cappatec

Core Engine

Activity "Hello, World!"

Version 1, February 2013

Applicable for Core Engine 1.0 P, Core Engine 1.0 A, Core Engine 1.1 A

Authors:

Fabian Knirsch, BSc
Oliver Langthaler, BSc

Table of Contents

About this tutorial

Introduction

This tutorial is intended to provide a simple but comprehensive example on how to create and deploy an Activity for Cappatec Core Engine. It is designed for developers that do have experience with Java or similar programming languages and want to start developing Activities for Cappatec Core Engine.

Philosophy

Cappatec Core Engine is a powerful and high performance process server. It provides scalability on multiple levels, runs on different hardware and software platforms and provides a powerful toolset for the development of business services. Additionally, Core Engine offers many advanced features such as caching, scheduling and Single Sign On.

As the Core Engine is a general purpose process server designed to be able to host any kind of corporate process and activity, it can be tailored to individual needs and requirements. Activities are lightweight modules which can be developed quickly and perform specific tasks such as data transmission, transposition and enhancement. They can be tailored to fulfill any requirement and may even be added during runtime.

Prerequisites

For this tutorial you need:

i) A running instance of Cappatec Core Engine 1.x, non-productive system recommended.
ii) A Cappatec Core Engine user with access to Activities in the Core-Management process.
iii) A Core Engine Process for development purposes named *Test*.
iv) A database for development purposes named *Test* containing a dummy data table named *test*.
v) A client to test your Activity, Cappatec Client recommended.

Further Reading

For deeper understanding of the R×XML interface, reading of Cappatec R×XML Specification is recommended.

Environment setup

This chapter describes how to set up an integrated development environment (IDE) for writing Cappatec Core Engine activities.

This instruction is designed for an Eclipse based development on Windows XP/7/8 and most Linux distributions. Other platforms and operating systems may be possible, but are not supported.

Resources

Download the following packages:

- Java 7 JDK
  You will need a Java 7 JDK, e.g.:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html

- Eclipse
  Either Eclipse Classic (recommended), Eclipse IDE for Java Developers or Eclipse for Java EE Developers are most suitable for Activity development.
  http://www.eclipse.org/downloads/

- Cappatec Core Library
  Download the version that matches your Core Engine. Core Library distributions are generally upward compatible. With older releases, however, you may not have access to all features of your Core instance.
  http://www.cappatec.com/

Install JDK and set classpaths as necessary. Unzip or install Eclipse to your working directory and create an empty workspace. Place the Core Library JAR-file in this directory.

Create a new empty Java Project and make sure you are using Java 7. Add the Core Library JAR to your project.

Processes and Activities

Processes are represented as packages and Activities are represented as Java classes. To use a certain process (e.g. *Test*), create a new package named *process.Test*. To create an Activity named *HelloWorld* in Process Test, create a new Java class named *HelloWorld.java* and place it in the package *Test*.

Deployment

Please be aware that it is not possible to run Activities within the development environment. It is necessary to deploy the Activity to a running instance of Core Engine first.

Deployment is described in greater detail below. Basically, it comprises the following two steps:

i) Copying the compiled Activity to a Core Engine instance
ii) Registering the Activity and assigning access privileges

If an Activity is already registered, e.g. when making changes to an existing Activity, step (ii) can be skipped and the new class file may simply be copied to the Core Engine.

"Hello, World!"

This chapter describes how to create a Core Engine Activity that displays the string "Hello, World!" when being queried by a data request.

Setup

Create a new Java class "HelloWorld1" that derives from the class Activity as provided in the Cappatec Core Library.

```
public class HelloWorld1 extends Activity
{
      …
}
```

Make sure the class is placed in the right package, e.g. process.Test and you use the appropriate package declaration as well as all the necessary imports, e.g. activityloader.Activity.

Request Handler

To handle a data request, it is necessary to define a specific method that will be invoked by the Core Engine upon any request. This method is defined in the abstract Activity class and needs to be overwritten.

```
@Override
public AbstractResponse onDataRequest(DataRequest request)
      throws Exception {
      …
}
```

In this example, we will not evaluate any arguments, but always return a data response with the hello-world-string. R×XML is a set oriented format. Therefore, the representation of scalar values (such as a single string) needs to be expressed in a table structure. We define the following table as the data response:

text
Hello, World!

