How-to use your own MOColumn subclasses to control column behaviour of tables

This page describes how the SNMP4J-Agent MOFactory concept can be used to apply customised columnar objects to SNMP table implementations.

Avoid Using Request Data to Process Requests

Although this article explains how to access SNMP request data while processing sub-requests, it is not recommended to do so without reasons.

`SubRequests` can be SNMPv1, v3, or AgentX requests, for example. So you should not rely on a particular content in the sub-request that might not be available in all cases.

Step-by-step guide

This guide assumes that you have generated your MIB instrumentation code with AgenPro, but you can do the same with manual written code too.

1. Generate the MIB module instrumentation code with AgenPro and activate **factoryColumn** property (set its value to **yes**) for your entire MIB module (or all affected tables). This will create the code to create `MOColumn` instances for read-only columns using the `MOFactory` provided to your MIB module.
   This step is important, because we will create our own `MOFactory` implementation to create our own `MOColumn` subclasses.
   The generated code for each table will look like (sample taken from the `Snmp4jAgentTutorialMib.java`):

```
  @SuppressWarnings(value={"unchecked"})
  private void createSnmp4jAgentTutorialFileTreeEUEntry(MOFactory moFactory) {
    // Index definition
    snmp4jAgentTutorialFileTreeEUEntryIndexes =
      new MOTableSubIndex[] {
      moFactory.createSubIndex(oidSnmp4jAgentTutorialFileTreeEUIndex,
                               SMIConstants.SYNTAX_OBJECT_IDENTIFIER, 0, 128)    };
    snmp4jAgentTutorialFileTreeEUEntryIndex =
      moFactory.createIndex(snmp4jAgentTutorialFileTreeEUEntryIndexes,
                            false,
                            new MOTableIndexValidator() {
      public boolean isValidIndex(OID index) {
        boolean isValidIndex = true;
     //--AgentGen BEGIN=snmp4jAgentTutorialFileTreeEUEntry::isValidIndex
     //--AgentGen END
        return isValidIndex;
      }
    });
    // Columns
    MOColumn[] snmp4jAgentTutorialFileTreeEUEntryColumns = new MOColumn[10];
    snmp4jAgentTutorialFileTreeEUEntryColumns[idxSnmp4jAgentTutorialFileTreeEUPath] =
      moFactory.createColumn(colSnmp4jAgentTutorialFileTreeEUPath,
                             SMIConstants.SYNTAX_OCTET_STRING,
                             moFactory.createAccess(MOAccessImpl.ACCESSIBLE_FOR_READ_ONLY),
                             tcModuleSNMPv2Tc,
                             tcDefDisplayString);
    snmp4jAgentTutorialFileTreeEUEntryColumns[idxSnmp4jAgentTutorialFileTreeEUType] =
      moFactory.createColumn(colSnmp4jAgentTutorialFileTreeEUType,
                             SMIConstants.SYNTAX_INTEGER,
                             moFactory.createAccess(MOAccessImpl.ACCESSIBLE_FOR_READ_ONLY));
    snmp4jAgentTutorialFileTreeEUEntryColumns[idxSnmp4jAgentTutorialFileTreeEUCreationTime] =
      moFactory.createColumn(colSnmp4jAgentTutorialFileTreeEUCreationTime,
                             SMIConstants.SYNTAX_OCTET_STRING,
                             moFactory.createAccess(MOAccessImpl.ACCESSIBLE_FOR_READ_ONLY),
                             tcModuleSNMPv2Tc,
                             tcDefDateAndTime);
    snmp4jAgentTutorialFileTreeEUEntryColumns[idxSnmp4jAgentTutorialFileTreeEULastModified] =
      moFactory.createColumn(colSnmp4jAgentTutorialFileTreeEULastModified,
                             SMIConstants.SYNTAX_OCTET_STRING,
                             moFactory.createAccess(MOAccessImpl.ACCESSIBLE_FOR_READ_ONLY),
                             tcModuleSNMPv2Tc,
                             tcDefDateAndTime);
    snmp4jAgentTutorialFileTreeEUEntryColumns[idxSnmp4jAgentTutorialFileTreeEULastAccessed] =
          moFactory.createColumn(colSnmp4jAgentTutorialFileTreeEULastAccessed,
                             SMIConstants.SYNTAX_OCTET_STRING,
                             moFactory.createAccess(MOAccessImpl.ACCESSIBLE_FOR_READ_ONLY),
                             tcModuleSNMPv2Tc,
                             tcDefDateAndTime);
    snmp4jAgentTutorialFileTreeEUEntryColumns[idxSnmp4jAgentTutorialFileTreeEUSizeInBytes] =
moFactory.createColumn(colSnmp4jAgentTutorialFileTreeEUSizeInBytes,
                             SMIConstants.SYNTAX_GAUGE32,
                             moFactory.createAccess(MOAccessImpl.ACCESSIBLE_FOR_READ_ONLY));

...
}
```

2. As you can see in the above code, the `moFactory` member of the generated MIB module class is used to create the actual `MOColumn` instances. To create our custom MOColumn instances we need our own MOFactory instance first:

```
myMOFactory = new DefaultMOFactory() {

        @Override
        public <V extends Variable> MOColumn<V> createColumn(int columnID, int syntax, MOAccess access,
                                                             V defaultValue, boolean mutableInService) {
                  return new MOMutableColumn<V>(columnID, syntax, access, defaultValue,
mutableInService) {
                        @Override
                        public V getValue(MOTableRow row, int column, SubRequest subRequest) {
                                if (subRequest.getScope() instanceof MOContextScope) {
                                        MOContextScope moContextScope = (MOContextScope)subRequest.
getScope();
                                          if ("myContext".equals(moContextScope.getContext())) {
                                            ...
                                          }
                                }
                                return (V) row.getValue(column);
 }
                }
        }

        public <V extends Variable> MOColumn<V> createColumn(int columnID, int syntax, MOAccess access,
                                                                                  V defaultValue,
boolean mutableInService,
                                                                                   String
tcModuleName, String textualConvention) {
                  TextualConvention<V> tc = getTextualConvention(tcModuleName, textualConvention);
                  if (tc != null) {
// Here you might need to create a MOColumn proxy object to apply your interception code
 // to columns based on textual conventions too:
                    return tc.createColumn(columnID, syntax, access, defaultValue, mutableInService);
                  }
                  return createColumn(columnID, syntax, access, defaultValue, mutableInService);
        }
}
```

3. Now you need to use the created MOFactory with your instance(s) of the MIB module. Basically you have to change the MIB module creation to the following. In fact, there are several locations where this can be done when using AgenPro generated code:

   1. When creating the MIB module (see below)

   2. When creating the `Module` instance that encapsulates all MIB modules generated for an agent

   3. By overriding the `Agent.getFactory()` method and returning your custom `MOFactory`

```
Snmp4jAgentTutorialMib snmp4jAgentTutorialMib = new Snmp4jAgentTutorialMib(myMOFactory);
```

4. For GET type requests overriding the `MOColumn.getValue` method is sufficient. If you also want to control the processing of SET requests, you need also to override the following methods defined in `MOMutableColumn` as needed:
   a. `prepare`
   b. `commit`
   c. `undo`
   d. `cleanup`

Related articles

- How-to use your own MOColumn subclasses to control column behaviour of tables
- How to add row to a table?