Implementing an Activity

EPCC, University of Edinburgh
Tom Sugden (tom@epcc.ed.ac.uk)

First International Summer School on Grid Computing, Vico Equense, Italy

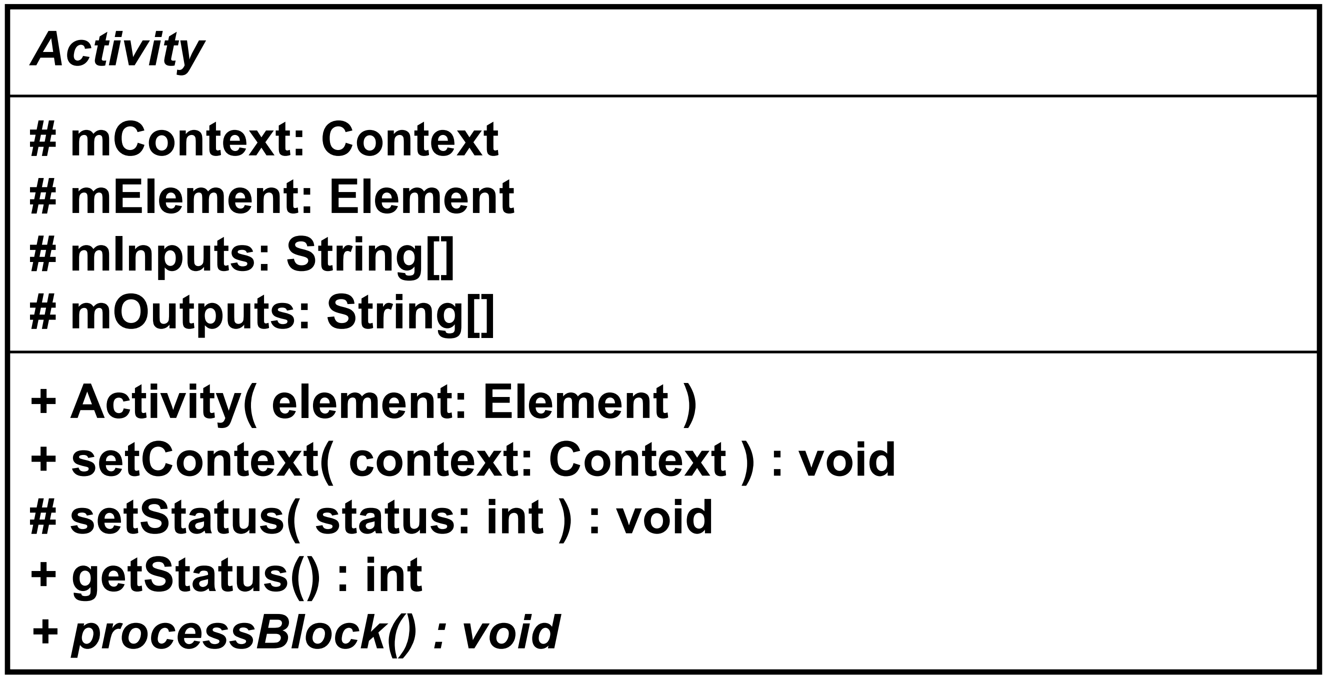
Motivation and Outline

- OGSA-DAI R3 provides many activities, however, it may sometimes be necessary to develop additional activities
  - To support different query languages (XQuery)
  - To perform different kinds of transformation (STX)
  - To deliver results using a different mechanism (WebDAV)

- Talk outline
  - Examine the abstract Activity class.
  - Walk-through the XPathStatementActivity implementation

The abstract Activity class

- All Activity implementations extend the abstract Activity class


Activity
# mContext: Context
# mElement: Element
# mInputs: String[]
# mOutputs: String[]
+ Activity(element: Element)
+ setContext(context: Context) : void
# setStatus(status: int) : void
+ getStatus() : int
+ *processBlock() : void*

The Lifecycle of an Activity

- There are three stages to the life cycle of an Activity:
  1. Construction
  2. Initialisation
  3. Processing and Output

- How do these correspond to the abstract Activity class?

