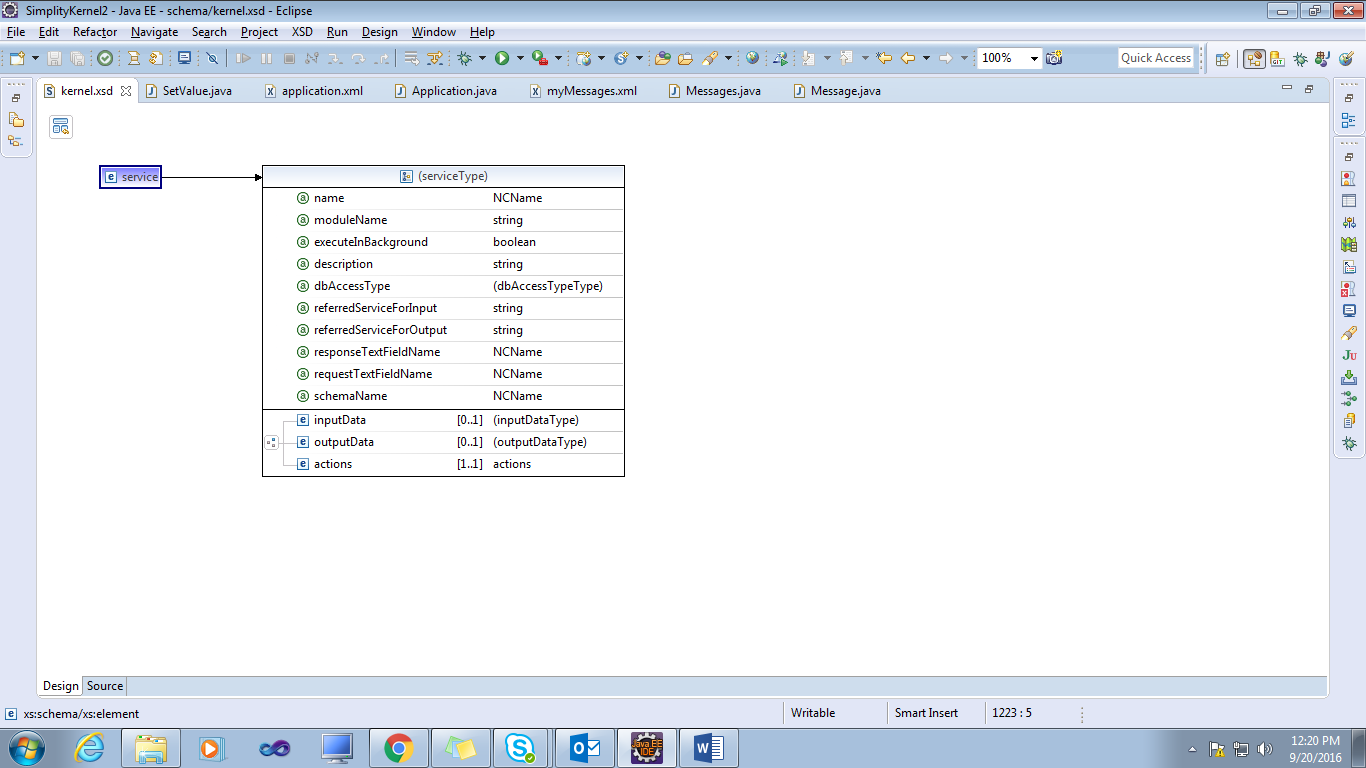
# **Simplity**

Simple set of APIs/framework for typical transaction processing systems

## **Service:**

Component which helps in Transaction Processing Service



|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <service dbAccessType=*"none"* description=*"testDescription"* executeInBackground=*"true"* moduleName=*"Testmodule"* name=*"NCName"* referredServiceForInput=*"testserviceinput"*referredServiceForOutput=*"testserviceoutput"* requestTextFieldName=*"NCName"* responseTextFieldName=*"NCName"*  schemaName=*"NCName"* xmlns=*"http://www.simplity.org/schema"*>  <inputData>  <inputFields />  <inputRecords>  <inputRecord description=*"inputrecord"* fieldNames=*"testfield"*  maxRows=*"0"* minRows=*"0"* purpose=*"read"* recordName=*"testrecordname"*  sheetName=*"NCName"* />  </inputRecords>  </inputData>  <outputData fieldNames=*"testooutputdata"*>  <outputRecords>  <outputRecord linkColumnInParentSheet=*"testouputsheet"*  linkColumnInThisSheet=*"testlink"* parentSheetName=*"testparentsheet"*  recordName=*"tesadf"* sheetName=*"NCName"* />  </outputRecords>  </outputData>  <actions>  <addColumn actionName=*"testaction"* columnName=*"NCName"*  columnValue=*"testcolumn"* columnValueExpression=*"testexpression"*  columnValueType=*"text"* description=*"fasdfa"* executeIfNoRowsInSheet=*"klfasj"*  executeIfRowsInSheet=*"ijskdjfa"* executeOnCondition=*"jhasd"* sheetName=*"sfas"* />  </actions>  </service> |

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| Name | Simple name | Required |
| moduleName | Module name.simpleName would be fully qualified name. | Optional |
| dbAccessType | Database access type | Required |
| **DbAccessType:** none, readOnly, readWrite, autoCommit, subService |
| requestTextFieldName | Do not parse the request text. Just set it to this field. Service will take care of that | Optional |
| inputData | Input fields/grids for this service. not valid if requestTextFieldName is specified | Optional |
| referredServiceForInput | copy input records from another service | Optional |
| referredServiceForOutput | copy output records from another service | Optional |
| responseTextFieldName | use this field as response | Optional |
| schemaName | schema name, different from the default schema, to be used specifically for this service | Optional |
| outputData | Output fields and grids for this service. Not valid if responseTextFieldName is specified | Optional |
| Actions | actions that make up this service | Required |
| executeInBackground | Should this be executed in the background ALWAYS?. | Optional |

Service child components.

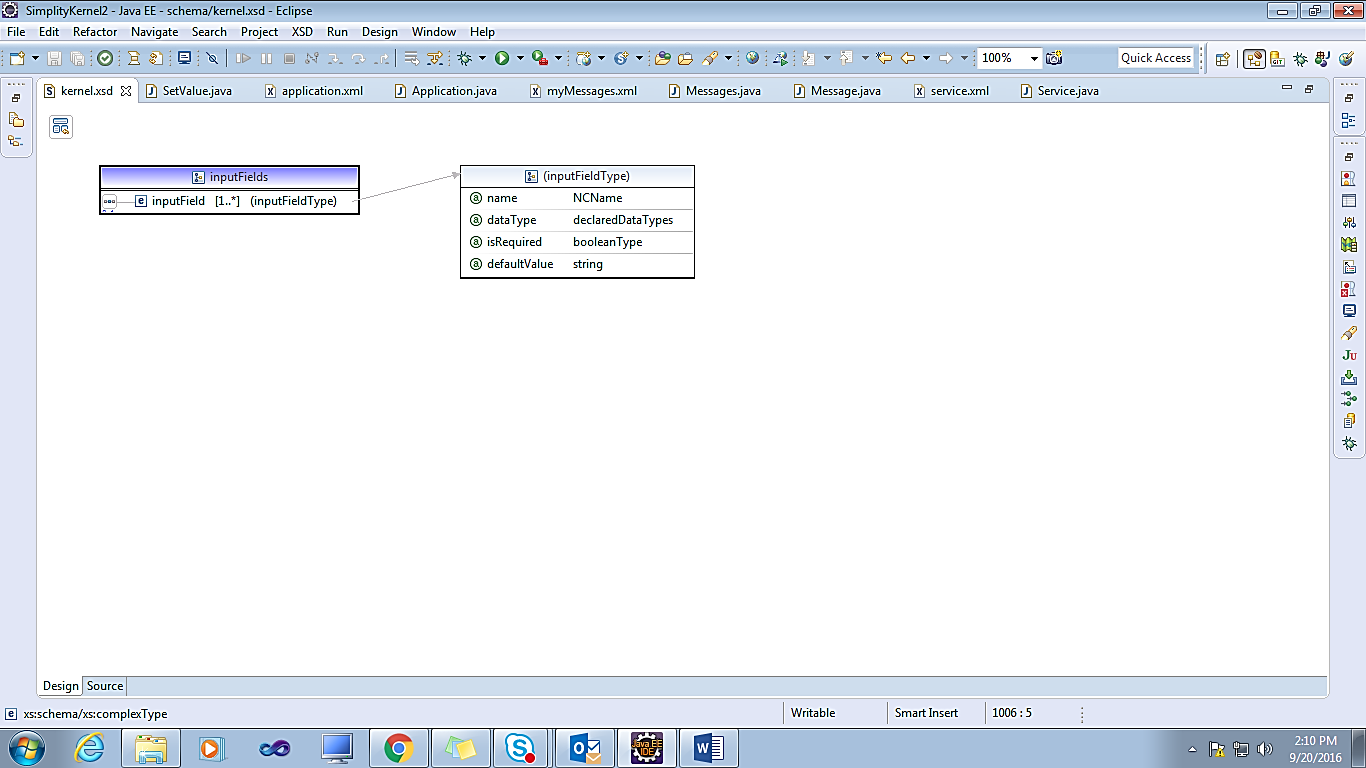
### **1. InputData**

Component that specifies what inputs are expected

InputFields data structure that holds name and data type of an input/output field.

#### 1.1InputField

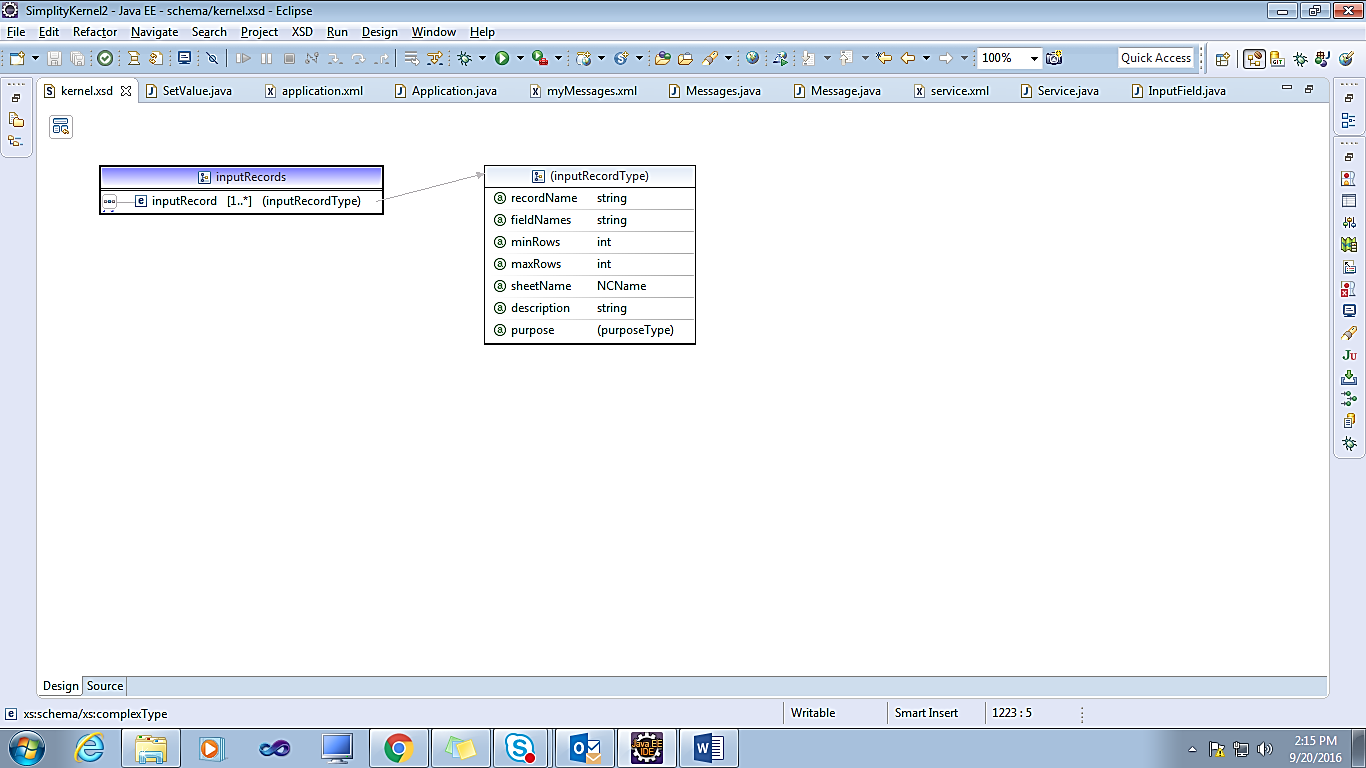
|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| Name | Name | Required |
| dataType | Data type. Used for validating in case this is used as input. Used for valueType otherwise | Required |
| isRequired | Used for validation during input | Optional |
| defaultValue | Used in case value is not available at run time, and isRequired is true | Optional |



InputRecords represents a record/row/table that is used as input for a service

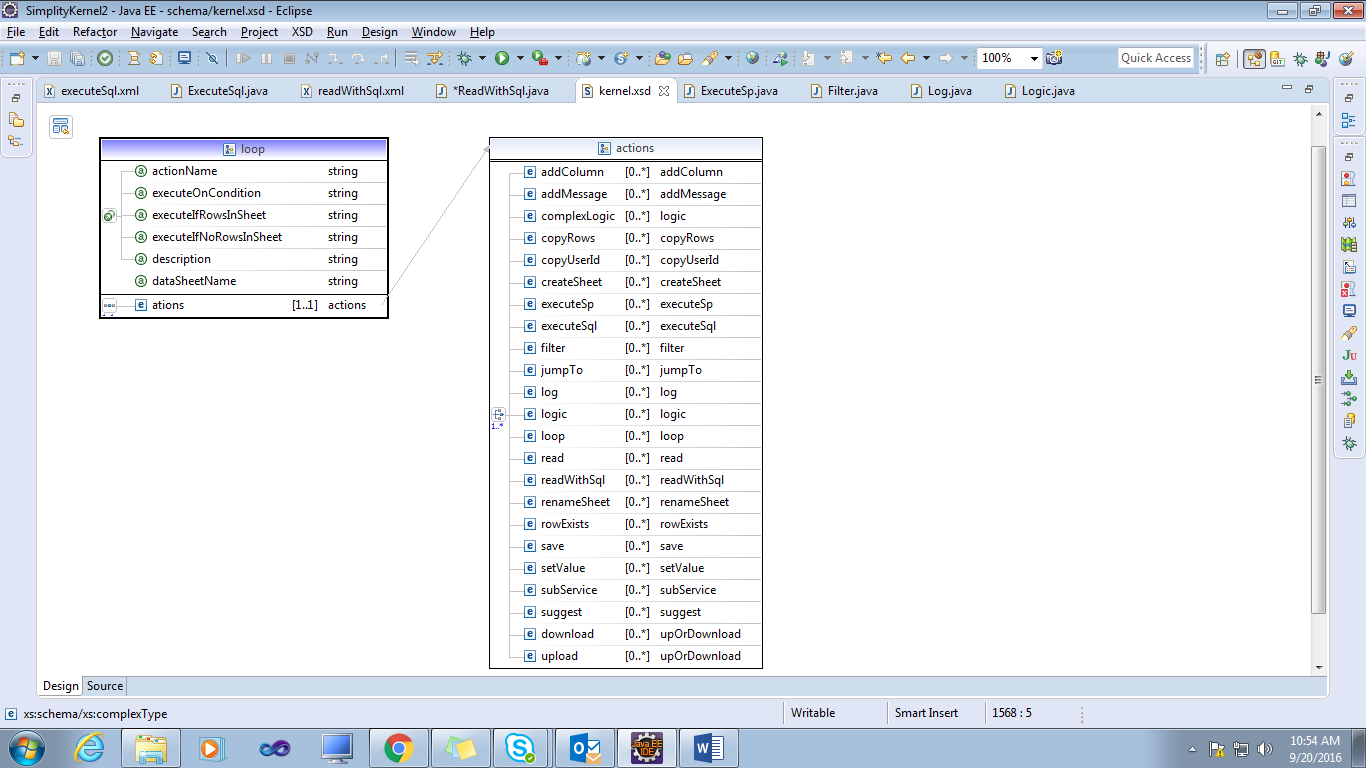
#### 1.2 InputRecord

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| recordName | Fully qualified name of the record that we are expecting as input. In very special case, like some utility service that is internally used, we may skip record. If record is skipped, input is accepted as it is with no validation. Must be used with care. | Required |
| fieldNames | For the purpose of input parsing and validation, assume this record has only these subset of fields. null if full row is in force. | Optional |
| minRows | Min rows expected. Used for validating input. | Optional |
| maxRows | Certainly a good idea to limit rows from a practical view. | Optional |
| Purpose | Why is this record being input? we extract and validate input based on this purpose | Optional |
| sheetName | Name of sheet in which we are expecting data. null if data is not  expected in a sheet | Optional |



### **2. Action**

An action inside a service



#### **2.1 AddColumn:**

This Component helps in adding column to a data sheet.

