Matlab Guide

Version: *0.2*
Date: *20.05.2012*
Status: *Valid*
Author: *A. Hueni & D. Kuekenbrink, Remote Sensing Laboratories, University of Zurich*
File: *\SPECCHIO_Matlab_Guide_V0.1.0.docx*
Pages: *13*

Classification:
Distribution: *SPECCHIO Users*

History

Version	Date	Author	Remark
0.1.0	18.05.2012	A. Hueni & D. Kueken-brink	First version
0.2.0	20.05.2012	A. Hueni & D. Kueken-brink	Added static classpath definition and a metadata extraction example.

Table of Contents

1 Introduction

The Matlab environment is a well-established tool in engineering, research and science. Matlab includes a Java Virtual Machine and thus allows the use of Java classes within Matlab code. Starting from version 2.1.2, SPECCHIO includes new methods that allow the easy integration of SPECCHIO classes, giving easy access to the spectral data and metadata stored in SPECCHIO databases.

2 Installation and Configuration

2.1 Setup

2.1.1 Static Definition of the Java Classpath

In the Matlab prompt type: edit classpath.txt
Add the following lines to your classpath.txt file (change the path to reflect your installation location):

```
<your_installation_path>/SPECCHIO_APP_V2.2.0/SPECCHIO_APP_V2.2.0.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/mysql-connector-java-5.1.20-
bin.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jcommon-1.0.5.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jgraph.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/ganymed-ssh2-build251beta1.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jfreechart-1.0.2.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jhdf.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jhdf4obj.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jhdf5.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jhdf5obj.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jhdfobj.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/jsch-0.1.44.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/qcchart3djava.jar
<your_installation_path>/SPECCHIO_APP_V2.2.0/ujump-complete-0.2.5.jar
```

Note: restart Matlab for the changes to take effect.

2.1.2 Non-static Definition of the Java Classpath

Alternatively you can you can define your Java-classpath in the following way:
Create a new m-file in Matlab. First you need to declare the directories of the classes of the SPECCHIO distribution. For that, type in `javaaddpath ({})`
In between the curly brackets, you need to declare the directories of the following jar-files:

```
SPECCHIO_APP_V2.2.0.jar
mysql-connector-java-5.1.20-bin.jar
jcommon-1.0.5.jar
jgraph.jar
ganymed-ssh2-build251beta1.jar
jfreechart-1.0.2.jar
jhdf.jar
jhdf4obj.jar
jhdf5.jar
jhdf5obj.jar
jhdfobj.jar
jsch-0.1.44.jar
qcchart3djava.jar
ujump-complete-0.2.5.jar
```

The code should look like this:

```
javaaddpath
({'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/SPECCHIO_App_V2.2.0.
jar', '/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/mysql-connector-
```

```
java-5.1.20-bin.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jcommon-1.0.5.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jgraph.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/ganymed-ssh2-
build251beta1.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jfreechart-1.0.2.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jhdf.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jhdf4obj.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jhdf5.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jhdf5obj.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jhdfobj.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/jsch-0.1.44.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/gcchart3djava.jar',
'/Users/dkuekenb/Documents/SPECCHIO_MATLAB/SPECCHIO_App_V2/ujump-complete-
0.2.5.jar'})
```

2.1.3 Importing the SPECCHIO Packages

Afterwards you need to import several packages, so that MATLAB knows all the java classes and Methods needed to run the SPECCHIO application. The code for that looks like this:

```
import specchio.*
import eav_db.*
import processors.*
import specchio_proc_modules.*
```

2.2 Connecting to the SPECCHIO Database

Use the `DatabaseConnection` class to open a connection with a SPECCHIO database instance:

```
con = DatabaseConnection.getInstance();
```

This call will try to get the known database connections from the SPECCHIO db_config.txt file and connect to the first entry within this file automatically. Make sure that the db_config.txt file is within the current directory.
To manually change the database connection, use the `connection_details_class(String server, String db_name, String port, String user, String password)` class and the set_connection_parameters(conn_det) method in the following way:

```
conn_det = connection_details_class('specchio-pub.geo.uzh.ch',
'specchio21', '3306', 'JDoe', ' XXXXXXXX ');

con.set_connection_parameters(conn_det);
```

The `DatabaseConnection` class is using the Singleton pattern (Gamma, Helm et al. 1997). This implies that once the connection is set, all SPECCHIO objects connecting to the database automatically utilise this connection.

