%2F%3E%0D%0ASELECT+DISTINCT+%3FfactorName+WHERE+{%0D%0A%0D%0A+++%3Fdir1++++m%3Adirector_name+%22Woody+Allen%22.%0D%0A+++%3Fdir2++++m%3Adirector_name+%22John+Waters%22.%0D%0A+++%3Fdir1film+m%3Adirector+%3Fdir1%3B%0D%0A+++++++m%3Aactor+%3Ffactor.%0D%0A%0D%0A+++%3Fdir2film+m%3Adirector+%3Fdir2%3B%0D%0A+++++++m%3Aactor+%3Ffactor.%0D%0A%0D%0A+++%3Ffactor++++m%3Aactor_name+%3FfactorName.%0D%0A}%0D%0A&stylesheet=http%3A%2F%2Fwww.linkedmdb.org%2Fsnorql%2Fxml-to-html.xsl`



# With an .htaccess redirect

```
RewriteEngine On
```

```
ErrorDocument 404 /404page.html
```

```
redirect 301 old_filename.html new_filename.html
```

```
RewriteCond %{QUERY_STRING} dir1=(.*) & dir2=(.*)
```

```
# The rest is actually one line
```

```
RewriteRule ^.*twodirectors.*$
```

```
http://www.linkedmdb.org/sparql?query=PREFIX+m: <http://  
data.linkedmdb.org/resource/movie/>+SELECT+DISTINCT+?  
actorName+WHERE+{+?dir1+m:director_name+"%1"+?dir2+m:di  
rector_name+"%2"+?dir1film+m:director+?dir1;+m:actor+?  
actor.+?dir2film+m:director+?dir2;+m:actor+?actor.+?act  
or+m:actor_name+?actorName.+}&stylesheet=http://www.li  
nkedmdb.org/snorql/xml-to-html.xsl
```



# Entering this query...

Find common actors between two directors - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.snee.com/sparqlforms/commonActors.html

Google

## Find common actors between two directors

Enter the "official" name of each director (check [IMDB](#) if you're not sure) and click "Search" to list actors who have appeared in movies by both directors.

Woody Allen John Waters search

Find: ( Next Previous Highlight all Match case

Done



# Gets us this result.

SPARQL Query Results

actorName
"Sam Waterston"
"John Waters"
"Melanie Griffith"
"Tracey Ullman"
"Christina Ricci"



# HTML source of query form

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Find common actors between two directors</title>
    <link href="simple.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <h1>Find common actors between two directors</h1>
    <form action="commonActors.cgi" method="get">

      <p>Enter the "official" name of each director (check <a
href="http://www.imdb.com">IMDB</a> if you're not sure) and click
"Search" to list actors who have appeared in movies by both
directors.</p>
      <p>
        <input type="text" name="dir1"/>
        <input type="text" name="dir2"/>
        <input type="submit" value="search"/>
      </p>

    </form>
  </body>
</html>
```



# commonactors.cgi main() part 1

```
#!/usr/local/bin/python

import sys
# following needed for hosted version
sys.path.append('/usr/home/bobd/lib/python/')
from SPARQLWrapper import SPARQLWrapper, JSON
import string
import urllib
import cgi

def main():
    form = cgi.FieldStorage()
    dir1name = form.getvalue('dir1')
    dir2name = form.getvalue('dir2')

    sparql = SPARQLWrapper("http://data.linkedmdb.org/sparql")
```



# commonactors.cgi main() part 2

```
queryString = ""
```

```
PREFIX m: <http://data.linkedmdb.org/resource/movie/>
```

```
SELECT DISTINCT ?actorName WHERE {
```

```
    ?dir1      m:director_name "DIR1-NAME".
```

```
    ?dir2      m:director_name "DIR2-NAME".
```

```
    ?dir1film  m:director ?dir1;
```

```
                m:actor ?actor.
```

```
    ?dir2film  m:director ?dir2;
```

```
                m:actor ?actor.
```

```
    ?actor     m:actor_name ?actorName.
```

```
}
```

```
""
```



# commonactors.cgi main() part 3

```
queryString = queryString.replace("DIR1-NAME", dir1name)  
queryString = queryString.replace("DIR2-NAME", dir2name)
```