E.g.:

<actions>

<logic actionName="1" className="org.simplity.test.CustomAddColumn" />

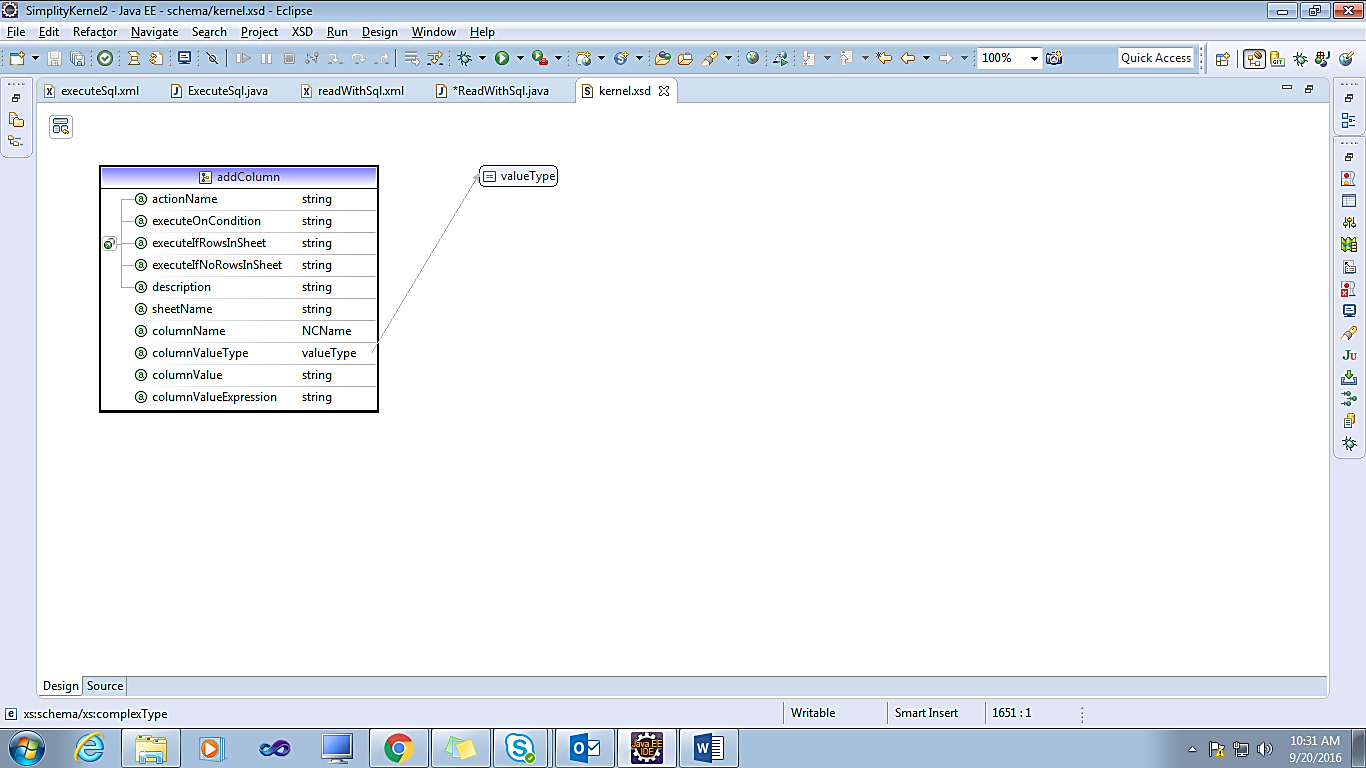
<addColumn sheetName="weekendBoxOffice" columnName="testcolumn" columnValueType="text"

columnValue="testValue" actionName="addColumnsAction"/>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| SheetName | Sheet to which we want to add a column | Required |
| ColumnName | Name of column to be added | Required |
| ColumnValueType | value type of the column | Required |
| **ValueType:** text, boolean, integer, decimal, date |
| ColumnValue | If the value of the column is known at design time, provide the value. In \* case it is the value of a field, use $fieldname as the value | Optional |
| ColumnValueExpression | If the column value is to be calculated as an expression that involves \* other columns, then provide the expression | Optional |



#### **2.2 AddMessage:**

Add a message to the context

E.g.:

<actions>

<setValue actionName=*"1"* fieldName=*"TestUser"* fieldValue=*"NeverLand"* description=*"Where did they live?"*></setValue>

<addMessage messageName=*"myMessages.a"* actionName=*"AddMessageAction"* parameters=*"$TestUser"* description=*"Testing Add message action"* />

</actions>

Sample message Component

<?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"no"*?>

<messages name=*"myMessages"* xmlns=*"http://www.simplity.org/schema"*>

<components>

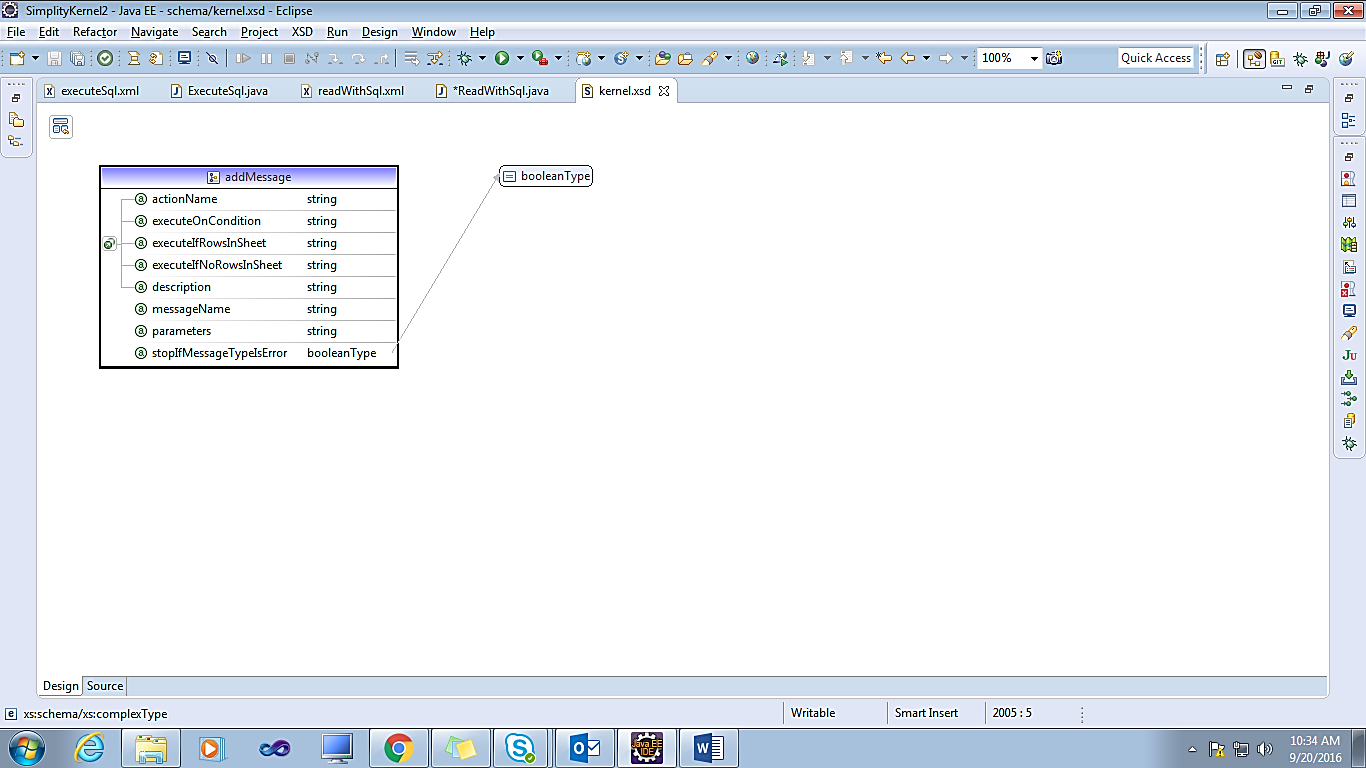
<message messageType=*"error"* name=*"myMessages.a"* text=*"$1 Custom Message From My Messages XML File"*/>

</components>

</messages>

**Attributes of AddMessage**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| messageName | Mandatory name of a predefined message | Required |
| Parameters | List of parameters | Optional |
| stopIfMessageTypeIsError | Should we stop this service in case the message added is of type error | Optional |



#### **2.3 Complexlogic**

Complex logic that is implemented in a java code. Complexity in this is the driver. We believe that a logic that requires db operations in between is leads to maintenance issues, and hence must be developed and reviewed by experienced folks. For enabling such a process, we have separated simple and complex logic interfaces.

#### **2.4 CopyRows**

Copy rows from a compatible sheet.

E.g.:

<actions>

<logic actionName=*"1"* className=*"org.simplity.test.CustomCopyRowsAction"* />

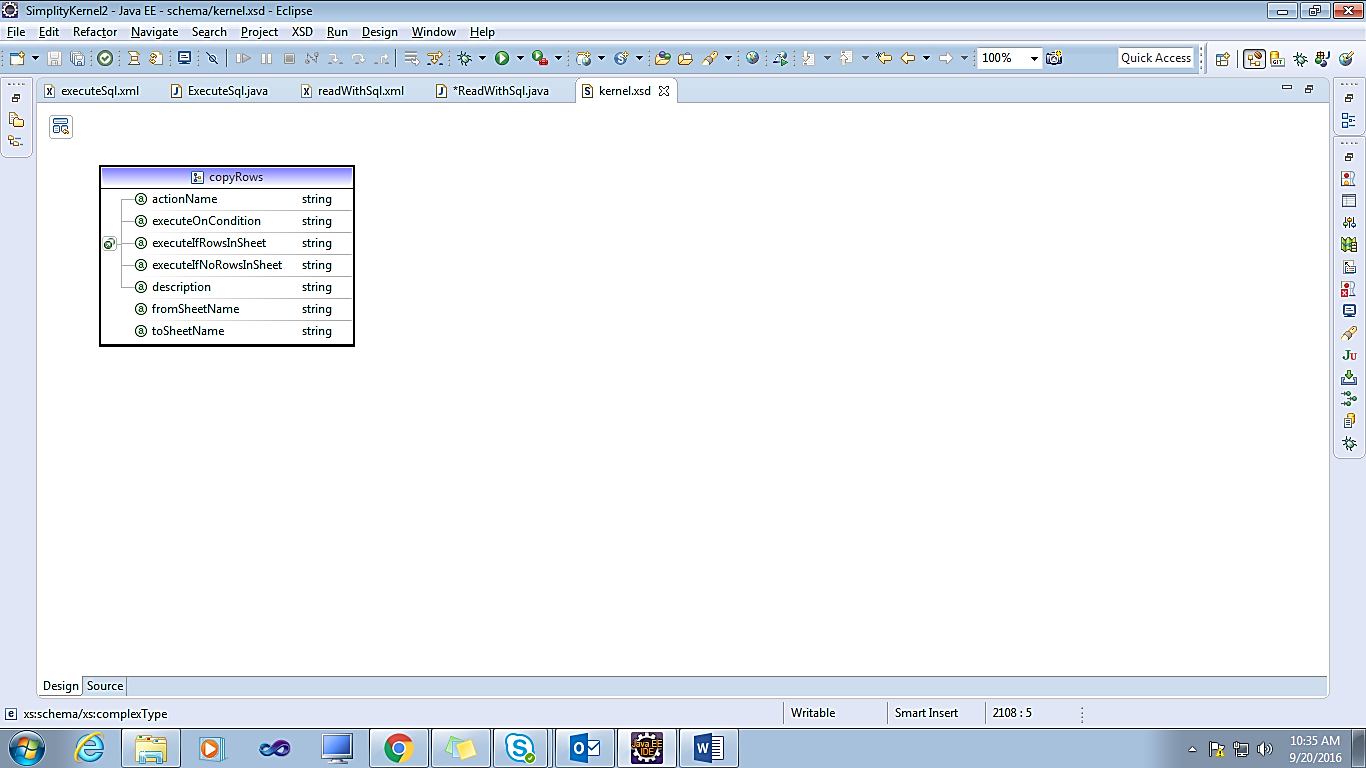
<copyRows fromSheetName=*"weekendBoxOffice"* toSheetName=*"copiedweekendBoxOffice"*

actionName=*"copyRowsAction"* />

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| toSheetName | Sheet to which we want to add rows | Required |
| fromSheetName | Sheet from which to copy rows | Required |



#### **2.5 CopyUserId**

Set the special value userId that has the logged-in user id.