2.3 Preparing Queries

SQL select queries are used to select spectral data from the database. These can be defined manually, but the easiest way is to generate queries using the SPECCHIO Query Builder and use these in Matlab.

2.3.1.1 Using the Query Builder to form Statements

Start the regular SPECCHIO Java Application, open the Query Builder and select the required data. Then open the pop-up menu of the SQL auto-built query panel by clicking the menu mouse button (Figure 1).

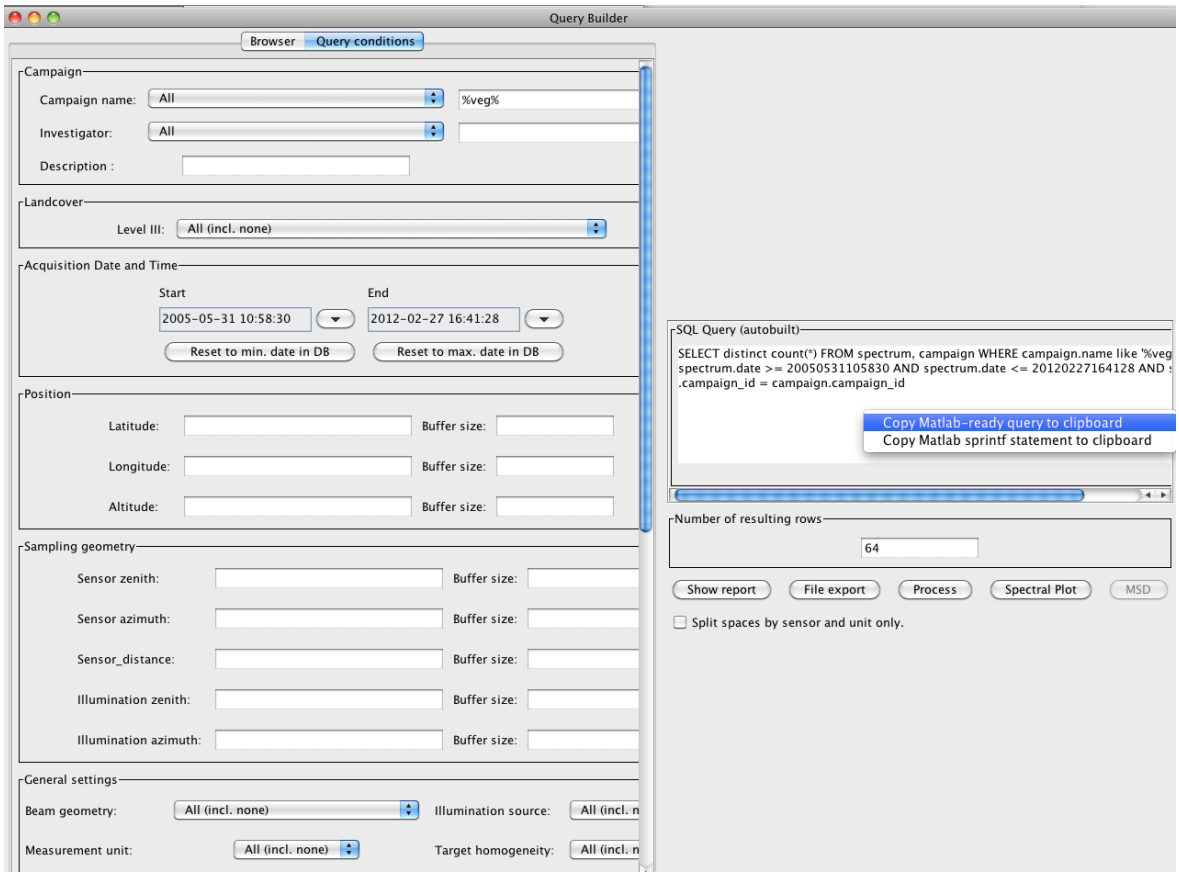

Figure 1: Matlab query pop-up menu in the Query Builder

Two options for query copying are available. The first returns a string ready for the inclusion in a sprintf statement in Matlab while the second returns the full sprintf statement:

```
'SELECT spectrum.spectrum_id FROM spectrum, campaign WHERE campaign.name
like ''%%veg%%'' AND spectrum.date >= 20050531105830 AND spectrum.date <=
20120227164128 AND spectrum.campaign_id = campaign.campaign_id'

query = sprintf('SELECT spectrum.spectrum_id FROM spectrum, campaign