```
sparql.setQuery(queryString)  
sparql.setReturnFormat(JSON)
```

```
try:  
    ret = sparql.query()  
    results = ret.convert()  
    requestGood = True  
except Exception, e:  
    results = str(e)  
    requestGood = False
```





# commonactors.cgi main() part 4

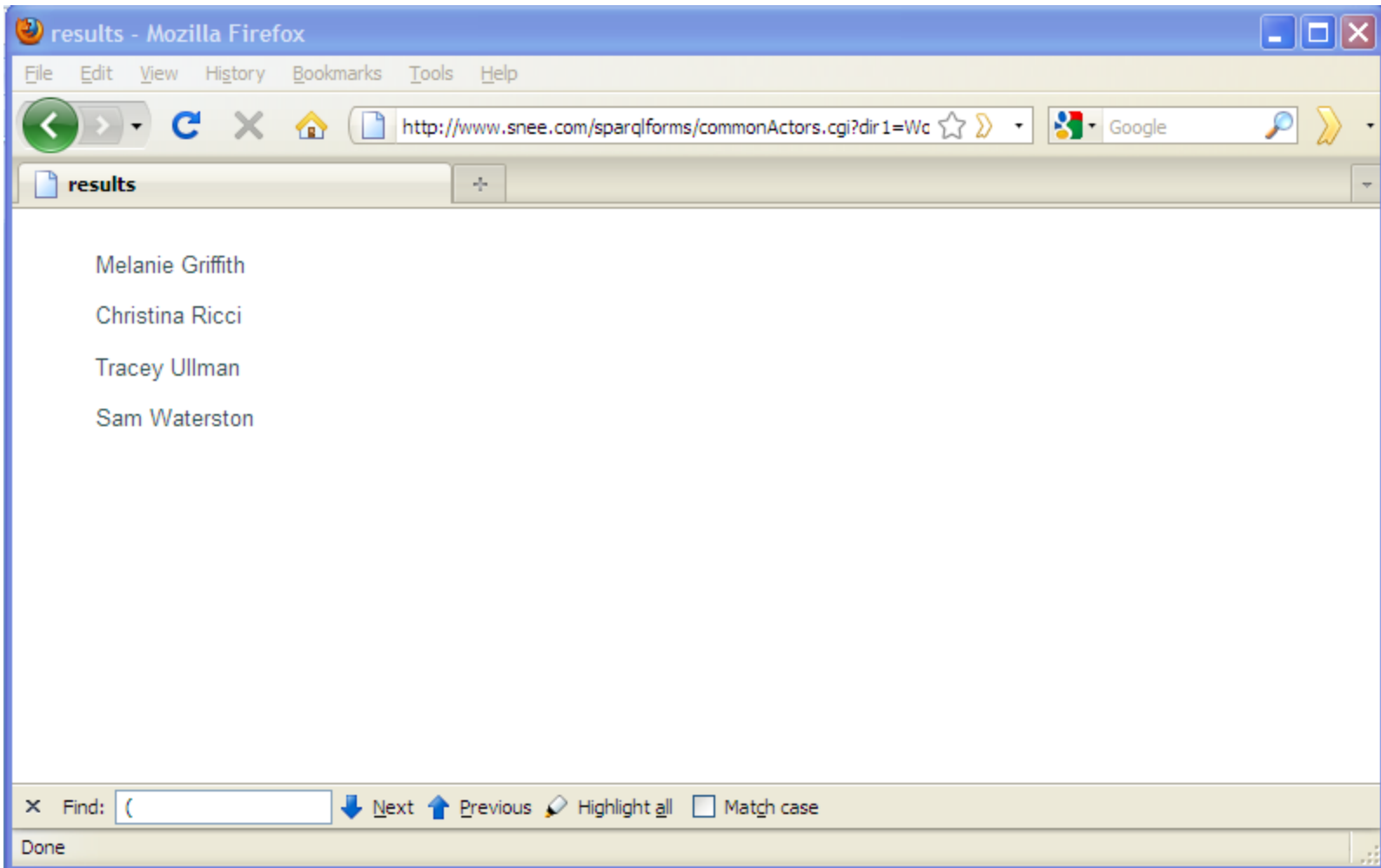
```
print """Content-type: text/html