<actions>

<copyUserId fieldName=*"UserIDCopy"* actionName=*"copyUserIdAction"* description=*"Copy user id from context"*/>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| fieldName | field name to which user id is to be set to | Required |



#### **2.6 CreateSheet**

Set rows and columns to a table

<actions>

<setValue fieldName=*"dateField"* fieldValue=*"/2015-01-31/"* />

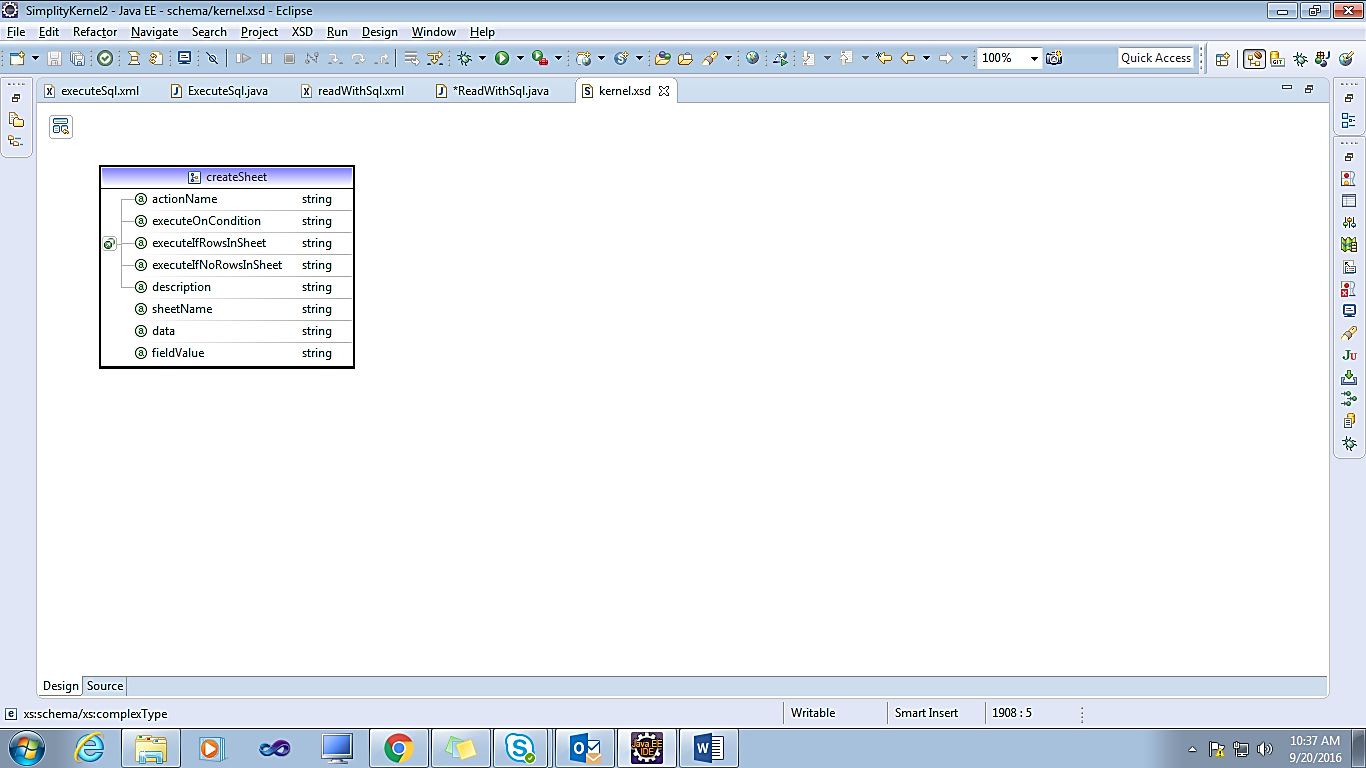
<createSheet sheetName=*"newsheet"* data=*"text, int, decimal, date, bool; first row first column text, 1, 1.1, /2016-12-31/,true;secondrow,2,2.2, $dateField, true"* />

<log names=*"newsheet"*/>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| sheetName | Name of sheet to be created | Required |
| Data | This is a semi-colon (;) separated set of rows. Each row is a comma(,)  separated set of columns; First row is header. e.g.name, value; red,1;green,2;blue,3. Note that the syntax does not allow text that may contain comma or semi-colon | Required |



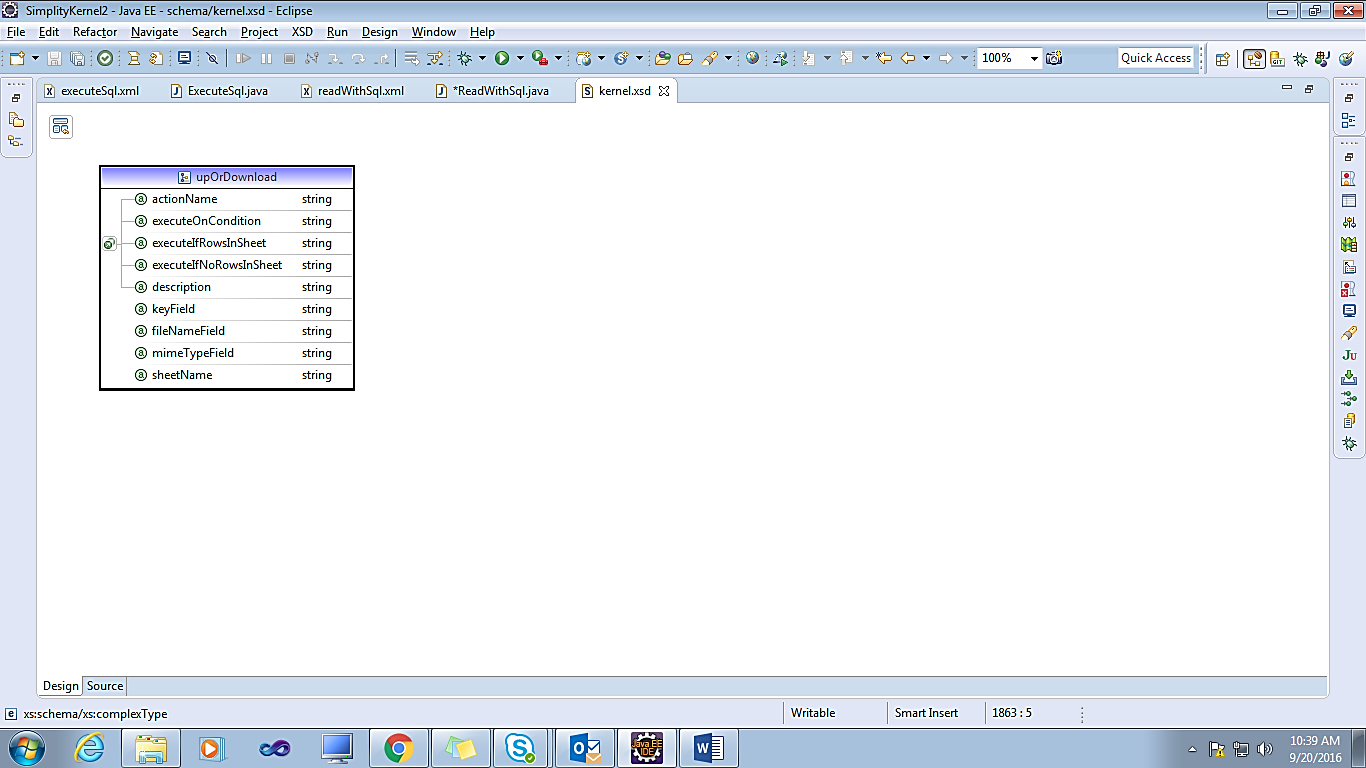
#### **2.7 Download**

Download a file from permanent storage to temp storage

|  |
| --- |
| <actions>  <logic className=*"org.simplity.test.GetStoredFiles"* description=*"Creates sheet named files that has the keys for files to be downaded"* />  <download sheetName=*"files"* keyField=*"key"* fileNameField=*"name"* mimeTypeField=*"mime"* description=*"Downloads a file for each row in sheet files.*  *Uses column named key as key to the file to be downloaded. copies file name and mime-type of teh downloaded file to the columns file and mime respectively."* />  </actions> |

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| keyField | Field/column name that has the key for the file | Required |
| fileNameField | Field to which file name is to be copied to. This is optional | Optional |
| mimeTypeField | Field to which mime-type of this file is to be copied to. This is optional | Optional |
| sheetName | Sheet name in case this action is for all rows of a sheet | Optional |

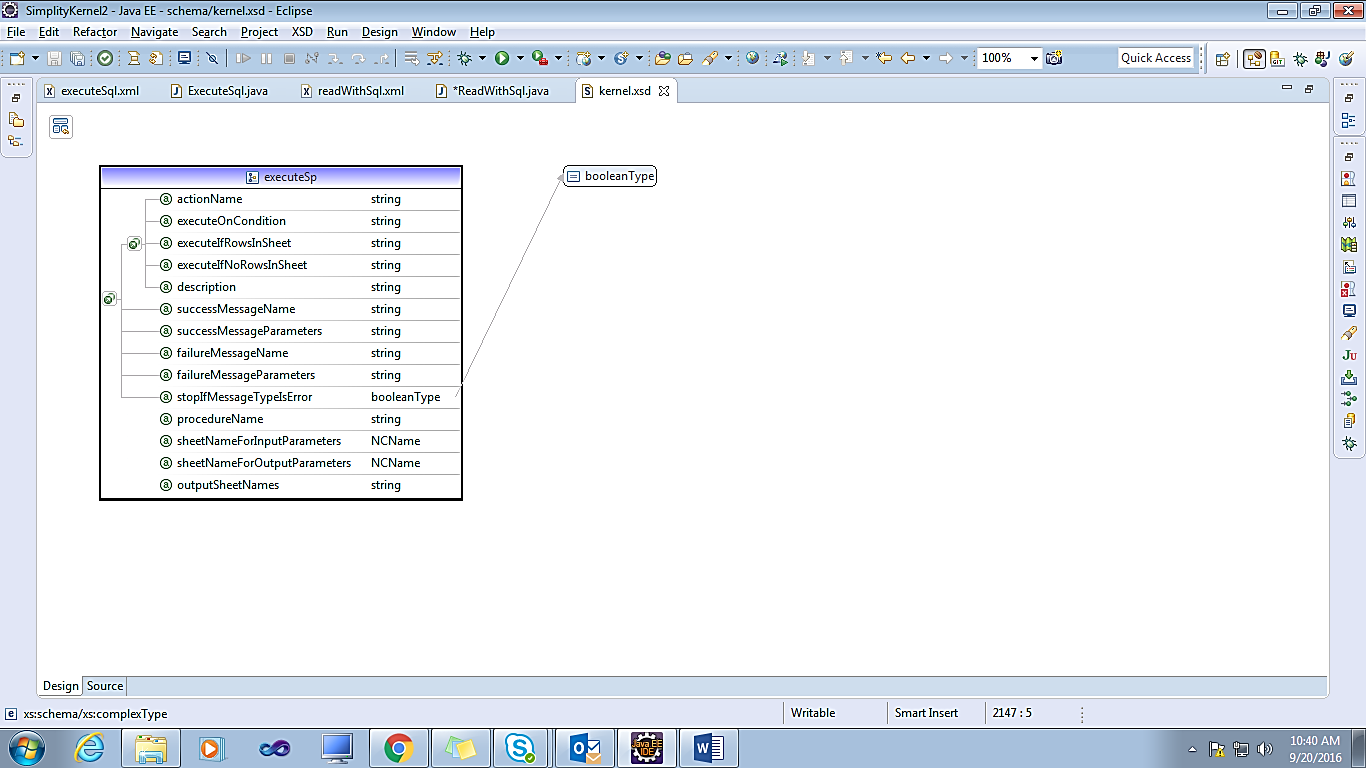


#### **2.8 ExecuteSP**

Execute a stored procedure

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| procedureName | Qualified name | Required |
| sheetNameForInputParameters | Sheet name for input data. Null implies that input, if any, would be from fields collection of ctx | Optional |
| sheetNameForOutputParameters | Output parameters from this SP may be extracted out to a sheet. Else, if present, they will be extracted to fields collection of ctx | Optional |
| outputSheetNames | If this procedure has defined outputRecordNames, then you may specify the sheet names. Default is to use definition from record. | Optional |



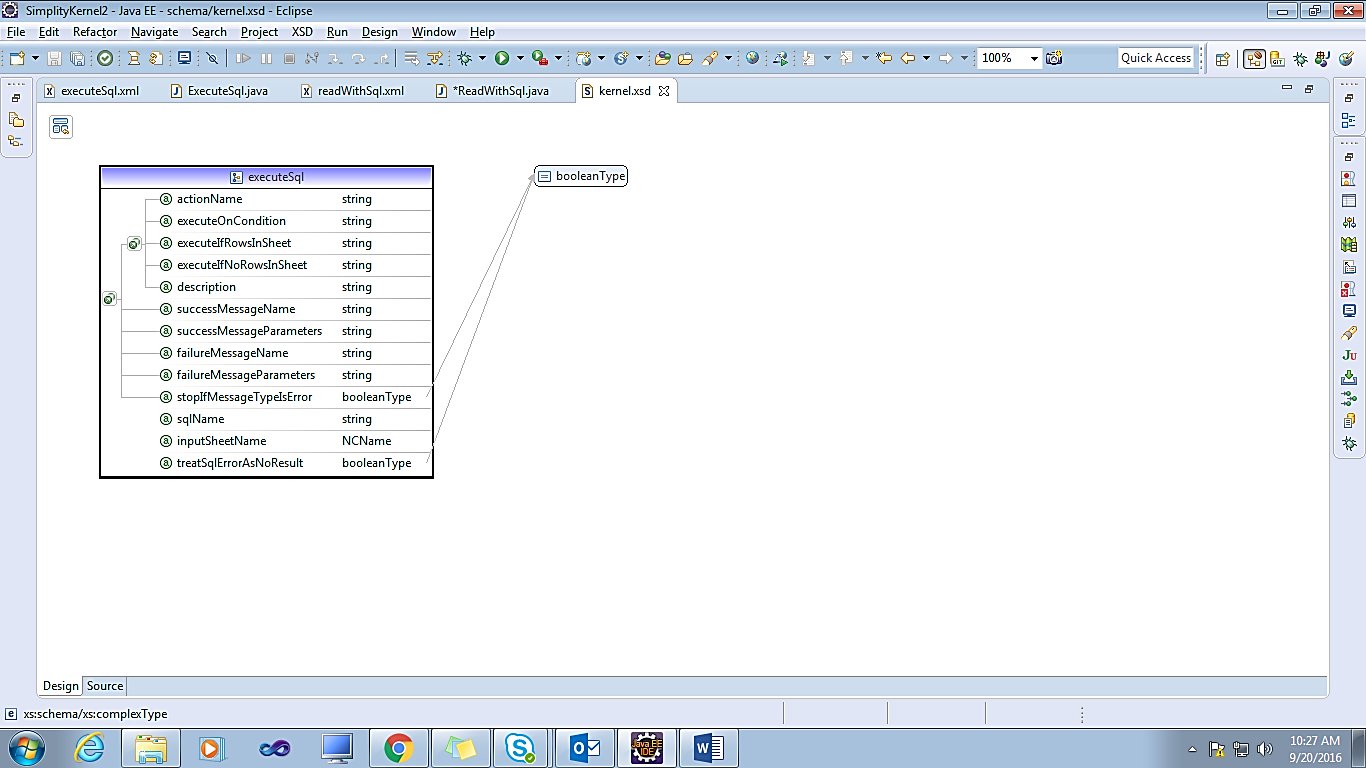
#### **2.9 ExecuteSql**

Execute a sql

|  |
| --- |
| <actions>  <executeSql sqlName=*"example.Offices"* actionName=*"ExecuteSqlQuery"* />  <setValue fieldName=*"updateSql"* fieldValue=*"$ExecuteSqlQueryResult"* ></setValue>  </actions> |

