Taverna Platform

What is done to run a workflow?
Command Line Tool

- This presentation gives a brief introduction to the capabilities of Taverna Platform
- Uses the Command Line Tool as an example
- Builds on
  - SCUFL2
  - Bundle
WorkflowBundle

- Read in the workflow
  
  URL workflowURL = ...  
  
  WorkflowBundle workflowBundle = workflowBundleIO.readBundle(workflowURL, null);  
  
  Workflow workflow = workflowBundle.getMainWorkflow();

- WorkflowBundleIO is an OSGi service for reading and writing workflows

- WorkflowBundle is a SCUFL2 container that holds all the information about the workflow

- Need the platform to run the workflow
The **RunService** is the main class in the Platform that looks after workflows runs and how they can be run
- The **RunService** is set up as part of the OSGi / Spring environment

The available **ExecutionEnvironments** are obtained from the RunService

```java
Set<ExecutionEnvironment> executionEnvironments = runService.getExecutionEnvironments();
```

// Then choose an ExecutionEnvironment

Different **ExecutionEnvironments** can run different types of activity and cope with different ways of executing the workflow
Use the concept of a Research Object to associate the input data values with the input ports

```java
InputsHandler inputsHandler = new InputsHandler();

Map<String, InputWorkflowPort> portMap =
    new HashMap<String, InputWorkflowPort>();

for (InputWorkflowPort port : workflow.getInputPorts()) {
    portMap.put(port.getName(), port);
}

Bundle inputs =
    inputsHandler.registerInputs(portMap, commandLineOptions, null);
```
RunProfile

- **RunProfile** gathers together the information about how you want a workflow to be run

  ```java
  RunProfile runProfile = new RunProfile(executionEnvironment, workflowBundle, inputs);
  ```

- Where to run
- What to run
- With what values
Start the run and look at it

String runId = runService.createRun(runProfile);

runService.start(runId);

• **RunService** internally creates a Run

• Need to be able to check its status

  WorkflowReport report = runService.getWorkflowReport(runId);

  while (!workflowFinished(report)) {
      try {
          Thread.sleep(500);
      } catch (InterruptedException e) {
  
          ...  

          return 1;
      }
  }
WorkflowReport

- Part of the platform
- Contains
  - ActivityReport and
  - ProcessorReport
- Holds information about the state of the activities, processors and the overall workflow
- Is polled by the workbench or command line tool
- Similar to T2’s captured provenance
- Can be exposed as JSON
(Data) Bundle - again

- **Bundle** is also used to associate the results of a workflow run with the ports

```java
Bundle finishedBundle = runService.getDataBundle(runId);
Path outputs = DataBundles.getOutputs(finishedBundle);

for (OutputWorkflowPort outputWorkflowPort : workflowOutputPorts) {
    String workflowOutputPortName = outputWorkflowPort.getName();
    Path output = DataBundles.getPort(outputs, workflowOutputPortName);
    ...
}
```

- **DataBundles** is a utility class to help handle (Data)Bundles
Usage summary

- **SCUFL2**
  - Read the *WorkflowBundle*

- **Platform**
  - **RunService** – OSGi service to handle runs
  - **ExecutionEnvironment** – where/how workflows can be run
  - **RunProfile** – information to make a run
  - **Run** – the actual Run
  - **StatusReport** – state etc. of a workflow run

- **Bundle**
  - Holds the input values
  - Populated with output values