<html>
  <head>
    <title>results</title>
    <link href="simple.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
""""

if requestGood == False:
    print "<h1>Problem communicating with the server</h1>"
    print "<p>" + results + "</p>"
elif (len(results["results"]["bindings"]) == 0):
    print "<p>No results found.</p>"

else:
    for result in results["results"]["bindings"]:
        print "<p>" + result["actorName"]["value"] + "</p>"
print "</body></html>"
```

# Form-based SPARQL app: results



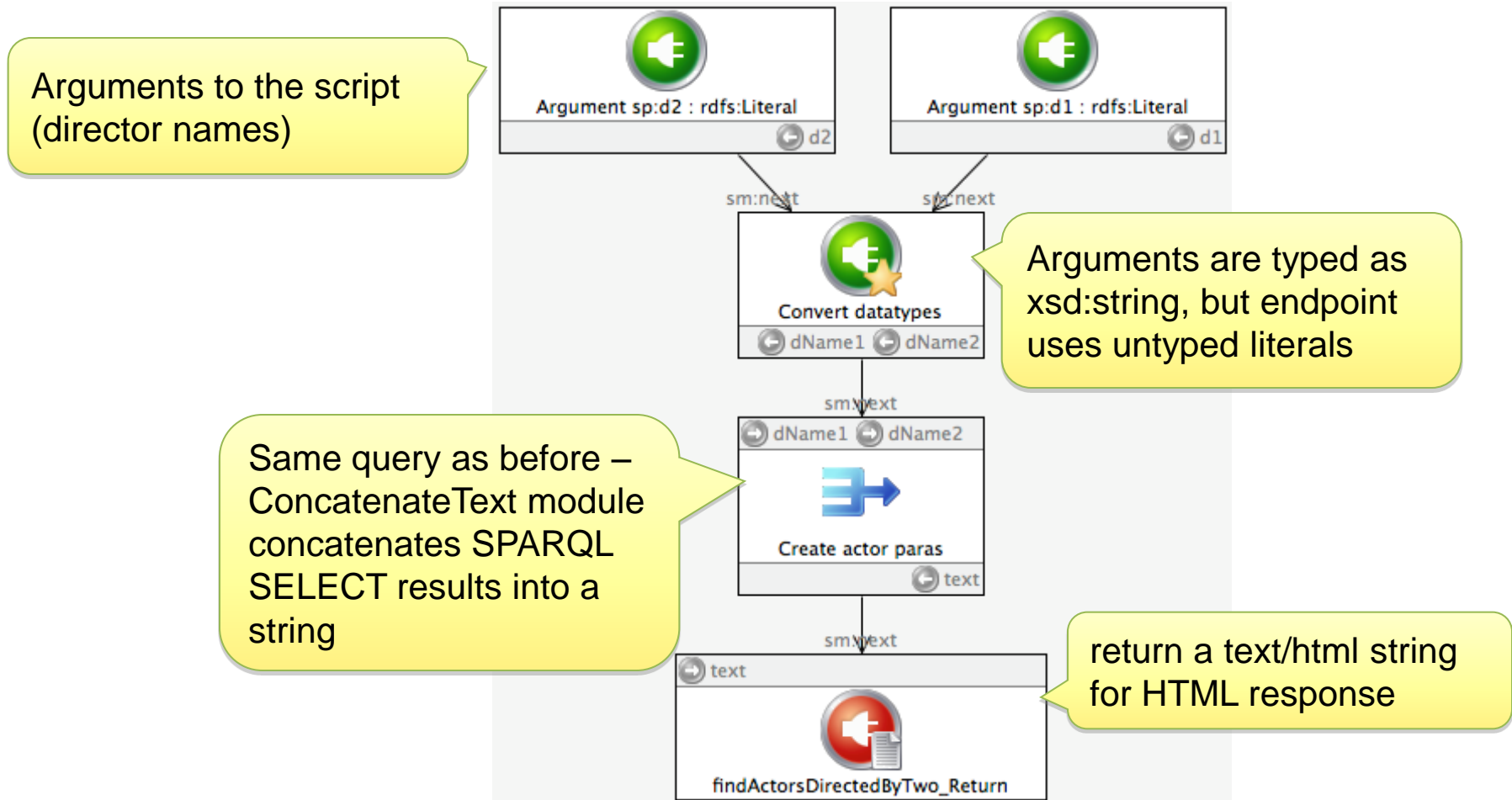
The screenshot shows a Mozilla Firefox browser window with the title "results - Mozilla Firefox". The address bar contains the URL `http://www.snee.com/sparqlforms/commonActors.cgi?dir1=Wc`. The page content displays a list of actor names:

- Melanie Griffith
- Christina Ricci
- Tracey Ullman
- Sam Waterston

At the bottom of the browser window, there is a search bar with the text "Find: (" and navigation buttons for "Next", "Previous", "Highlight all", and "Match case". The status bar at the very bottom shows "Done".

# TopBraid SPARQLMotion service

- SPARQLMotion Script: get arguments, process query, return text/html





# Calling SPARQLMotion Web Service

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Find common actors between two directors</title>
  <link href="simple.css" type="text/css" rel="stylesheet" /> </head>
<body>
  <h1>Find common actors between two directors</h1>
  <form action="http://localhost:8083/tbl/sparqlmotion">
    <p>Enter the "official" name of each director (check
    <a href="http://www.imdb.com">IMDB</a> if you're not sure)
      and click "Search" to list actors
    who have appeared in movies by both directors.</p>
    <p> <input type="text" name="d1"/>
      <input type="text" name="d2"/>
      <input type="submit" value="search"/></p>
    <input type="hidden" name="id" value="FindActorsDirectedByTwo"/>
  </form>
</body></html>
```