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| sqlName | Qualified sql name | Required |
| inputSheetName | Sheet name for input data | Optional |
| treatSqlErrorAsNoResult | Many a times, we put constraints in db, and it may be convenient to use that to do the validation. For example we insert a row, and db may raise an error because of a duplicate columns. In such a case, we treat this error as "failure", rather than an exception | Optional |

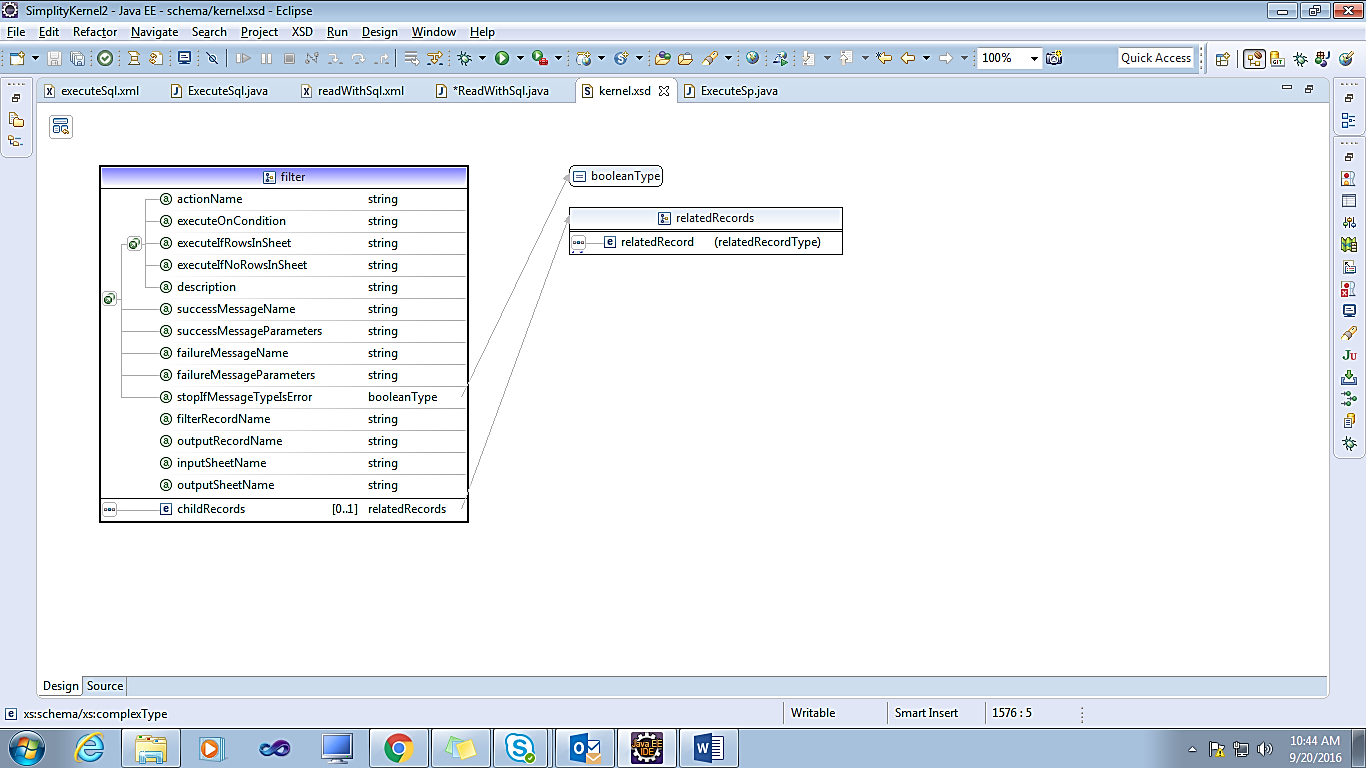


#### **2.10 Filter**

Read a row from a record, and possibly read relevant rows from related records

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| filterRecordName | Record that is used for inputting and creating filter criteria | Required |
| outputRecordName | Optional. defaults to filterRecordName | Optional |
| inputSheetName | Name of the sheet in which data is received. If null, we take data from fields. Sheet cannot contain more than one rows | Optional |
| outputSheetName | Name of the sheet in which output is sent. Defaults to simple name of outputRecordName | Optional |
| childRecords | child records from which to read rows, for the row read in this record | Optional |



#### **2.11 JumpTo**

Service has actions that are executed in a sequence. JumpTo allows you to change this sequence.

<actions>

<setValue actionName=*"one"* fieldName=*"island"* fieldValue=*"NeverLand"* description=*"Where did they live?"*></setValue>

<setValue actionName=*"two"* fieldName=*"leader"* fieldValue=*"Peter Pan"*></setValue>

<setValue actionName=*"three"* fieldName=*"gang"* fieldValue=*"Lost Boys"*></setValue>

<jumpTo toAction=*"six"* executeOnCondition=*"gang='Lost Boys'"*/>

<setValue actionName=*"four"* fieldName=*"adversary1"* fieldValue=*"Captain Hook"*></setValue>

<setValue actionName=*"five"* fieldName=*"adversary2"* fieldValue=*"Mr.Smee"*></setValue>

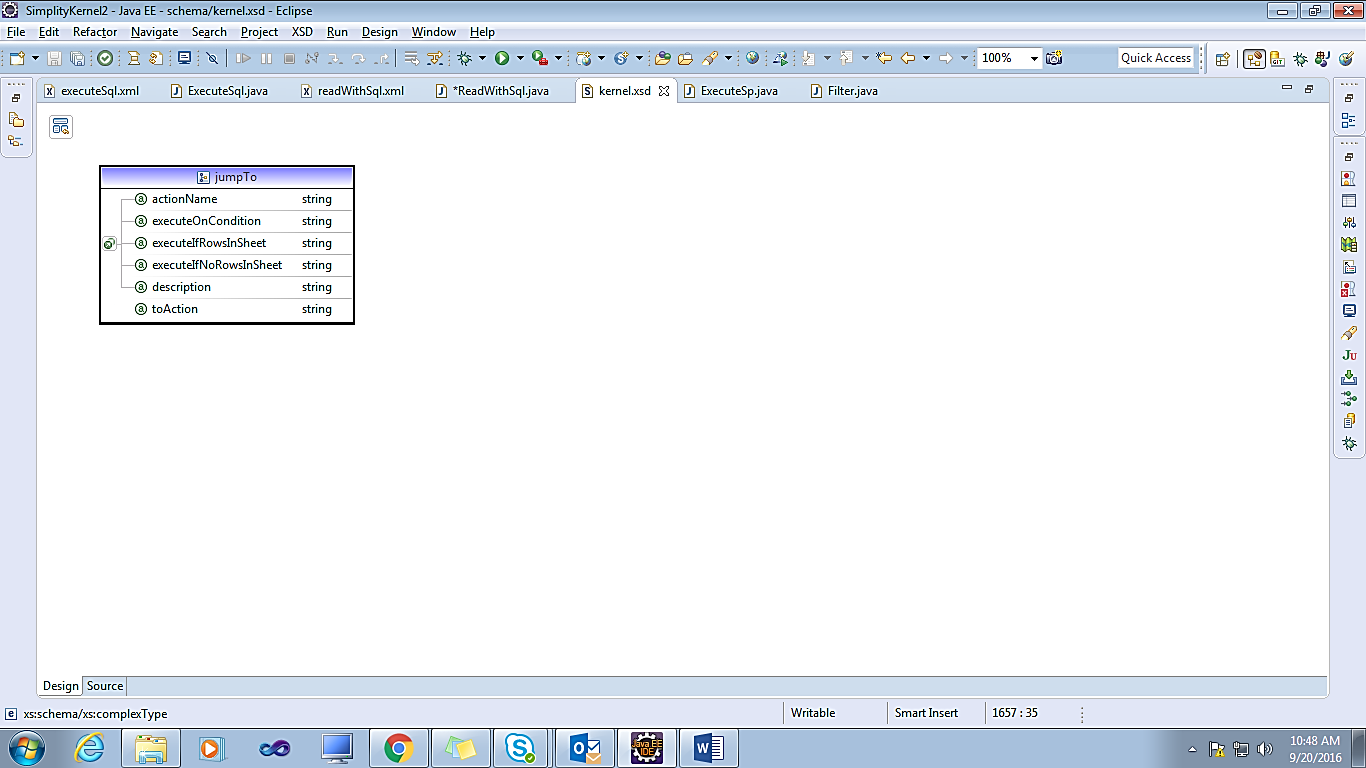
<setValue actionName=*"six"* fieldName=*"comma"* fieldValue=*"James Patrick"*></setValue>

<setValue actionName=*"seven"* fieldName=*"novel"* fieldValue=*"The Little White dfssdf Bird"* ></setValue>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| toAction | returns either a name of action to go to, or "\_stop", "\_error”, continue", "\_break" | Required |

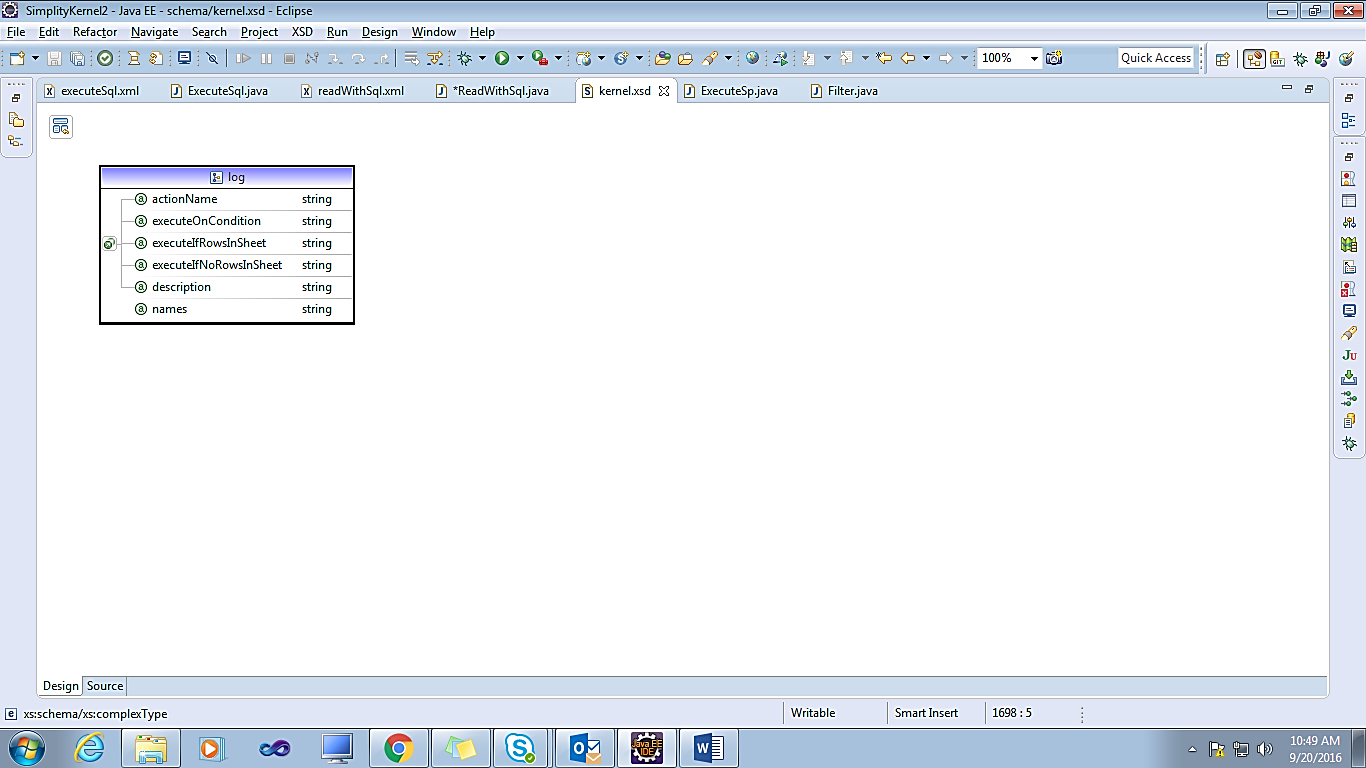


#### **2.12 Log**

Show values of fields/table: meant for debugging

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| Names | Field/table names to be logged | Required |



#### **2.13 Logic**

Logic that is implemented in a java code

<actions>

<logic actionName=*"1"* className=*"org.simplity.test.CustomLogicAction"* />

<setValue actionName=*"2"* fieldName=*"novel"* fieldValue=*"The Little White Bird"* executeOnCondition=*"?leader"*></setValue>

<setValue actionName=*"3"* fieldName=*"emptyDS"* fieldValue=*"Sheet is empty"* executeIfNoRowsInSheet=*"emptySheet"*></setValue>

<setValue actionName=*"4"* fieldName=*"NotEmptyDS"* fieldValue=*"Sheet is not empty"* executeIfNoRowsInSheet=*"weekendBoxOffice"*></setValue>

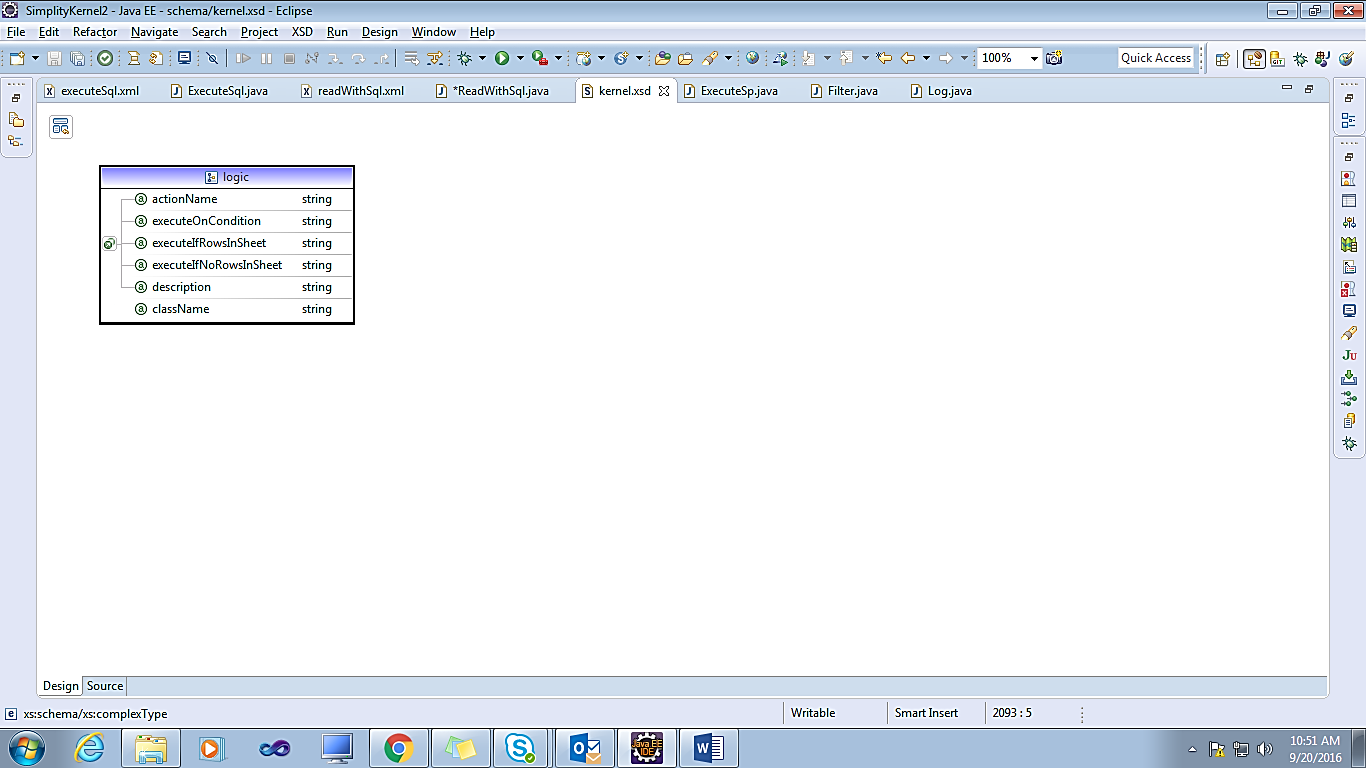
<setValue actionName=*"5"* fieldName=*"checkExpression"* expression=*"add(1,2)"*></setValue>

