Building An Application Using Procedures

Exercise Objectives

In this exercise, you will:

- Add a Procedure to the example application project
- The Procedure returns either the aggregates timeseries sentiment data or specific sentences for a given sentiment
- Build, deploy and run the Procedure
- Retrieve results from the DataSets using the Procedure

Exercise Steps

- Add imports
- Modify the `ApplicationSpecification`
- Add the Procedure
- Build and deploy
- Run the application and use the Dashboard to retrieve queries

Application Specification

Change the ApplicationSpecification, replacing `.noProcedure()` with:

```
.withProcedures()
  .add(new SentimentAnalysisProcedure())
```

Add these imports:

```
import com.continuuity.api.procedure.ProcedureResponse;
import com.continuuity.api.procedure.ProcedureSpecification;
import com.google.common.collect.Maps;
import java.util.List;
import java.util.concurrent.TimeUnit;
import com.continuuity.api.ResourceSpecification;
```

`ResourceSpecification` sets the minimum amount of memory (512MB) and cores (1) used by an instance

---

Add Procedure to `SentimentAnalysisApp`

```
public static class SentimentAnalysisProcedure extends AbstractProcedure {

  @UseDataSet("sentiments")
  private Table sentiments;

  @UseDataSet("text-sentiments")
  private SimpleTimeseriesTable textSentiments;

  @Handle("aggregates")

  @Handle("sentiments")

  @Override
  public ProcedureSpecification configure() {
    return ProcedureSpecification.Builder.with()
      .setName("sentiment-query")
      .setDescription("Sentiments Procedure")
      .withResources(ResourceSpecification.BASIC)
      .build();
  }
}
```

Implement `@Handle("aggregates")`

```
@Handle("aggregates")
public void sentimentAggregates(ProcedureRequest request, ProcedureResponder response)
  throws Exception {
  Row row = sentiments.get(new Get("aggregate"));
  Map<byte[], byte[]> result = row.getColumns();
  if (result == null) {
    response.error(ProcedureResponse.Code.FAILURE, "No sentiments processed.");
    return;
  }
  Map<String, Long> resp = Maps.newHashMap();
  for (Map.Entry<byte[], byte[]> entry : result.entrySet()) {
    resp.put(Bytes.toString(entry.getKey()), Bytes.toLong(entry.getValue()));
  }
  response.sendJson(ProcedureResponse.Code.SUCCESS, resp);
}
```

Implement `@Handle("sentiments")`

```
@Handle("sentiments")
public void getSentiments(ProcedureRequest request, ProcedureResponder response)
  throws Exception {
  String sentiment = request.getArgument("sentiment");
  if (sentiment == null) {
    response.error(ProcedureResponse.Code.CLIENT_ERROR, "No sentiment sent.");
    return;
  }

  long time = System.currentTimeMillis();
  List<SimpleTimeseriesTable.Entry> entries =
    textSentiments.read(sentiment.getBytes(Charsets.UTF_8),
                        time - TimeUnit.MILLISECONDS.convert(1, TimeUnit.DAYS),
                        time);

  Map<String, Long> textTimeMap = Maps.newHashMapWithExpectedSize(entries.size());
  for (SimpleTimeseriesTable.Entry entry : entries) {
    textTimeMap.put(Bytes.toString(entry.getValue()), entry.getTimestamp());
  }
  response.sendJson(ProcedureResponse.Code.SUCCESS, textTimeMap);
}
```

Build, Deploy and Run

- Build the App using `mvn clean package`
- If Reactor is running, stop or reset it; otherwise start Reactor
- Deploy the App by dragging and dropping
- Use the Continuuity Reactor Dashboard to start and query the Procedure
- Test the Procedure `sentiments` by sending the parameters `{"sentiment":"positive"}` (or `neutral` or `negative`)
- Test the Procedure `aggregates` (no parameters) to see how many results have been obtained
- You may need to resend the sentences of earlier exercises if there is no data in the DataSets

Exercise Summary

You should now be able to:

- Add a Procedure to an application project
- Build, deploy and run the Procedure
- Retrieve results from DataSets using Procedures

Exercise Completed

Chapter Index