HowtoQueryFromWebBrowser

Howto Run SQL Queries From Your Browser

- TARGET-AUDIENCE: ***beginner*** advanced expert
- COCOON-RELEASES: 2.0.3, 2.0.4
- DOCUMENT-STATUS: ***draft*** reviewed released

What you will get from this page

Simple instructions describing how to connect your browser to your SQL database. You will then be able to execute SQL commands in your browser's location bar, like this:

```
http://localhost:8080/cocoon/~pete/sql?stmt=select * from user
```

or:

```
http://localhost:8080/cocoon/~pete/sql?stmt=insert into user values(NULL,"pete","pete@foo.com")
```

or, more dangerously:

```
http://localhost:8080/cocoon/~pete/sql?stmt=delete from user where uid="pete"
```

or even:⚠️

```
http://localhost:8080/cocoon/~pete/sql?stmt=drop table user
```

...which is why I need to warn you right now:

Warning! do not use this on a production system, or a system exposed to the public Internet, etc.

This howto example severly compromises system security. Kids, don't try this at home, don't run with scissors, don't do drugs, wear a rubber, etc.

So why might you want to do this?

Well, if you're a developer, or a Cocoon user, you might want to use this as a handy SQL client. If you're neither, then you probably shouldn't be reading this. This technique might amuse your audience during software demo presentations, or provide a very minor amount of help with development, but it will not win you a Nobel Prize, and it is unlikely to enhance your allure to exotic women. File this under "Stupid Cocoon Tricks" or "Worst Practices" and go outside and play instead!

Your basic skills

Modest familiarity with trivial Cocoon applications.

Technical prerequisites

Database connected to Cocoon.

Links to other information sources

SpecificDatabaseConnection

Overview

Here are the steps we will follow:

- Step 1: add the `ServerPagesGenerator` component to your `sitemap.xmap`
- Step 2: add a matcher snippet to your `<pipeline>` to match your SQL requests
- Step 3: create an `.xsp` file to generate the XML from the database query

- Step 4: optionally, create an `.xsl` stylesheet to transform the generated XML output to a flatter format, which may be better suited to your personal and private wants and needs

Step 1 - Add the ServerPagesGenerator to your sitemap.xmap

Add this to your `sitemap.xmap`:

```
<map:components>
 <map:generators>
  <map:generator
       label="content,data"
       logger="sitemap.generator.serverpages"
       name="xsp"
       pool-grow="2"
       pool-max="32"
       pool-min="4"
       src="org.apache.cocoon.generation.ServerPagesGenerator"/>
 </map:generators>
</map:components>
```

Step 2: Add a matcher to your sitemap.xmap

Add this to your `<pipeline>`:

```
  <map:match pattern="sql">
   <map:generate type="xsp" src="sql.xsp">
    <map:parameter name="use-request-parameters" value="true"/>
   </map:generate>
   <map:serialize type="xml"/>
  </map:match>
```

Step 3: Create sql.xsp

Copy and save as `sql.xsp`: (be sure to substitute the name of your pool, as configured in `cocoon.xconf` in the `<esql:pool>` tag below)

```
<?xml version="1.0" encoding="iso-8859-1"?>

<xsp:page
     language="java"
     xmlns:xsp="http://apache.org/xsp"
     xmlns:esql="http://apache.org/cocoon/SQL/v2"
     xmlns:xsp-request="http://apache.org/xsp/request/2.0">

<resultset>
 <esql:connection>
  <esql:pool>mydevpool</esql:pool> <!-- your pool here -->
  <esql:execute-query>
   <esql:query>
    <xsp-request:get-parameter name="stmt"/>
   </esql:query>
   <esql:results>
    <esql:row-results>
     <xsp:element name="result">
      <esql:get-columns/>
     </xsp:element>
    </esql:row-results>
   </esql:results>
   <esql:no-results/>
   <esql:error-results>
    <xsp:element name="result">
     <esql:get-message/>
     <esql:get-stacktrace/>
     <esql:to-string/>
    </xsp:element>
   </esql:error-results>
   <esql:update-results>
    <xsp:element name="result">
     <update>
     <esql:get-update-count/>
     </update>
    </xsp:element>
   </esql:update-results>
  </esql:execute-query>
 </esql:connection>
</resultset>
</xsp:page>
```

Test

Now you should be able to execute an SQL statement on your database by typing the statement in the location bar of your browser like this:

```
http://path_to_your_db_host:your_port/path_to_your_cocoon_work_dir/sql?stmt=select * from user
```

Of course, whether this works or not depends on whether you have a table called 'user' in the db specified by your `<datasource>` (defined in `cocoon.xconf`) If it works, you should see something like:

```
<resultset>
 <result>
  <recnum>1</recnum>
  <uid>pete</uid>
  <name>Pete Farmer</name>
 </result>
 <result>
  <recnum>2</recnum>
  <uid>bob</uid>
  <name>Robert Vesco</name>
 </result>
 <result>
  <recnum>3</recnum>
  <uid>sonny</uid>
  <name>Sonny Barger</name>
 </result>
</resultset>
```

If it doesn't work, you may want to make sure that your database is configured properly to sing and dance merrily with Cocoon. See SpecificDatabaseConnection

Step 4: (Optional) Post-process the results

You may want to flatten your `<resultset>`, and collapse the XML elements returned by the XSP generator into attributes of the `<result>` element. This may be convienient if your are using other products in your software stack, or whatever. Otherwise, you're done. Go outside and ride your skateboard!

If you've decided you're not done, then add this line to the matcher `<map:match pattern="sql">` in your `sitemap.xmap`:

```
    <map:transform src="sql.xsl"/>
```

so that it now looks like this:

```
   <map:match pattern="sql">
    <map:generate type="xsp" src="sql.xsp">
     <map:parameter name="use-request-parameters" value="true"/>
    </map:generate>
    <map:transform src="sql.xsl"/>
    <map:serialize type="xml"/>
   </map:match>
```

And create the file `sql.xsl`:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:strip-space elements="*"/>
 <xsl:template match="*">
  <xsl:copy>
   <xsl:for-each select="@*|*[not(* or @*)]">
    <xsl:attribute name="{name(.)}"><xsl:value-of select="."/>
    </xsl:attribute>
   </xsl:for-each>
   <xsl:apply-templates select="*[* or @*]|text()"/>
  </xsl:copy>
 </xsl:template>
</xsl:stylesheet>
```

Now when you run your query your results will look like this:

```
<resultset>
 <result recnum="1" uid="pete" name="Pete Farmer"/>
 <result recnum="2" uid="bob" name="Robert Vesco"/>
 <result recnum="3" uid="sonny" name="Sonny Barger"/>
</resultset>
```

which, depending on your application, may be more useful.

The End

With a few drops of the magical Cocoon krazy glue, you have now "glued" your browser to your db. Enjoy. That's all. Finito. What do you want for nothing?

---

page metadata

- AUTHOR:PeteFarmerBR
- AUTHOR-CONTACT: pete.farmer@gmail.comBR
- REVIEWED-BY:BR
- REVIEWER-CONTACT:BR