<log names=*"switch,island,leader,gang,adversaries,novel,lostBoys,emptyDS,NotEmptyDS,checkExpression"*/>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| className |  | Required |

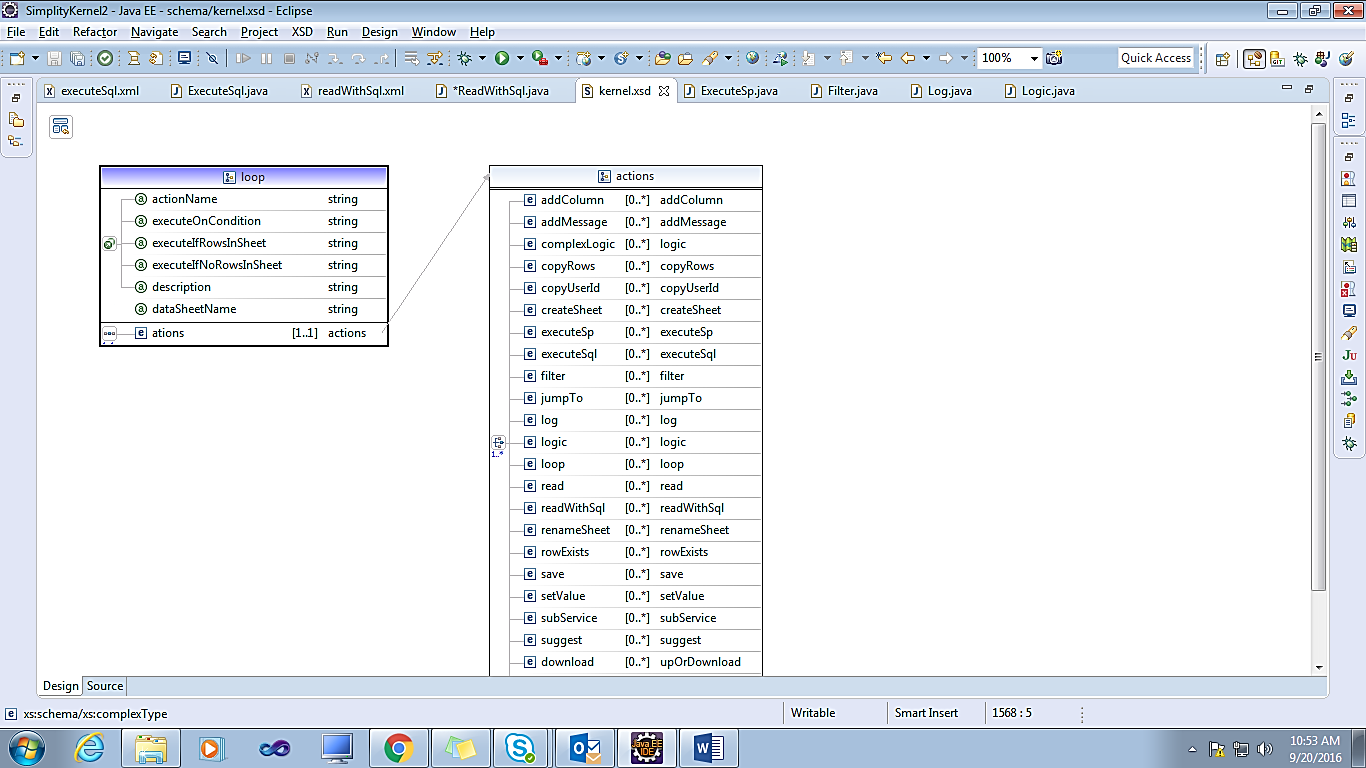


#### **2.14 Loop**

Loop through a set of actions for each row in a data sheet

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| dataSheetName | Returns either a name of action to go to, or "\_stop", "\_error" | Required |
| Actions | Actions that are to be performed for each row of the data sheet | Required |



#### **2.15 Read**

Read a row from a record, and possibly read relevant rows from related records

<actions>

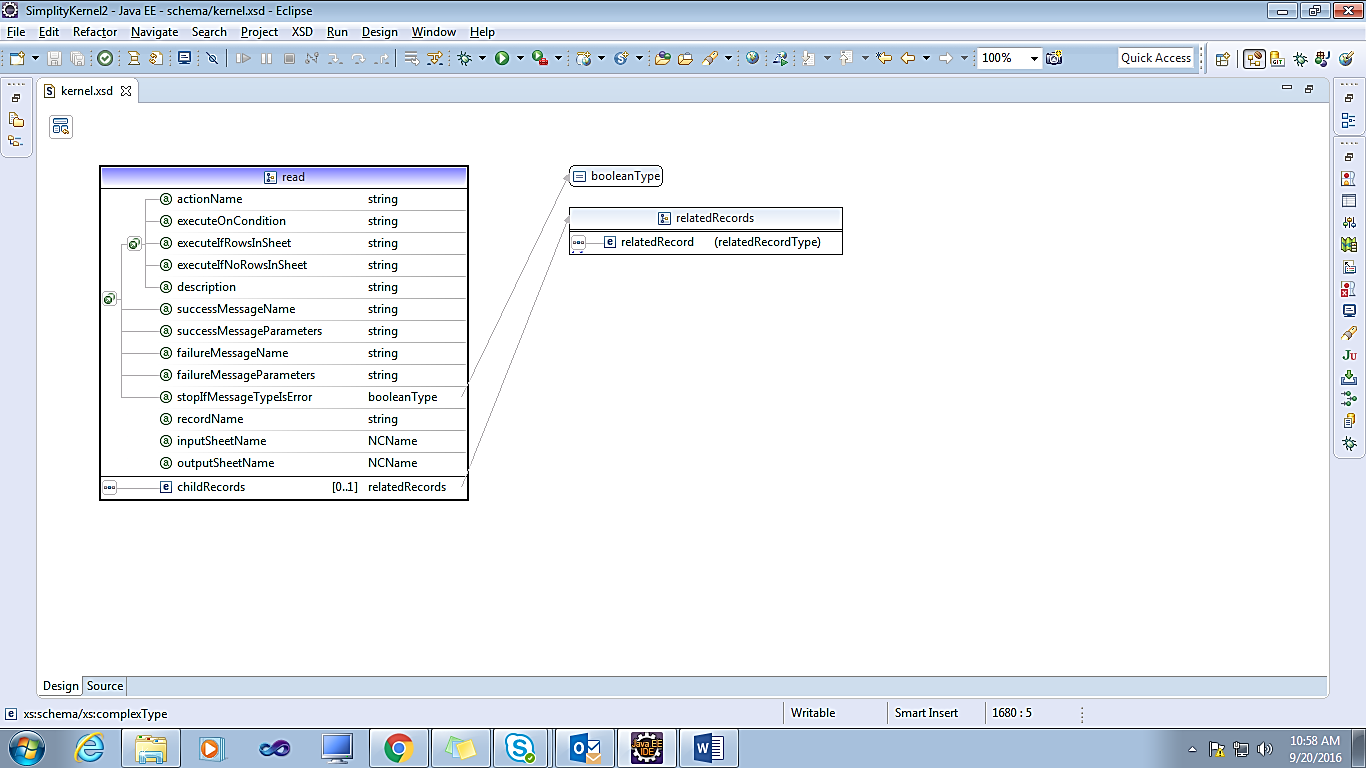
<setValue fieldName=*"customerNumber"* fieldValue=*"112"*></setValue>

<read recordName=*"birt.customers"* actionName=*"readCustomerData"*></read>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| recordName | Qualified record name | Required |
| inputSheetName | Sheet in which input is expected. defaults to simple name of record | Optional |
| outputSheetName | Sheet name in which output is read into. defaults to simple name of record | Optional |
| childRecords | Child records from which to read rows, for the row read in this record | Optional |



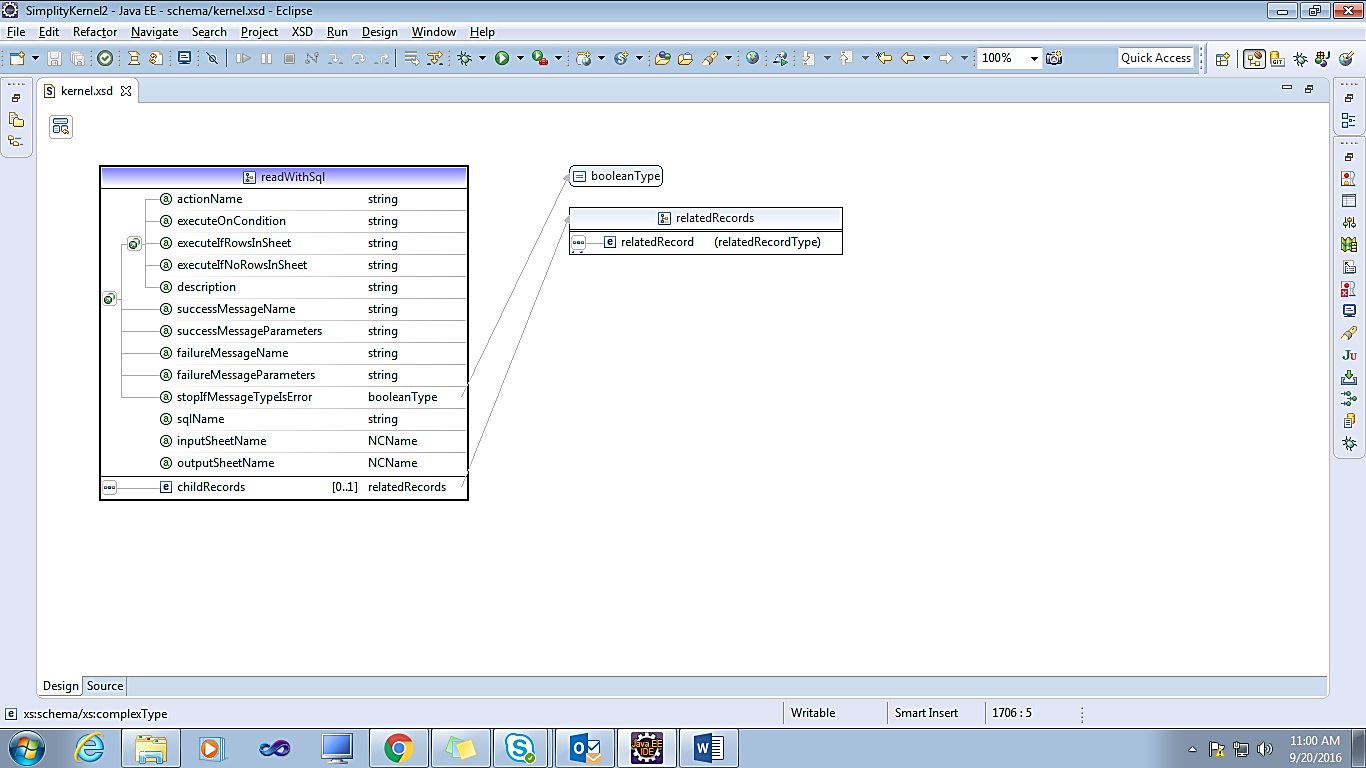
#### **2.16 ReadWithSql**

Read a row/s from as output of a prepared statement/sql

|  |
| --- |
| <actions>  <readWithSql sqlName=*"example.Employees"* outputSheetName=*"Employees"*></readWithSql>  </actions> |

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| sqlName | Fully qualified sql name | Required |
| inputSheetName | Input sheet name | Optional |
| outputSheetName | Output sheet name | Required |
| childRecords | Any other child records to be read for this record? | Optional |



#### **2.17 RenameSheet**

Rename a data sheet. Returns true if renaming is successful, false otherwise

<actions>

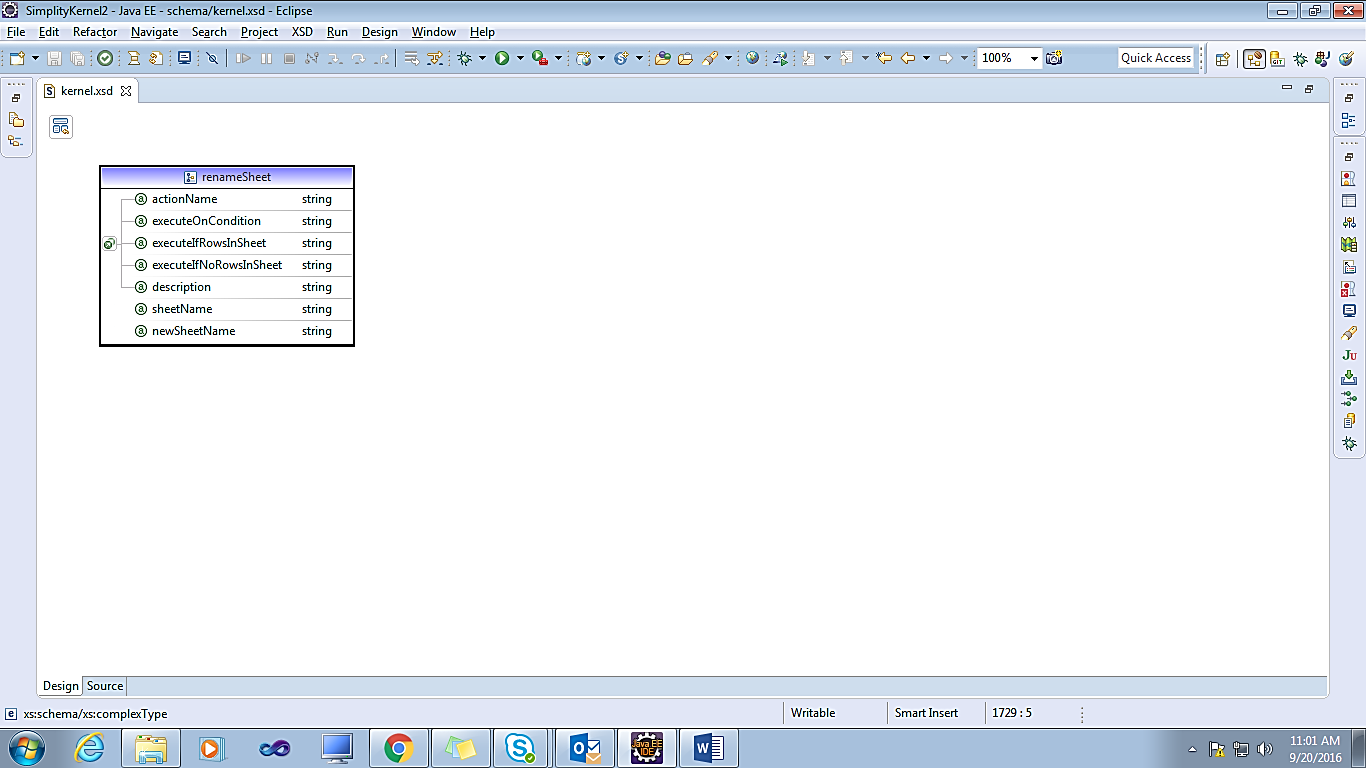
<logic actionName=*"1"* className=*"org.simplity.test.CustomLogicAction"* />