The corresponding R×XML representation is:

```
<Response type="data">
      <Data>
            <Set>
                  <Description>
                        <Group>
```

```
                              <Column type="string">text</Column>
                        </Group>
                  </Description>
                  <Row>
                        <Group>
                              <Cell>Hello, World!</Cell>
                        </Group>
                  </Row>
            </Set>
      </Data>
</Response>
```

Using predefined classes and methods, this structure can easily be created in Java. In the data request method, the following code produces the required output:

```
// Data Response object
DataResponse dr = new DataResponse("HelloWorld1");

// Define Set for response
Set s = new Set();
dr.addSet(set);

// Define description header
Description d = new Description();
s.addDescription(d);

Group g1 = new Group();
d.addGroup(g1);

Column col = new Column("text", XMLDataType.STRING);
g1.addColumn(col);

// Define data body
Row r = new Row();
s.addRow(r);

Group g2 = new Group();
r.addGroup(g2);

Cell cel = new Cell();
c.setContent("Hello, World!");
g2.addCell(cel);

// Return Data Response object
return dr;
```

Deployment

To deploy your Activity, open Cappatec Client or a similar Core management tool and register the Activity (Add Activity) with the following properties:

- Name: HelloWorld1
- Process: Test
- Type: select

Build your Java class file (when using Eclipse as IDE, you will find an automatic build in the /bin directory in your workspace) and copy the class file to your running Core Engine instance. Make sure to copy the Activity to the same process that you used during development.

Activities can be deployed with Cappatec Client using the Deploy Activity. Select the Activity as registered above and choose your class file.

Finally, assign the activity or the entire Test process (if not yet done) to a group that your user has access to.

Testing

Run Cappatec Client or a similar client tool and navigate to your new Activity. If all steps have been completed correctly, the string "Hello, Core!" in a table with column header "text" should be displayed.

"Hello, Core!"

This chapter describes how to perform a simple database query and return a set of results.

Setup

Create a new Java class "HelloWorld2" that derives from the class Activity as provided in the Cappatec Core Library.

```
public class HelloWorld2 extends Activity
{
      @Override
      public AbstractResponse onDataRequest(DataRequest request)
            throws Exception {
            …
      }
}
```

We will again handle a data request without evaluating any arguments.

Data Retrieval

To query the database, it is necessary to acquire a database connection. Corresponding functionality is provided by the Activity parent class. The returning object provides many powerful database query methods. In this example, only Select will be used. It is not necessary to step through every single result row and build the data response object manually. Instead, the Transformer toolset may be utilized to get the resulting set.

```
// Acquire connection
Database db = super.getDatabase("test");

// Query database
DataSet ds = db.select("SELECT * FROM test.test");

// Transform result
DataResponse r = new DataResponse();
r.addSet(Transformer.DataFromResultSet(ds, true));

// Return Data Response object
return r;
```

Deployment

To deploy your Activity, open Cappatec Client or a similar Core management tool and register the Activity (Add Activity) with the following properties:

- Name: HelloWorld2
- Process: Test
- Type: select

Build your Java class file (when using Eclipse as IDE, you will find an automatic build in the /bin subdirectory of your workspace) and copy it to your running Core Engine instance. Make sure to copy the Activity to the same process that you used during development.

Activities can be deployed with Cappatec Client using the Deploy Activity. Select the Activity as registered above and choose your class file.

Finally, assign the activity or the entire Test process (if not yet done) to a group that your user has access to.

Testing

Run Cappatec Client or a similar client tool and navigate to your Activity. If all steps have been completed correctly, the contents of the table test in the database Test should be displayed.

Known Issues

The following provides a list of frequent and known issues and problems, probable causes and how to solve them.

213: Activity Not Found Error

This error may occur:

- When deploying your Activity with Deploy Activity
  For security reasons, the deployment of activities may be restricted, even when using the Deploy Activity in your home menu. Contact your Core Engine administrator for further assistance.
- When trying to open your Activity
  Your Activity has been registered but may not yet have been deployed. It is registered in the home Menu but not physically present. Deploy your activity.

Disclaimer and Contact Information

Disclaimer

The information herein is solely provided for informational and development purposes. No license to any intellectual property right is granted by this document or in connection with the usage of Cappatec products.

Cappatec assumes no liability whatsoever and disclaims any warranty relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or non-infringement. Under no circumstance shall Cappatec be liable for any direct, indirect, consequential, punitive, special or incidental damages. This includes, without limitation, damages for business interruption and damages for loss of information or profits arising out of the use or inability to use this document, even if Cappatec has been advised of the possibility of such an event.

Cappatec makes no representations or warranties concerning the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Cappatec does not guarantee any updates to the information contained within this document. Cappatec products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life unless specifically stated otherwise.

Contact Information

The Cappatec Core Engine developer team:

Fabian Knirsch, BSc fabian.knirsch@cappatec.com

Oliver Langthaler, BSc oliver.langthaler@cappatec.com

www.cappatec.com

Copyright Note