

# Two Minute Tutorial

## Table of contents

1 Installation.....	2
2 Use Scriptella as Ant Task.....	2
3 Command-Line Execution.....	2
4 Executing ETL Files from Java.....	2
5 Examples.....	3
5.1 Copy table to another database.....	3
5.2 Working with BLOBs.....	4
5.3 Supporting several SQL dialects.....	5

## 1. Installation

- [Download](#) Scriptella binary distribution.
- Unpack it and add a <SCRIPTELLA\_DIR>/bin to a system PATH variable.  
Use `set PATH=%PATH%;SCRIPTELLA_DIR\bin` for Windows and `export PATH=${PATH}:SCRIPTELLA_DIR/bin` for Unix.
- Check if JRE has been installed correctly by running `java -version`.
- *Optional step:* Put JDBC drivers required by your scripts to <SCRIPTELLA\_DIR>/lib directory or directly specify classpath attribute in script connection elements.

## 2. Use Scriptella as Ant Task

In order to use Scriptella as Ant task you will need the following taskdef declaration:

```
<taskdef resource="antscriptella.properties"
classpath="/path/to/scriptella.jar[;additional_drivers.jar]"/>
```

### Note:

Additional drivers classpath entries are optional. You may directly specify classpath attribute in a connection XML element declaration. Example:

```
<connection driver="sybase" classpath="jconn3.jar"/>
```

Running Scriptella files from Ant is simple:

```
<etl/> <!-- Execute etl.xml file in the current directory -->
```

or

```
<etl file="path/to/your/file/> <!-- Execute ETL file from specified
location -->
```

## 3. Command-Line Execution

Just type `scriptella` to run the file named `etl.xml` in the current directory. Alternatively you can use java launcher:

```
java -jar scriptella.jar [arguments]
```

## 4. Executing ETL Files from Java

It is extremely easy to run Scriptella ETL files from java code. Just make sure `scriptella.jar` is on classpath and use any of the following methods to execute an ETL file:

```
EtlExecutor.newExecutor(new File("etl.xml")).execute(); //Execute etl.xml
file
EtlExecutor.newExecutor(getClass().getResource("etl.xml")).execute();
//Execute etl.xml file loaded from classpath
```

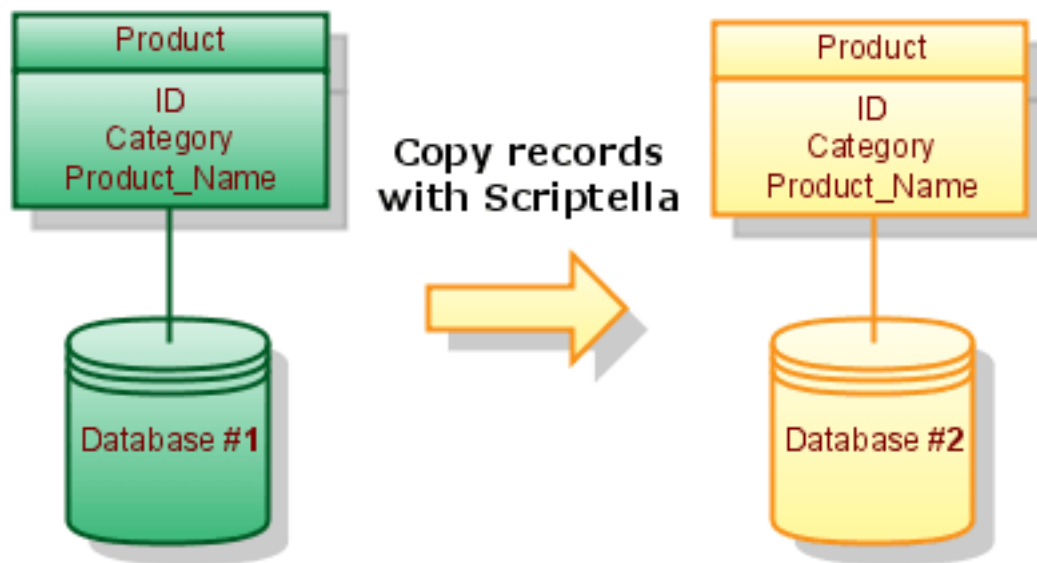
```
EtlExecutor.newExecutor(  
    servletContext.getResource("/WEB-INF/db/init.etl.xml")).execute();  
//Execute init.etl.xml file from web application WEB-INF directory
```

See [EtlExecutor Javadoc](#) for more details on how to execute ETL files from Java code.

## 5. Examples

For a quick start type `scriptella -t` to create a template `etl.xml` file.