<renameSheet newSheetName=*"newweekendBoxOffice"* sheetName=*"weekendBoxOffice"* actionName=*"renameSheetAction"*/>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| sheetName | Current name | Required |
| newSheetName | New name | Required |



#### **2.18 RowExists**

Check if a row exists in this record for the primary key

<actions>

<setValue fieldName=*"employeeNumber"* fieldValue=*"1056"*></setValue>

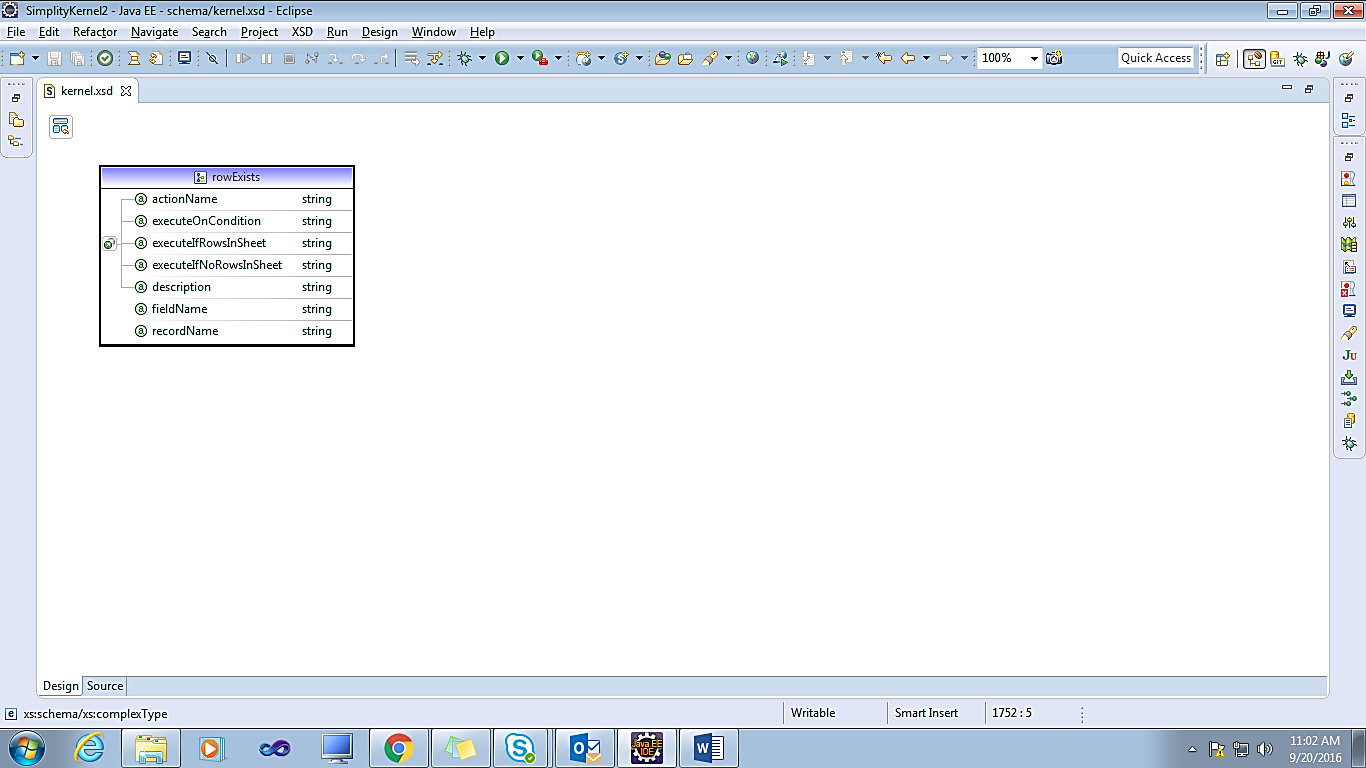
<rowExists recordName=*"birt.Employees"* actionName=*"rowExistsAction"* description=*"Check if row exists"* fieldName=*"employeeNumber"* />

<setValue fieldName=*"testValue"* fieldValue=*"1234"* executeOnCondition=*"rowExistsActionResult"*></setValue>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| recordName | Qualified record name | Required |
| fieldName | Specify the field that has the value for the key. Defaults to key specified in record | Optional |



#### **2.19 Save**

Save (add/modify/delete) a row from a record, and possibly save relevant rows from related records

**Save service with action type ADD**

|  |
| --- |
| <save saveAction="add" recordName="birt.Employees" actionName="saveAction"></save> |

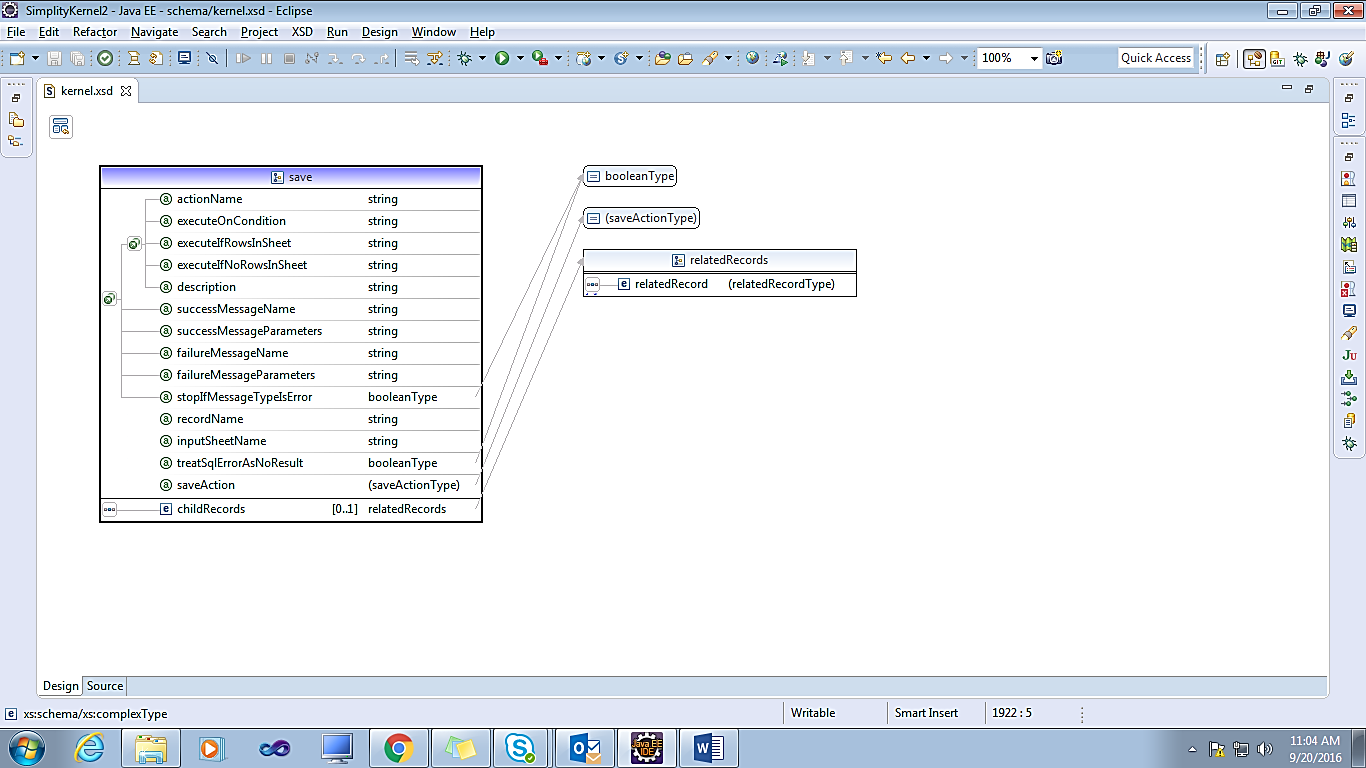
**Save service with action type Delete**

|  |
| --- |
| <save saveAction="delete" recordName="birt.Employees" actionName="deleteAction"></save> |

**Save service with action type Modify**

|  |
| --- |
| <save saveAction="modify" recordName="birt.Employees" actionName="modifyAction"></save> |

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| recordName | Qualified record name | Required |
| inputSheetName | If this is for more than one rows, and the data is to be received in a sheet | Optional |
| saveAction | Add/update/delete/auto/replace. Auto means update if primary key is specified, else add | Required |
| **SaveAction:** save, add, modify, delete |
| childRecords | Do we save child records | Optional |
| treatSqlErrorAsNoResult | At times, you may design an insert operation that will try to insert, failing which you may want to update. In such cases, you may get a sql error on key-violation. By default we would raise an exception. You may alter this behavior with this keyword. | Optional |



#### **2.20 SetValue**

Simply set value for a field

<actions>

<setValue actionName=*"1"* fieldName=*"island"* fieldValue=*"NeverLand"* description=*"Where did they live?"*></setValue>

<setValue actionName=*"2"* fieldName=*"leader"* fieldValue=*"Peter Pan"*></setValue>

<setValue actionName=*"3"* fieldName=*"gang"* fieldValue=*"Lost Boys"*></setValue>

<setValue actionName=*"4"* fieldName=*"adversary1"* fieldValue=*"Captain Hook"*></setValue>

<setValue actionName=*"5"* fieldName=*"adversary2"* fieldValue=*"Mr.Smee"*></setValue>

<setValue actionName=*"6"* fieldName=*"comma"* fieldValue=*","*></setValue>

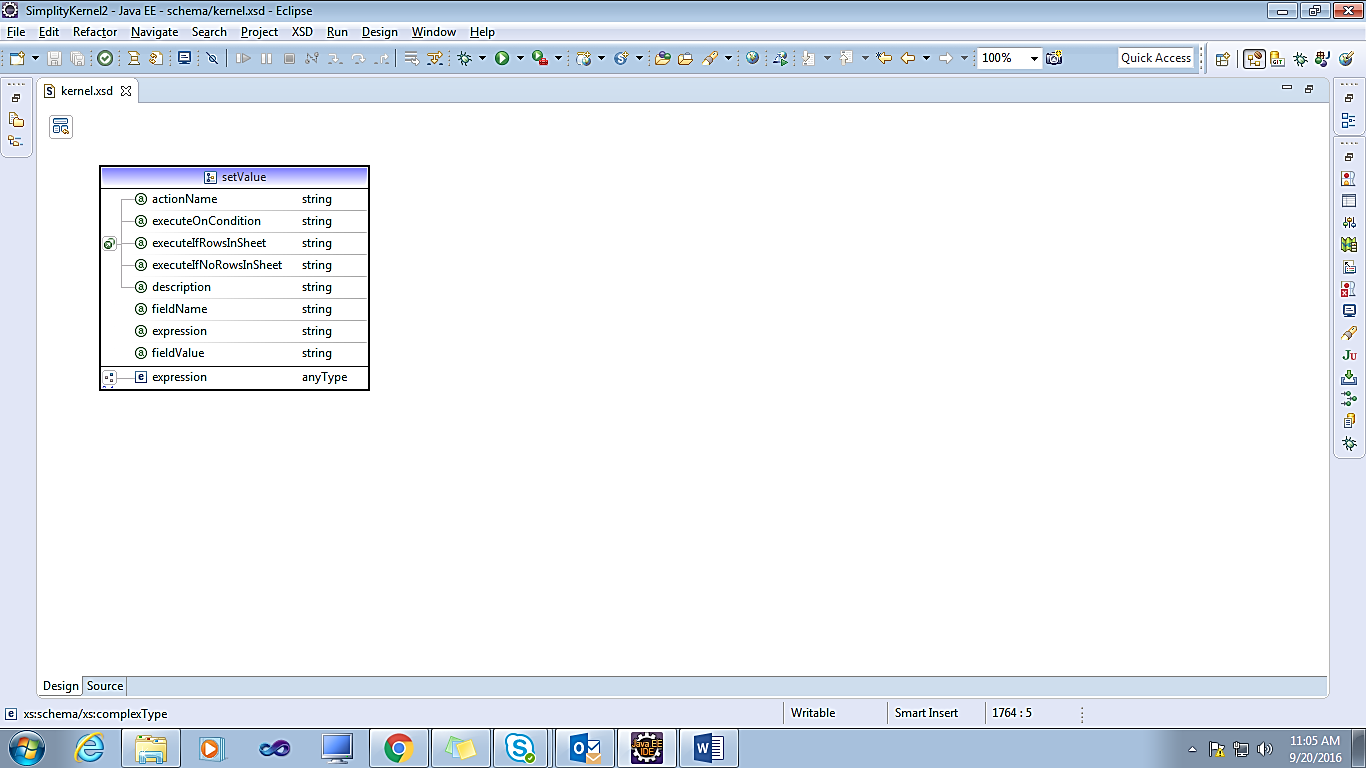
<setValue actionName=*"7"* fieldName=*"adversaries"* expression=*"adversary1 + comma + adversary2"*></setValue>

<setValue actionName=*"8"* fieldName=*"novel"* fieldValue=*"The Little White Bird"* executeOnCondition=*"?leader"*></setValue>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| fieldName | Field name | Required |
| fieldValue | value, if it is known at design time, or a single field, like 23 or $customerName | Optional |
| Expression | If value is more complex | Optional |