WHERE campaign.name like ''%%veg%%'' AND spectrum.date >= 20050531105830
AND spectrum.date <= 20120227164128 AND spectrum.campaign_id = cam-
paign.campaign_id')
```

The queries are automatically formatted for Matlab to allow the inclusion of single quotes and percentage signs in the query strings.

2.3.1.2 Calling the Query Builder from Matlab

Query statements can be returned from the Query Builder into Matlab by starting the Query Builder from Matlab, building the query and using a Matlab modal dialog to return the query to Matlab.

```
query = getquery('Title','Get Query');
```

The above call starts the getquery modal dialog, which in turn brings up the Query Builder. Once the query is formed, press the ‘Go’ button in the modal dialog (Figure 2).

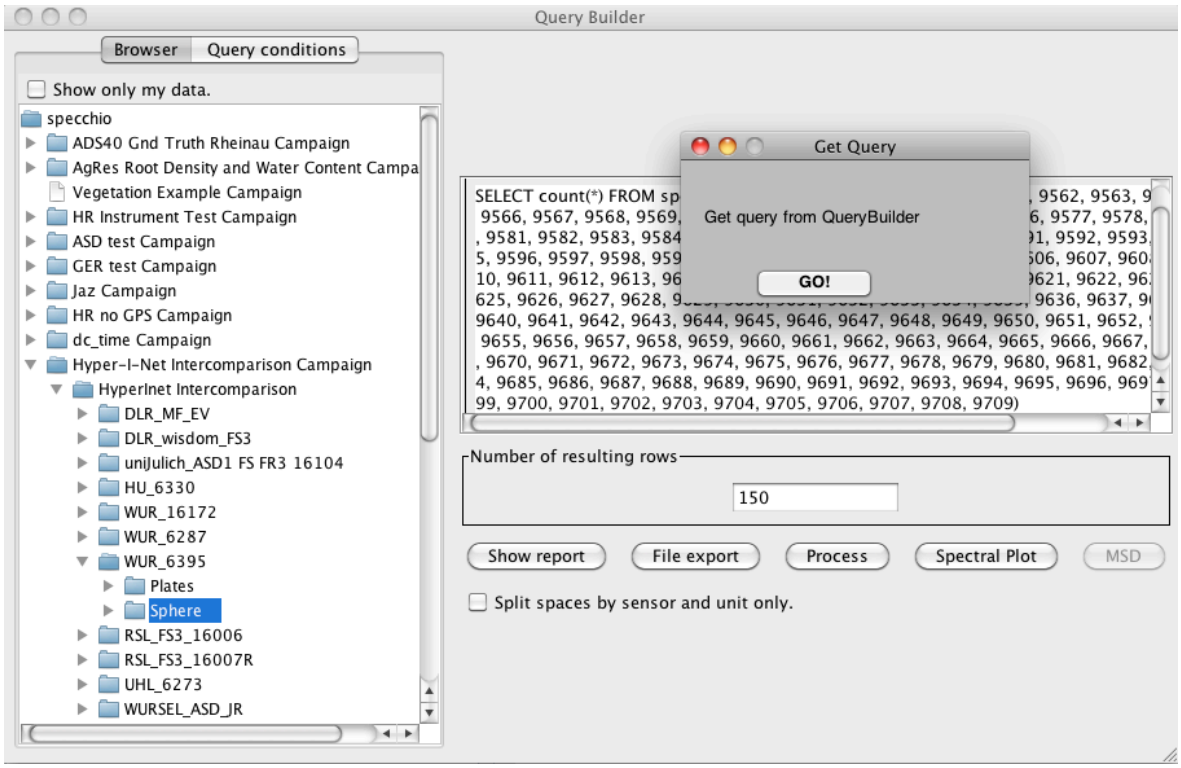

Figure 2: Query Browser and Get Query modal dialog

The returned query is a Java string object, i.e. no special formatting of the string in Matlab is required.
The getquery Matlab code is included in each SPECCHIO application.