### 5.1. Copy table to another database



#### Migration of table records

Assume Database #1 contains Table Product with id, category and name columns. The following script copies software products from this table to Database #2. Additionally Name column is changed to Product\_Name.

etl.xml:

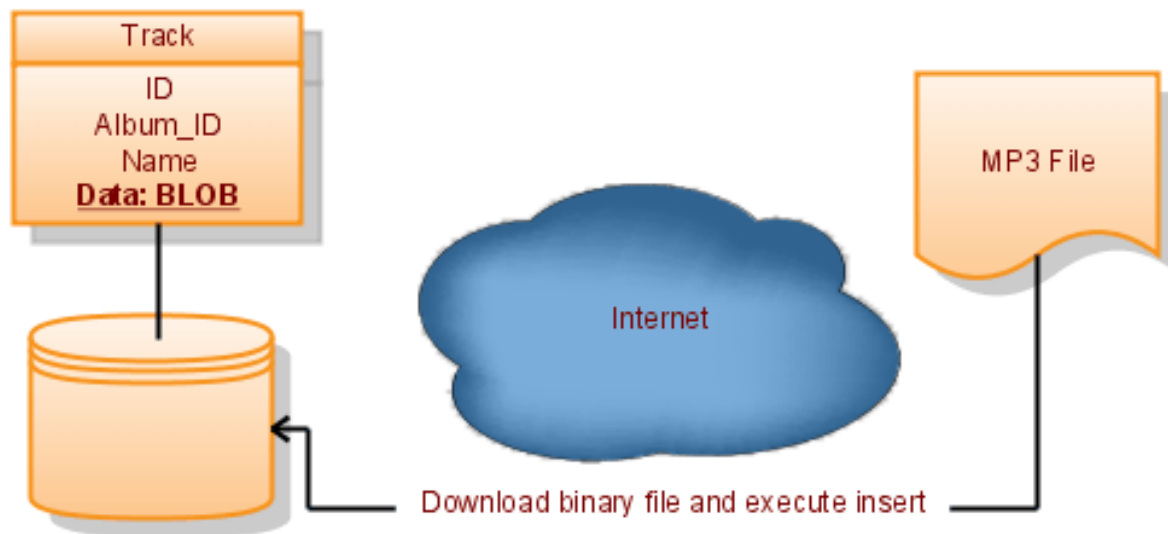
```
<etl>  
  <connection id="db1" url="jdbc:database1:sample" user="sa"  
password="" classpath="external.jar"/>  
  <connection id="db2" url="jdbc:database2:sample" user="sa"  
password=""/>  
  <query connection-id="db1">  
    <!-- Select product from software category in db1-->  
    SELECT * FROM Product WHERE category='software';
```

```

        <!-- for each row execute a script -->
        <script connection-id="db2">
            <!-- Insert all selected products to db2
            use ? to reference properties, columns or
?{expressions}-->
            INSERT INTO Product(id, category, product_name) values
(?id, ?{category}, ?name);
        </script>
    </query>
</etl>

```

## 5.2. Working with BLOBs



### Inserting BLOB content from URL

The following sample initializes table of music tracks. Each track has a DATA field containing a file loaded from an external location. File song1.mp3 is stored in the same directory as etl.xml and song2.mp3 is loaded through the web.

etl.xml:

```

<etl>
  <connection url="jdbc:hsqldb:file:tracks" user="sa" password="" />
  <script>
    CREATE TABLE Track (
      ID INT,
      ALBUM_ID INT,
      NAME VARCHAR(100),
      DATA LONGVARBINARY
    );
  </script>
</etl>

```

```

location -->
    <!-- Inserts file with path relative to executed script
    insert into Track(id, album_id, name, data) values
        (1, 1, 'Song1.mp3', ?{file 'song1.mp3'});
    <!-- Inserts file from URL-->
    insert into Track(id, album_id, name, data) values
        (2, 2, 'Song2.mp3', ?{file
'http://musicstoresample.com/song2.mp3'});
    </script>
</etl>

```

### 5.3. Supporting several SQL dialects

<dialect> element allows including vendor specific content. The following example creates database schema for Oracle/HSQLDB or MySql depending on a selected driver:

```

<etl>
  <properties><!-- Load external properties -->
    <include href="etl.properties"/>
  </properties>
  <connection url="$url" user="$user"
    password="$password" classpath="$classpath"/>
  <script>
    -- In this example dialects are used to
    -- include different DDLs for data types:
    -- Example: oracle-schema.sql for Oracle
    <dialect name="hsqldb"> <!-- Regular expressions syntax
supported-->
      <include href="hsqldb-schema.sql"/>
    </dialect>
    <dialect name="oracle">
      <include href="oracle-schema.sql"/>
    </dialect>
    <dialect name="mysql">
      <include href="mysql-schema.sql"/>
    </dialect>
    -- SQL92 inserts - no need to use dialects
    INSERT INTO Product(id, category, product_name)
      VALUES (1, 'ETL', 'Scriptella ETL');
    INSERT INTO Product(id, category, product_name)
      VALUES (2, 'Development', 'Java SE 6');
  </script>
</etl>

```

#### Note:

For additional samples [download](#) Scriptella examples distribution. See also [Scriptella examples on DZone Snippets](#)