@Translator(name = "interceptor", description = "Translator Interceptor")

**public** **class** DelegateExecutionFactory **extends** BaseDelegatingExecutionFactory<Object, Object> {

@Override

**public** ResultSetExecution createResultSetExecution(QueryExpression command, ExecutionContext executionContext,

RuntimeMetadata metadata, Object connection) **throws** TranslatorException {

*LOGGER*.info("inside createResultSetExecution method BaseDelegatingExecutionFactory ");

BaseResultSetExecution baseExecution = **new** BaseResultSetExecution(command, connection, executionContext,

metadata);

baseExecution.setResultSetExecution(**super**.createResultSetExecution(command, executionContext, metadata,

connection));

**return** baseExecution;

}

}

**public** **class** BaseResultSetExecution **implements** ResultSetExecution {

**public** BaseResultSetExecution(QueryExpression command, Object connection, ExecutionContext context,

RuntimeMetadata metadata) {

**this**.setCommand(command);

**this**.setConnection(connection);

**this**.setMetadata(metadata);

**this**.setCommandPayLoad((Properties) context.getCommandPayload());

}

/\*\* The Constant LOGGER. \*/

**private** **static** **final** Logger *LOGGER* = Logger.*getLogger*(BaseResultSetExecution.**class**.getName());

/\*\* The Constant UNIT\_CONVERT. \*/

**private** **static** **final** String *UNIT\_CONVERT* = "unitConvert";

/\*\* The metadata. \*/

**private** RuntimeMetadata metadata;

/\*\* The command pay load. \*/

**private** Properties commandPayLoad;

/\*\* The result. \*/

List<?> result;

/\*\*

\* **TODO**: need to change this to Command class and check for particular class before typecasting.

\*/

**private** QueryExpression command;

/\*\*

\* Connection Object.

\*/

**private** Object connection;

/\*\* The actual result set execution. \*/

**private** ResultSetExecution actualResultSetExecution;

/\*\*

\* **@return** the actualResultSetExecution

\*/

**public** ResultSetExecution getResultSetExecution() {

**return** actualResultSetExecution;

}

/\*\*

\* Sets the command pay load.

\*

\* **@param** commandPayLoad

\* the new command pay load

\*/

**public** **void** setCommandPayLoad(Properties commandPayLoad) {

**this**.commandPayLoad = commandPayLoad;

}

@Override

**public** **void** close() {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** cancel() **throws** TranslatorException {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** execute() **throws** TranslatorException {

**this**.getResultSetExecution().execute();

}

@Override

**public** List<?> next() **throws** TranslatorException, DataNotAvailableException {

*LOGGER*.info("Inside next() method........Do Unit Conversion");

**final** List<TableReference> tableReferences = command.getProjectedQuery().getFrom();

**for** (TableReference tableReference : tableReferences) {

**final** Table table = ((NamedTable) tableReference).getMetadataObject();

**final** List<Column> columns = table.getColumns();

**for** (Column column : columns) {

**final** Map<String, String> properties = column.getProperties();

**if** (properties.containsKey(*UNIT\_CONVERT*) && properties.get(*UNIT\_CONVERT*).equals(Boolean.*TRUE*)) {

System.*out*.println("UnitConvert extension property found here");

}

}

}

// **TODO** change this to provide actual results post unit conversion.

**return** **this**.getResultSetExecution().next();

}

/\*\*

\* Sets the metadata.

\*

\* **@param** metadata

\* the new metadata

\*/

**public** **void** setMetadata(RuntimeMetadata metadata) {

**this**.metadata = metadata;

}

**public** **void** setCommand(QueryExpression command) {

**this**.command = command;

}

**public** **void** setConnection(Object connection) {

**this**.connection = connection;

}

**public** **void** setResultSetExecution(ResultSetExecution resultSetExecution) {

**this**.actualResultSetExecution = resultSetExecution;

}