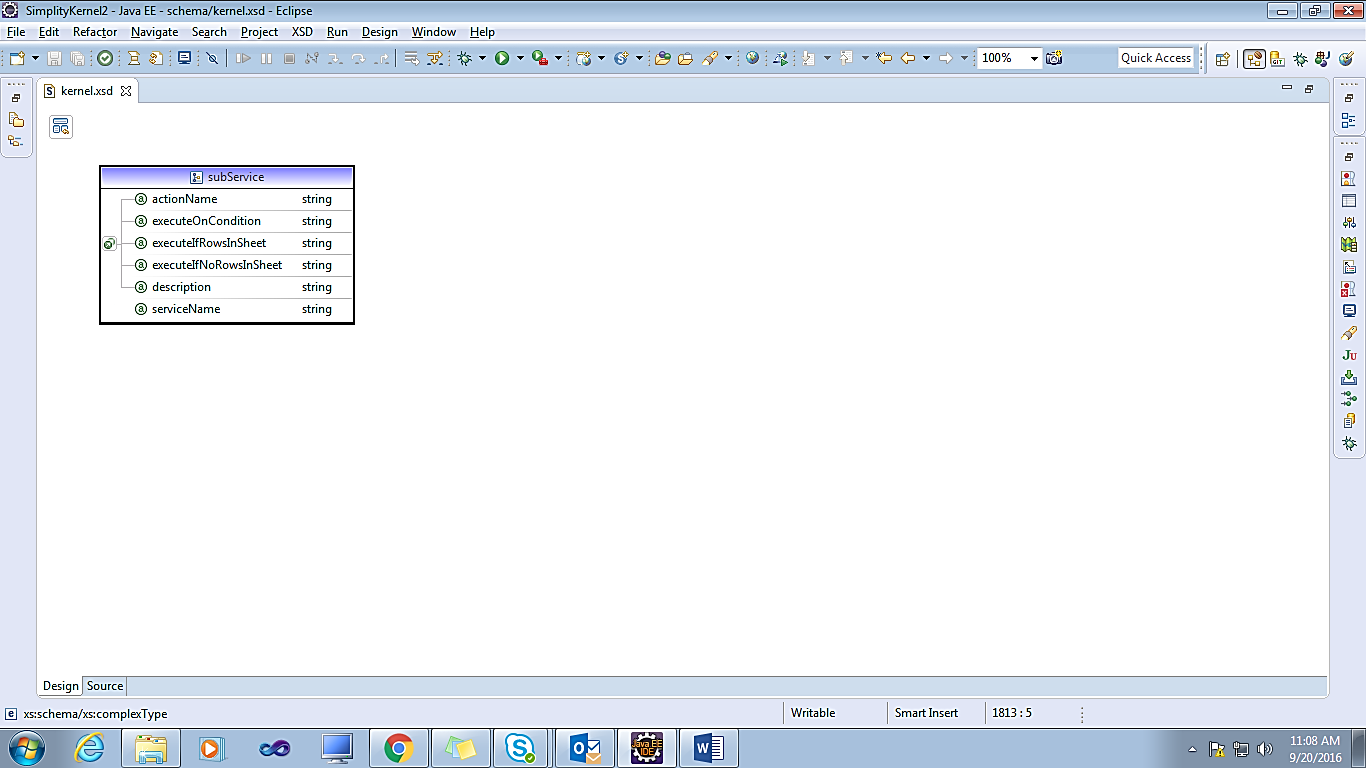
#### **2.21 SubService**

Service that is to be executed as a step/action in side another service.

<subService serviceName=*"tutorial.getAttachments"* description=*"Refresh attachment list"* />

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| serviceName | Name of another service provider | Required |



#### **2.22 Suggest**

Get suggestions from the table associated with a record based on the inputs. This action is "auto-generated" for on-the-fly service, but action is useful for a service that has to do other things as well. This is a thin wrapper on record.suggest ()

<actions>

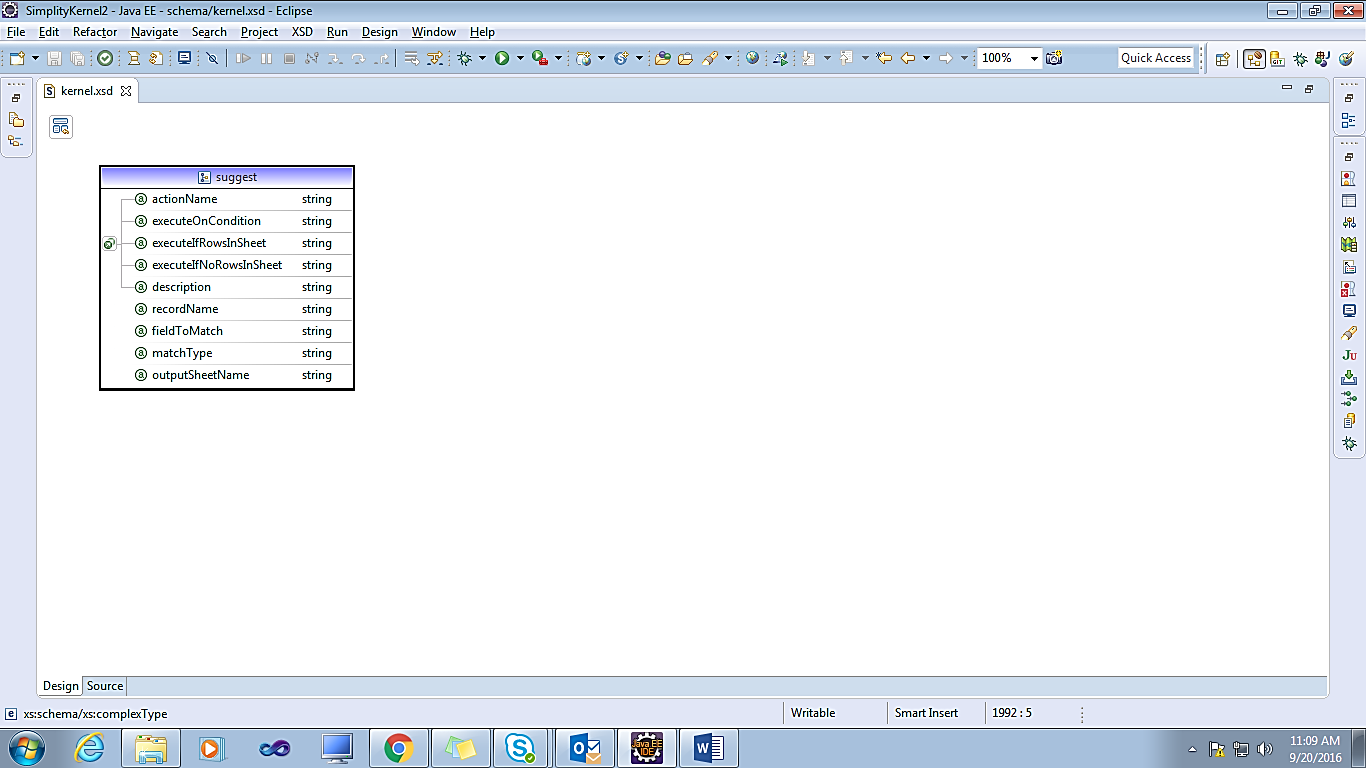
<setValue fieldName=*"firstName"* fieldValue=*"Mar"*></setValue>

<suggest recordName=*"birt.Employees"* actionName=*"suggestAction"* fieldToMatch=*"firstName"* outputSheetName=*"Employees"*></suggest>

</actions>

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| recordName | Record that is to be used | Required |
| fieldToMatch | Field to filter on. Defaults to specification in record | Optional |
| matchType | Should the matching criterion be fixed at design time or do we allow request-time input? | Optional |
| outputSheetName | Name of the output data sheet | Optional |



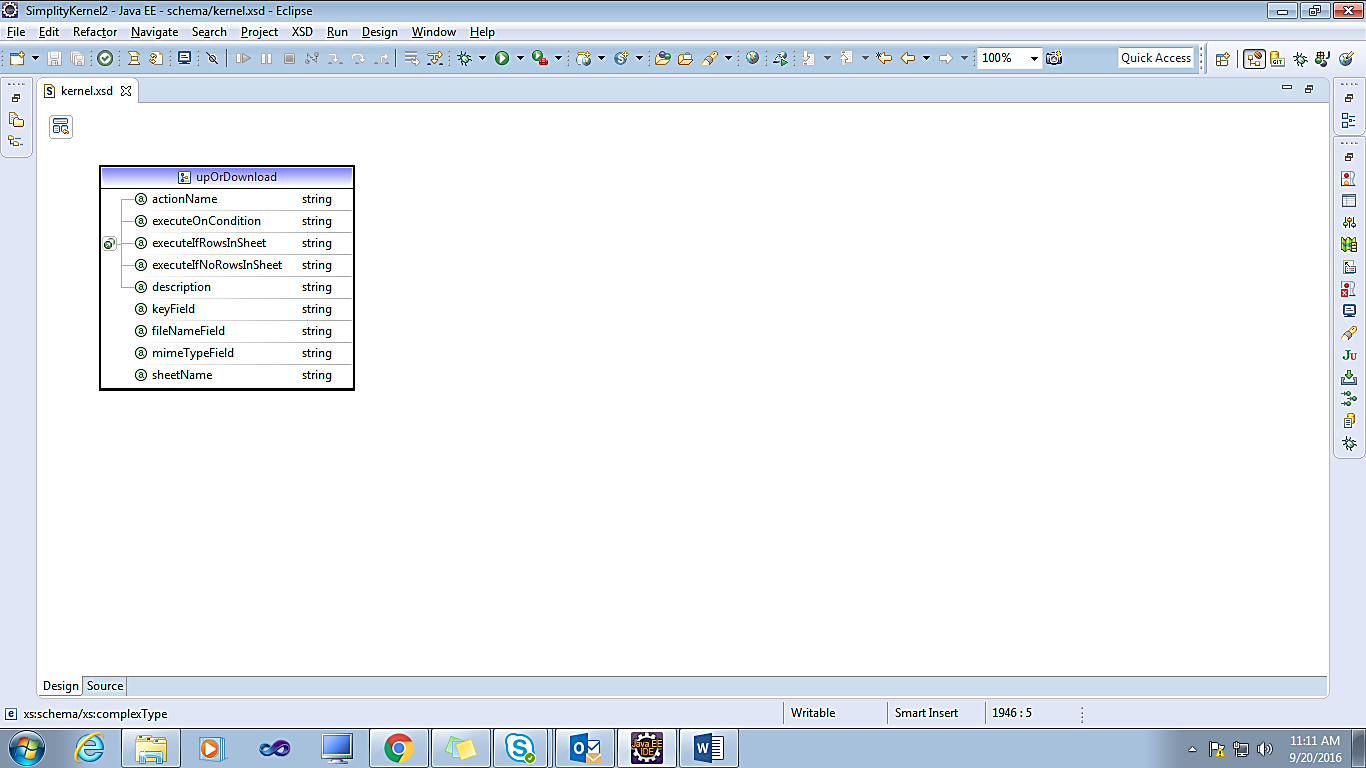
#### **2.23 Upload**

Upload a file from temp storage to permanent storage and change the file key/token.

<upload keyField=*"key"* fileNameField=*"name"* mimeTypeField=*"mime"* description=*"We save the file that was uploaded with key."* />

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| keyField | Field/column name that has the key for the file | Required |
| fileNameField | Field to which file name is to be copied to. This is optional | Optional |
| mimeTypeField | Field to which mime-type of this file is to be copied to | Optional |
| sheetName | Sheet name in case this action is for all rows of a sheet | Optional |

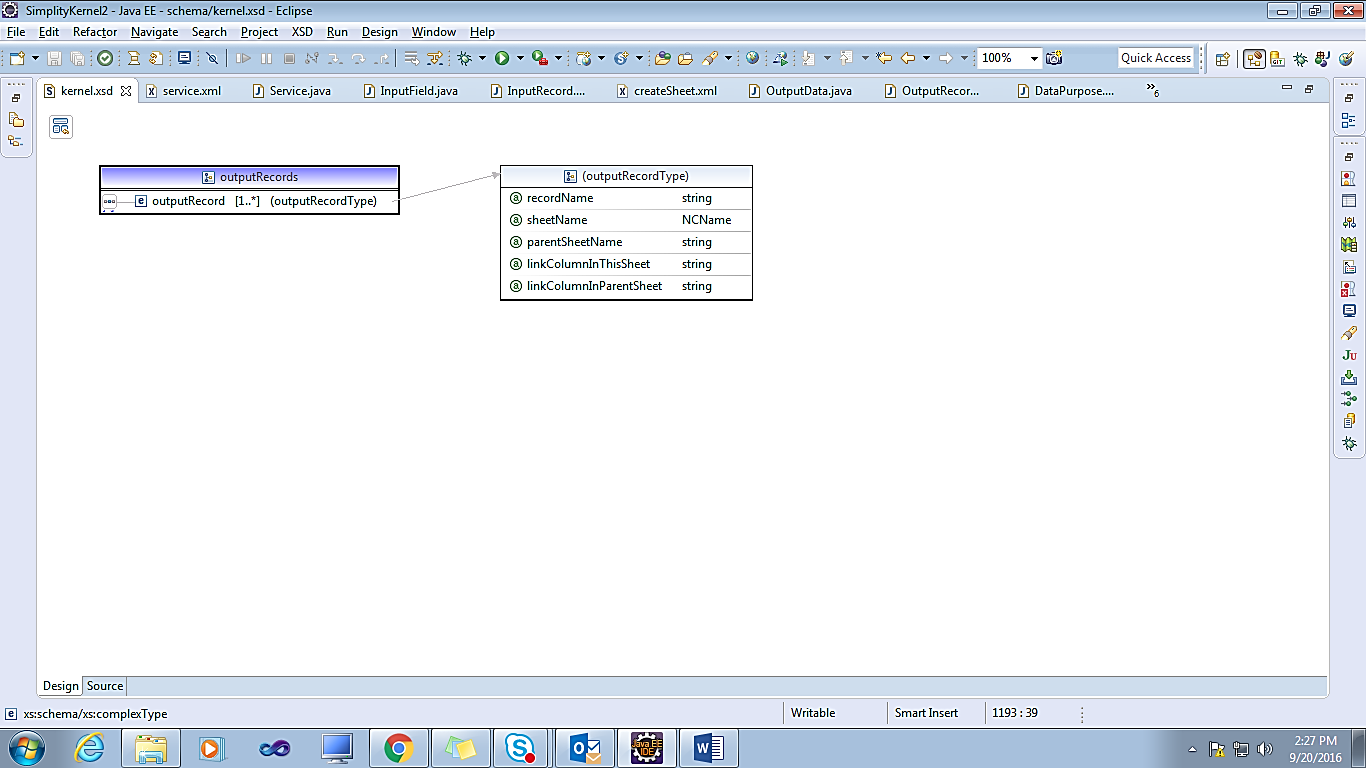


### **3. OutputData**

Component that specifies what inputs are expected

|  |
| --- |
| <outputData>  <outputRecords>  <outputRecord sheetName=*"newsheet"* />  </outputRecords>  </outputData> |

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| sheetName | Sheet into which output should get into. If null, then recordName should be specified are output | Optional |
| recordName | If you want all fields form a record to be output, this is a convenient, rather than listing all fields in fields list | Optional |
| parentSheetName | If this sheet has child rows for a parent, we have to output hierarchical data. | Optional |
| linkColumnInThisSheet | And we need to know how to identify child rows for a parent row. we use common columns in the two sheets to link them | Optional |
| linkColumnInParentSheet | Required if parentSheetName is specified. Column  in parent sheet that has the primary key to parent sheet,  which is used to match with parentKey in child sheet for  embedding. | Optional |