2.4 Retrieving Spectral Data

Data is retrieved from the database by the following steps:

Instantiate a `QueryBuilderBaseClass` object:

```
qbb=QueryBuilderBaseClass('spectrum');
```

Set the prepared query string (see 2.3 on how to create queries):

```
qbb.setSelect_query(query);
```

Get the list of spectrum_id's:

```
ids = qbb.get_spectrum_ids();
```

Get a `SpaceFactory` object:

```
sf = SpaceFactory.getInstance();
```

Create spaces based on the selected spectrum ids. The SpaceFactory returns a Java Arraylist of spaces. For more information on the building of spaces refer to the documentation of the Space concept in the SPECCHIO User Guide.

```
spaces=sf.create_spaces(ids);
```

Get a space from the space Arraylist. A 'for' or 'while' loop can be used to get all returned spaces.

```
spaces_li = spaces.listIterator();

space = spaces_li.next();
```

Load the spectral data into the space:

```
space.load_data();
```

Return the spectral data as a Matlab array:

```
vectors = space.get_array();
```

Get the wavelength information from the space:

```
wvl = space.get_wvls();
```

3 Examples

3.1 Function to get Spectral Data from SPECCHIO

This function is included in the SPECCHIO application distribution.

```
function spectral_data=get_spectral_data_from_specchio(query)

    import specchio.*

    qbb=QueryBuilderBaseClass();

    qbb.setSelect_query(query);

    ids = qbb.get_spectrum_ids();

    sf = specchio.SpaceFactory.getInstance();

    spaces=sf. create_spaces(ids);

    spaces_li = spaces.listIterator();

    space = spaces_li.next();

    space.load_data();

    spectral_data.vectors = space.get_array();
    spectral_data.wvl = space.get_wvls();
```

```
end
```

3.2 Retrieving Data based on the Spectrum File Names and Calculating Means

This example connects to SPECCHIO, retrieves spectra of a certain campaign that match defined spectrum file names and calculates the mean of each returned set.

```
import specchio.*
con = DatabaseConnection.getInstance();

con.set_connection_parameters('specchio.geo.uzh.ch', 'specchio', '3306',
'JDoe', 'XXXXXX') %(String server, String db_name, String port, String
user, String password)


spectra_basenames = {'ND2'  '5'  '8'  '10'  '16'  '20'  '31'  '40'  '63'
'80'  '100'};



for i=1: length(spectra_basenames)

    query = sprintf( 'SELECT spectrum.spectrum_id FROM spectrum, campaign
WHERE campaign.name = ''ASD_Calibration_04.2010'' AND spec-
trum.measurement_unit_id = ''2'' AND spectrum.file_name like
''ASD2_%s.%%''', char(spectra_basenames(i)));


    spectral_data = get_spectral_data_from_specchio(query);

    mean_spectra(i, :) = mean(spectral_data.vectors);

end

figure
plot(spectral_data.wvl, mean_spectra);
title('Small sphere levels');
legend(spectra_basenames);
```