## Example:

http://localhost:8083/tbl/sparqlmotion?id=FindActorsDirectedByTwo&d1=John Waters&d2=Woody Allen

Live server URL

Function name

parameters

# Using SPARQL Web Pages

- SWP template

**Class Form**

Name:

Annotations

rdfs:label

Class Axioms

rdfs:subClassOf

- ui:Element

Other Properties

spin:constraint

- Argument sp:d1 : xsd:string
- Argument sp:d2 : xsd:string

ui:prototype

```

<ui:group let:dName1="{= spif:cast(?d1) }" let:dName2="{= spif:cast(?d2) }">
  <ui:forEach ui:resultSet="{#
    SELECT DISTINCT ?actorName
    WHERE {
      SERVICE <http://data.linkedmdb.org/
        ?dir1 m:director_name ?dName1 .
        ?dir2 m:director_name ?dName2 .
        ?dir1film m:director ?dir1 .
        ?dir1film m:actor ?actor .
        ?dir2film m:director ?dir2 .
        ?dir2film m:actor ?actor .
        ?actor m:actor_name ?actorName .
      } .
    } }">
    <p>{= ?actorName }</p>
  </ui:forEach>
</ui:group>
  
```

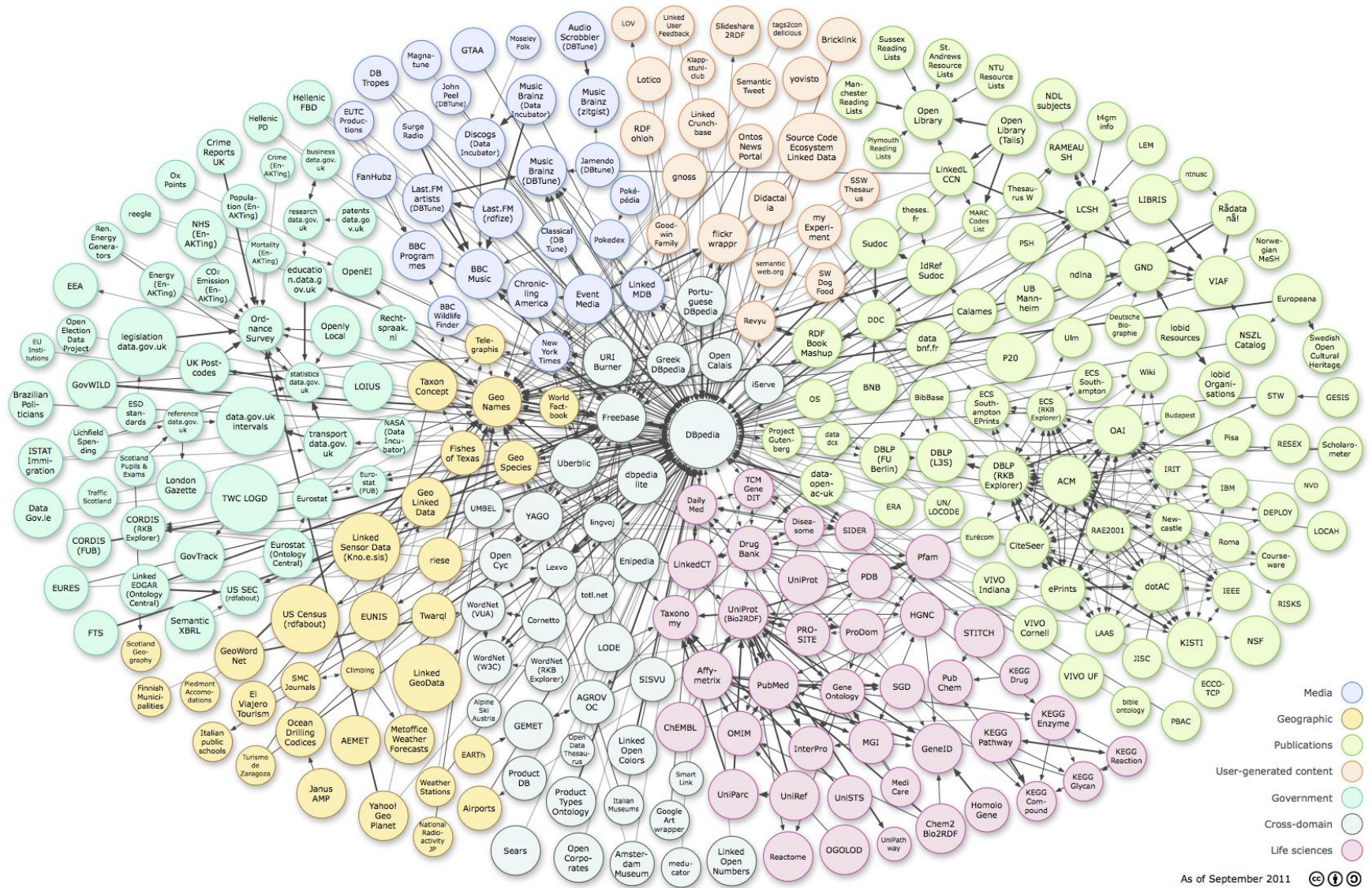
Arguments passed as SPIN constraints

ui:group is tag that isn't rendered – used to define scope for variables here

ui:forEach executes sub-tags for each query match

IMDB matches untyped literals

# Obligatory Linked Data Cloud slide



- Media (blue circle)
- Geographic (yellow circle)
- Publications (green circle)
- User-generated content (orange circle)
- Government (light green circle)
- Cross-domain (grey circle)
- Life sciences (pink circle)

# Is SPARQL Difficult?

File Edit View History Bookmarks Tools Help

http://www.cnn.com/2008/TECH/12/17/db.semanticweb/

Google

Making sense of the 'semantic Web' ...

**digitalbiz** IN ASSOCIATION WITH KONICA MINOLTA

updated 10:01 a.m. EST, Thu December 18, 2008

## Making sense of the 'semantic Web'


**STORY HIGHLIGHTS**

- The "semantic web" could make the Internet more interactive and rewarding
- Things on the Web will be described in languages that computers can understand
- An experimental kiosk lets users see the potential of this Web 3.0 application
- Widespread adoption of the semantic Web is still a long way off

[Next Article in Technology »](#)

By Steve Mollman  
For CNN

**(CNN)** -- The "semantic Web" does not sound like it's fun and easy to use, but it could make surfing Web 3.0 a more rewarding and interactive experience. Some believe it could even lead to a new form of artificial intelligence.