## **Application:**

Configure this application

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"no"*?>  <application  xmlns=*"http://www.simplity.org/schema"*  applicationId=*"simplityTutorial"*  dbVendor=*"mysql"*  dbDriverClassName=*"com.mysql.jdbc.Driver"*  connectionString=*"jdbc:mysql://localhost:3306/test?user=root"*  sendTraceToClient=*"true"*  logSqls=*"true"*>  </application> |



**Attributes**

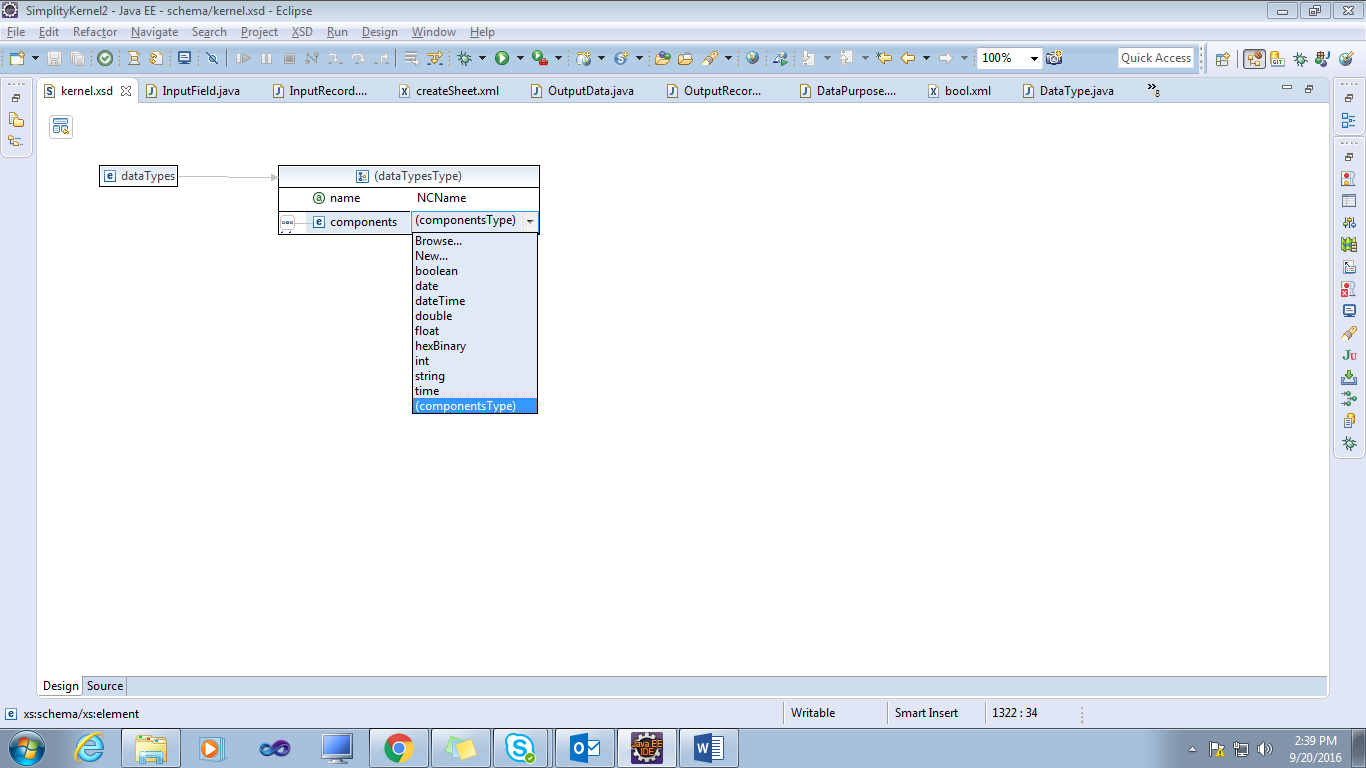
|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| applicationId | Unique name of this application within a corporate. This may be used as identity while trying to communicate with other applications within the corporate cluster | Required |
| sendTraceToClient | Should we send server trace to client? | Optional |
| cacheComponents | Do we cache components as they are loaded | Optional |
| dbVendor | The database vendor we are using | Optional |
| **DBVendor:** mysql, postgre, oracle, mssql, s2 |
| connectionString | For connecting to data base, we either use connection string with driver class name, or use dataSource. Connection string overrides. | Optional |
| dbDriverClassName | The database driver to use | Optional |
| dataSourceName | Data source name to be used to look-up in JNDI for dataSource | Optional |
| loginServiceName | The service identifier used to perform login. If this is null, we simulate a dummy login where loginId is accepted as userId | Optional |
| logoutServiceName | The service identifier used to perform logout action. No action is taken if this is left as null | Optional |
| userIdIsNumber | Does this project use an integer as user id? (default is string) | Optional |
| sessionHelperClassName | Session helper is a class that implements org.simplity.http.WebSessionHelper interface. Simplity uses an instance of this class to manage session variables. Session management is also associated with login requirements. Simplity provides three default classes. HelperForNoLogin during development if session variables are not used, DefaultHelper that uses a simple session management, and PassiveHelper when this application uses Simplity only as a module, and session is not managed by Simplity. You may write your session manager based on the application design | Optional |
| cacheManagerClassName | Response for certain services can be cached. Also, during development, we may want to fake some services with a pre-determined response. Use a class that implements org.simplity.service.CacheManager interface | Optional |
| accessControllerClassName | Class that decides whether a userId be served a given service | Optional |
| exceptionListenerClassName | Way to wire exception to corporate utility | Optional |
| logSqls | Should the sqls that are executed be added to trace?? Required during development. Some corporates have security policy that requires you not log sqls | Optional |
| sessionParams | Session parameters |  |
| userIdNameInSession | Name of the session parameter that has the value for userId | Optional |
| schemaDetails | Some projects use multiple schema. In such a case, it is possible that a given service may use a schema other than the default. We have optimized the design for a frequently used default schema that is set for the user, and a few services use their own schema. Provide such schema with dataSource/connection | Optional |
| mediaStorageFolderPath | Simplity provides a rudimentary, folder-based system that can be used for Storing and retrieving attachments/media/files. If you want to use that, provide the folder that is available for the server instance | Optional |
| mediStorageAssistantClass | If you have an infrastructure for storing and retrieving attachments/media/files that you want to use, provide the class name that implements MediaStorageAssistant. A single instance of this class is re-used | Optional |

## **DataTypes:**

DataType describes the constraints on the range of values that a data-element/field can have

|  |
| --- |
| <dataTypes name=*"myTypes"* xmlns=*"http://www.simplity.org/schema"*>  <components>  <booleanDataType name=*"boolean"* valueList=*"0:False,1:True"*  messageName=*"invalidBoolean"* />  <booleanDataType name=*"active"* valueList=*"0:Inactive,1:Active"*  messageName=*"invalidBoolean"* />  <booleanDataType name=*"yesNo"* valueList=*"0:No,1:Yes"* />  <dateDataType name=*"date"* maxDaysIntoFuture=*"730000"*  maxDaysIntoPast=*"3650000"* />  <numericDataType name=*"decimal3\_2"* maxValue=*"100"*  minValue=*"0"* nbrFractionDigits=*"2"* />  <numericDataType name=*"decimal4\_1"* maxValue=*"10000"*  minValue=*"0"* nbrFractionDigits=*"1"* />  <numericDataType name=*"rating"* maxValue=*"5"* minValue=*"0"*  nbrFractionDigits=*"2"* />  <numericDataType name=*"id"*  description=*"Internally generated number by the system as primary key for table"*  maxValue=*"99999999999999"* messageName=*"invalidId"* minValue=*"0"* />  <numericDataType name=*"number"* minValue=*"0"*  description=*"Non-negative whole number with a max of 14 digits is accepted."*  maxValue=*"99999999999999"* messageName=*"invalidNumber"*  nbrFractionDigits=*"0"* />  </components>  </dataTypes> |

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| name | Data type is identified by its name that is unique across all modules in an application | Required |



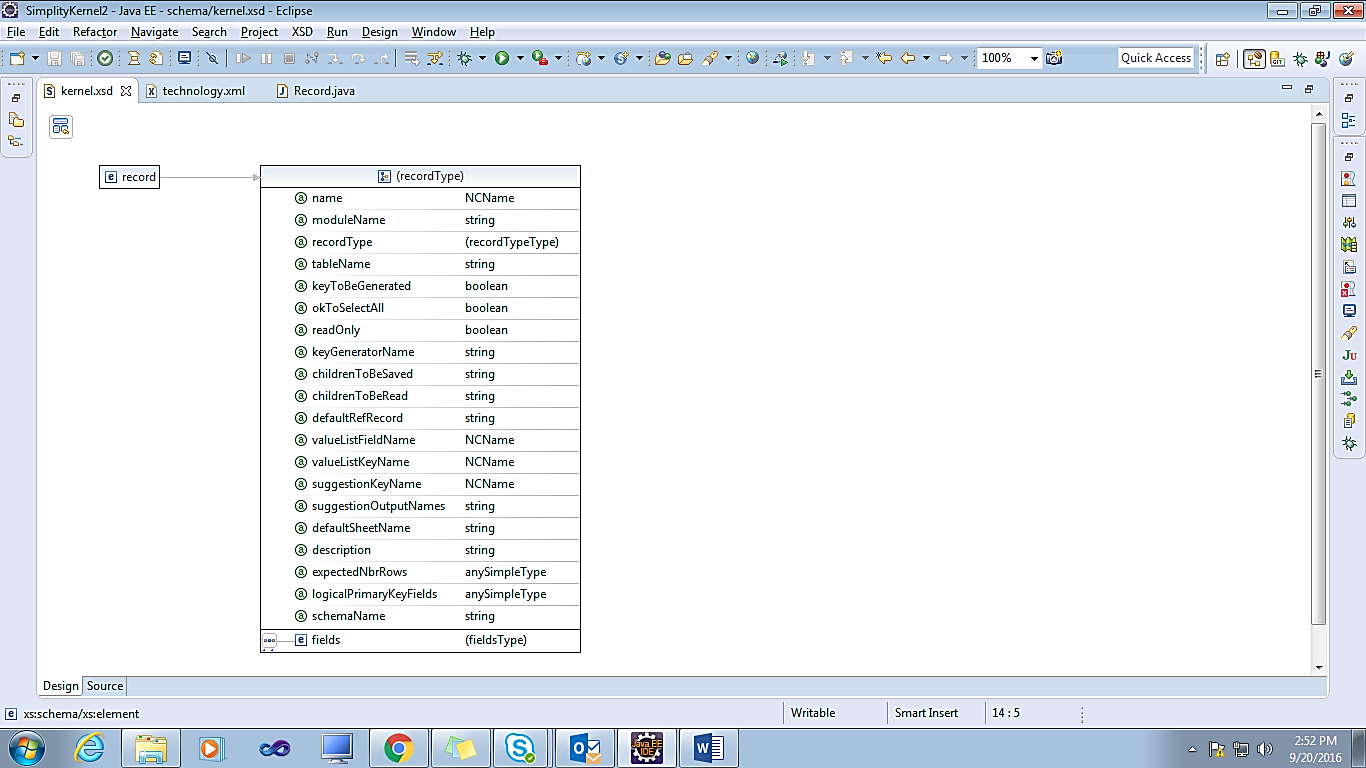
## **Record:**

This is the main part of our data model. Every piece of data that the application has to keep as part of "system of records" must be modeled into this. Data structures that are used as input to, or output from a service are modeled as records as well. It is common industry practice to have "physical data model" and "logical data model" for proper understanding. We encourage such designs before recording them using the Record concept. Record is a good candidate to represent any data structure that is used across components, even if it is not persisted.

|  |
| --- |
| <record xmlns=*"http://www.simplity.org/schema"* keyToBeGenerated=*"true"*  name=*"technology"* tableName=*"misuser.MWKMstTechnologies"* moduleName=*"xelBatch"*  okToSelectAll=*"true"* valueListFieldName=*"technology"*>  <fields>  <field name=*"technologyId"* columnName=*"intTechId"* label=*"Technology"*  dataType=*"number"* fieldType=*"primaryKey"* />  <field name=*"technology"* columnName=*"txtTechName"* label=*"Technology"*  dataType=*"text100"* />    </fields>  </record> |

**Attributes**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| Name | Name of this record/entity, as used in application | Required |
| moduleName | Module name + name would be unique for a component type within an application | Optional |
| recordType | type of this record | Optional |
| **Record Type:** storage, view, structure, output |
| tableName | name of the rdbms table, if this is either a storage table, or a view that is to be defined in the rdbms | Optional |
| keyToBeGenerated | Has this table got an internal key, and do you want it to be managed automatically? | Optional |
| okToSelectAll | If this table is expected to have large number of rows, we would like to protect against a select with no where conditions. Of course one can always argue that this is no protection, as someone can easily put a condition like 1 = 1 | Optional |
| childrenToBeRead | Child records that are to be read whenever a row from this record is read. | Optional |
| childrenToBeSaved | Child records to be saved along with this record. operations for this record | Optional |
| Fields | fields that this record is made-up of | Optional |
| valueListFieldName | In case this is a table that supplies key-value list for drop-downs, then we use primary key as internal key. Specify the field to be used as display value. key of this table is the internal value | Optional |
| valueListKeyName | Relevant only if valueListFieldName is used. If the list of values need be further filtered with a key, like country code for list of state, specify the that field name. | Optional |
| defaultSheetName | What is the sheet name to be used as input/output sheet. (specifically used in creating services on the fly) | Optional |
| keyGeneratorName | If keys are to be generated, we may have a sequence, as in oracle. Null implies that RDBMS auto-increments and we are not to set any value to this column | Optional |
| suggestionKeyName | if this record is used for a suggestion service, field that is used for search | Optional |
| suggestionOutputNames | what fields do we respond back with for a suggestion service | Optional |
| readOnly | Is this record only for reading? | Optional |
| schemaName | If this application uses multiple schemas, and the underlying table of this record belongs to a schema other than the default, then specify it here, so that the on-the-fly services based on this record can use the right schema. | Optional |



## **SQL:**

A prepared statement with which to interact with the data base.

|  |
| --- |
| <sql name=*"Employees"* sqlType=*"singleSelect"* xmlns=*"http://www.simplity.org/schema"*  preparedStatement=*"Select lastName From Employees where employeeNumber = ?"*  moduleName=*"example"*>  <inputParameters>  <sqlParameter dataType=*"id"* name=*"employeeNumber"*  defaultValue=*"1056"* />  </inputParameters>  <outputParameters>  <sqlParameter dataType=*"text"* name=*"lastName"* />  </outputParameters>  </sql> |

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Use |
| Name | Unique within a module | Required |
| moduleName | Module + name is unique | Optional |
| preparedStatement | Prepared statement. | Optional |
| sqlType | Purpose of this sql/procedure. Important to specify whether you are expecting output, and if so whether we may get more than one rows | Required |
| SQL Type: singleSelect, multiSelect, update |
| inputParameters | Input parameters. In the same order as in prepared statement. | Optional |
| outputParameters | Output parameters if this is a select sql. Alternately, you may specify an output record. You should not specify both. | Optional |