Construction

- An Activity is constructed using a DOM Element
  - Conforms to the schema in the activity map.
  - Existing schema location: **`schema/xsd/activities/`**
- The Element will be parsed to retrieve:
  - Input names
  - Output names
  - Configuration information (SQL expression, etc.)
- Must publish the names of its inputs and outputs
  - Stored in the **`mInputs`** and **`mOutputs`**
  - Accessed via **`getInputs`** and **`getOutputs`**

Initialisation

- An Activity is initialised using the **`setContext(Context)`** method.
- Performs context dependent initialisation
  - Obtaining references to the BlockReaders, BlockWriters and User Credentials for easy access during the processing/output stage.
- The Engine initialises activities and guarantees that the inputs and outputs published during construction are contained in the activity context.

Retrieving Objects from the Context

- Objects can be retrieved from the Context using the **`get`** method:

```
…
BlockReader myInput = (BlockReader) context.get(
      EngineImpl.PIPES + mInputs[0] );
BlockWriter myOutput = (BlockWriter) context.get(
      EngineImpl.PIPES + mOutputs[0] );
…
```

- Key constants are stored in
  – uk.org.ogsadai.service.OGSADAIConstants
  – uk.org.ogsadai.porttype.gds.engine.EngineImpl

Processing and Output

- An activity is expected to operate in an iterative fashion. For example, a simple iteration might be:
  - Read block from input
  - Process block in some way
  - Write block to output
- A call to the **`processBlock`** method is a request from the Engine for the Activity to provide a block of output.
- Activity status is used by the Engine to determine when processing and output is complete.

Activity Status

- An Activity must track its own status using the **`setStatus`** method
- There are 4 states:
  - **UNSTARTED**
    - Set before the **`processBlock`** method has been invoked.
  - **PROCESSING**
    - Set the first time the **`processBlock`** method is invoked and remains set until the processing is complete or there is an error.
  - **COMPLETE**
    - Set when the processing is complete and there are no more blocks to output.
  - **ERROR**
    - Set when there is a problem of some kind during the processing of a block.

XPathStatementActivity

- The activity element (excerpt from perform doc)

```
<xPathStatement name="myActivity">
  <collection>musicians/folksingers</collection>
  <namespace prefix="c">
    http://ogsadai.org.uk/contacts
  </namespace>
  <expression>/c:entry/c:address</expression>
  <resourceSetStream name="myActivityOutput"/>
</xPathStatement>
```

- Passed as a DOM Element to the XPathStatementActivity constructor

XPathStatementActivity Constructor

- Parses the xPathStatement element
  - Extract the collection name, resource ID, namespace bindings, query expression and output name.

- Publishes the Activity input and output names

```
mInputs = new String[0]; // no inputs to activity
mOutputs = new String[] {
    ElementParser.parseChildElementAttribute(
        element,
        Constants.ELEMENT_RESOURCE_SET_STREAM,
        Constants.ATTRIBUTE_NAME )};
```

XPathStatementActivity setContext method

- Retrieves a reference to the Data Resource Implementation, output Block Writer and User Credentials for easy access during the processing and output stage.

```
mDataResource =
    (XMLDBDataResourceImplementation) mContext.get(
        OGSADAIConstants.DATA_RESOURCE_IMPLEMENTATION );
mOutput =
    (BlockWriter) mContext.get(
       EngineImpl.PIPES + mOutputs[0] );
mUserCredentials =
    (String) mContext.get(
       OGSADAIConstants.GDS_USERCERTDN );
```

XPathStatementActivity processBlock method

- The first time the method is invoked
  - Status is set to **PROCESSING**
  - Collection is retrieved from **`mDataResource`**
  - XPath expression is executed, generating results.
  - Collection is returned to **`mDataResource`**
  - First block of result data is put onto the output.
- Each subsequent invocation
  - Checks whether there are any more result blocks
  - If so, puts the next result block onto output.
  - If not, the status is set to **COMPLETE**
- If any exceptions occur
  - Status is set to **ERROR**

XPathActivityStatement processBlock method cont.

```
try {
    if ( getStatus()==StatusMessage.UNSTARTED ) {
        setStatus(StatusMessage.PROCESSING);
        performStatement();
    }
    if ( mResults.hasNext() ) {
        mOutput.put(mResults.next());
    }
    else {
      setStatus(StatusMessage.COMPLETE);
    }
}
catch (Exception e) {
  setStatus(StatusMessage.ERROR, e);
}
```

Conclusion

- The abstract Activity class is straight-forward to implement.
- A more detailed “How to Write an Activity” document is being written for distribution from the OGSA-DAI web site.
- If you develop any interesting new activities, please send them to us! We may host them on the OGSA-DAI web site.