**COMET**  
Spotlets pop up on a 'semantic Web' kiosk run by a pair of German researchers.

The idea behind the semantic Web, very broadly, is that things on the Internet will be described with descriptor languages so that computers can "understand" what they are.

An object might be marked as a car part or a person, for instance. If objects were thus identified, an enormous network of linked data would emerge and machines, with their vast processing speeds, could suggest surprising and useful links that the human mind could never come up with, posing the possibility of a new sort of artificial intelligence.

The semantic Web is considered a key part of the upcoming "Web 3.0." It's starting to occur here and there, but widespread adoption is still a long way off.



**THIS WAY >**

**THE FUTURE IS HERE**  
Hot technology creates a whole new world.

A pair of German researchers have created an experimental kiosk that lets you easily use semantic Web capabilities -- even if you have no idea what they are. All that is needed is an iPhone and a finger with which to drag icons around on the kiosk's touch screen.

Find:  Next Previous Highlight all Match case

Done



# Is SPARQL Difficult?

“Consider, for instance, SPARQL, a query language. To find, say, music artists associated with the producer Timbaland, you'd have to type a long piece of convoluted code that most of us wouldn't bother to do.”





# Artists Timbaland produced

SPARQL Explorer for http://dbpedia.org/sparql

```
SPARQL:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX d: <http://dbpedia.org/ontology/>

SELECT DISTINCT ?artist
WHERE {
  ?album d:producer :Timbaland ;
         d:musicalArtist ?artist .
}
```

Results:

SPARQL results:

artist
<a href="#">:Chris_Cornell</a>
<a href="#">:Flo_Rida</a>
<a href="#">:Sebastian_%28rapper%29</a>
<a href="#">:M._Pokora</a>
<a href="#">:Bj%C3%B6rk</a>
<a href="#">:Ginuwine</a>
<a href="#">:Nelly_Furtado</a>
<a href="#">:Missy_Elliott</a>



# On the other hand...

Some JavaScript from a View Source of that same CNN page:

```
if(cnnWinExtraRegExp.test(cnnWinExtra)){var cnnOmniExtra =
cnnWinExtraRegExp.split(cnnWinExtra);cnnWinLoc = cnnWinLoc +
cnnOmniExtra[0];} else {cnnWinLoc = cnnWinLoc +
cnnWinExtra;}} if (typeof(cnnPageName) != "undefined")
{s.pageName = cnnPageName;s.eVar1 = cnnPageName;} else
{s.pageName = cnnWinLoc;s.eVar1 = cnnWinLoc;} if
(typeof(cnnSectionName) != "undefined")
{s.channel=cnnSectionName;s.eVar2=cnnSectionName;} else
{s.channel="Nonlabeled";s.eVar2="Nonlabeled";} if
(typeof(cnnSubSectionName) != "undefined")
{s.server=cnnSubSectionName;s.eVar3=cnnSubSectionName;} else
{s.server="";s.eVar3="";} if (typeof(cnnSectionFront) !=
"undefined") {s.prop1=cnnSectionFront;} if
(typeof(cnnContentType) != "undefined")
{s.prop4=cnnContentType;s.prop6=s.pageName;}
```



# Summary

- Simple web forms can send parameterized SPARQL queries to SPARQL endpoints:
  - Using .htaccess files
  - With CGI scripts
  - Using REST web services as shown by TopBraid SPARQLMotion script
  - Embedding SPARQL queries into HTML markup as shown by SPARQL Web Pages with TopBraid



# Thank you!



**TopBraid Suite™**



**TopBraid**  
Enterprise Vocabulary Net™

*“A wonderful harmony is created when we  
join together the seemingly unconnected.”*

- Heraclitus

**Woody Allen picture:**

**<http://www.flickr.com/photos/thomasthomas/504401865/> by Thomas**

**Thomas some rights reserved; James Cameron picture**

**<http://www.flickr.com/photos/jurvetson/4357736383> by Steve Jurvetson**

**some rights reserved**

**[bducharme@topquadrant.com](mailto:bducharme@topquadrant.com)**

**[info@topquadrant.com](mailto:info@topquadrant.com)**