3.3 Get Metadata of Spectra

This example selects data held by specchio@db.specchio.ch and retrieves their metadata. Metadata are printed for each spectrum. The second part of the function searches for specific metaparameters (lat/lon) and returns them as numeric values for plotting.

```
function get_metadata()


    import specchio.*
    import eav_db.*
    DatabaseConnection.getInstance();
```

```
    % query to get data from specchio@db.specchio.ch
    query = sprintf('SELECT spectrum.spectrum_id FROM spectrum, campaign
WHERE campaign.name = ''Tongariro NP Plants'' AND spectrum.date >=
20050531105830 AND spectrum.date <= 20120220142046 AND spectrum.file_name
like ''draco%%'' AND spectrum.campaign_id = campaign.campaign_id');


    qbb=QueryBuilderBaseClass('spectrum')

    qbb.setSelect_query(query);

    ids = qbb.get_spectrum_ids();


    % get metadata for these id's
    li = ids.listIterator();
    while(li.hasNext())

        spectrum = Spectrum(li.next());

        metadata = spectrum.md_attribute_list;

        % loop over all metadata
        md_li = metadata.listIterator();
        while(md_li.hasNext())

            metaparameter = md_li.next();

            disp([char(metaparameter.name) ' : '
char(metaparameter.toString())]);

        end

        disp('--------------------------------'); % separate the metadata
of each spectrum


    end


    % get lat/lon for these id's as numeric values
    lat = zeros(ids.size(), 1);
    lon = zeros(ids.size(), 1);
    cnt = 1;
    li = ids.listIterator();
    while(li.hasNext())

        spectrum = Spectrum(li.next());

        metadata = spectrum.md_attribute_list;

        % loop over all metadata
        md_li = metadata.listIterator();
        while(md_li.hasNext())

            metaparameter = md_li.next();
```

```
            if(metaparameter.name.equals('Latitude'))
                lat(cnt) = metaparameter.get_value();
            end

            if(metaparameter.name.equals('Longitude'))
                lon(cnt) = metaparameter.get_value();
            end

        end

        cnt = cnt + 1;

    end

    % plot locations
    figure
    plot(lat, lon, '*');


end
```

Expected output for each spectrum:

Number : 8
Comment :
Capture date : 2005.07.01 11:41:36
Loading date : 2009.07.05 17:09:46
Filename : dracoph.008
Internal no of avg : 10
Is reference : false
Latitude : -39.32328167
Longitude : -175.49934667
Altitude : 1270.5
Location :
Campaign name : Tongariro NP Plants
Campaign desc : New Zealand native plants of Tongariro National Park.
Landcover : Moors and heathland
Cloud cover [octas] :
Ambient temp. [□C] :
Air pressure :
Rel. humidity :
Wind direction :
Wind speed :
Sensor zenith :
Sensor azimuth :
Illumination zenith :
Illumination azimuth :
Sensor distance :
Illumination distance :
Measurement unit : Reflectance
Beam geometry :
Illumination source :
Sampling environment :

Spectrum names :
Target types :

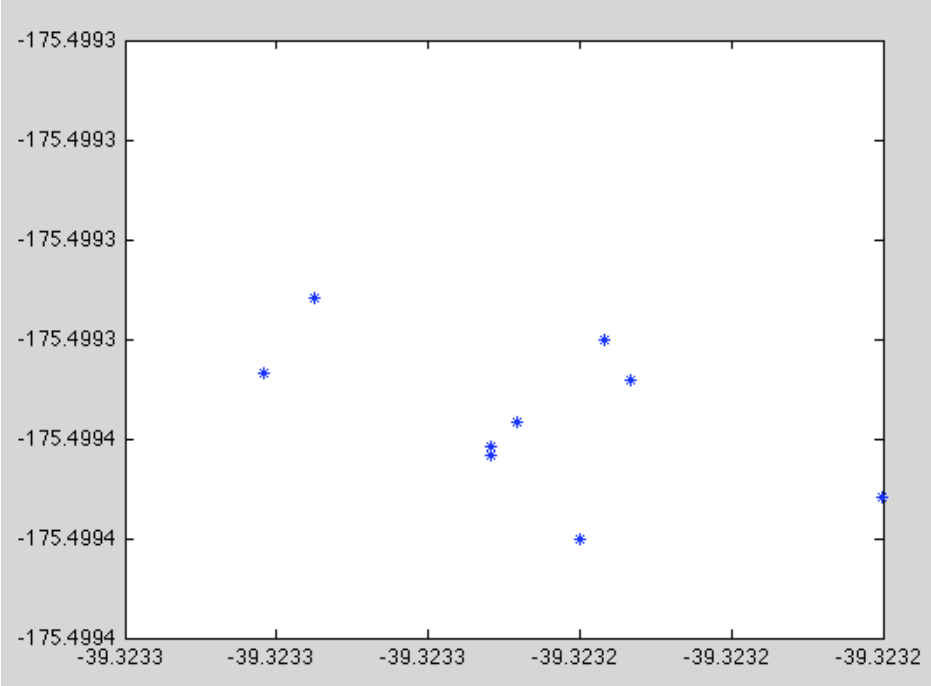

Figure 3: lat-lon plot