C:\checkout\openii3\project\java\source\org\openeai\implementations\services\egs\com
May 9, 2006 9:58:14 AM

```
 1  /*****************************************************************************
 2   $Source: /cvs/repositories/openii3/project/java/source/org/openeai/implementat
 3   $Revision: 1.2 $
 4  *****************************************************************************
 5
 6  /*************************************************************************
 7  This file is part of the OpenEAI sample, reference implementation,
 8  and deployment management suite created by Tod Jackson
 9  (tod@openeai.org) and Steve Wheat (steve@openeai.org).
10
11  Copyright (C) 2003 The OpenEAI Software Foundation
12
13  This program is free software; you can redistribute it and/or modify
14  it under the terms of the GNU General Public License as published by
15  the Free Software Foundation; either version 2 of the License, or
16  (at your option) any later version.
17
18  This program is distributed in the hope that it will be useful,
19  but WITHOUT ANY WARRANTY; without even the implied warranty of
20  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
21  GNU General Public License for more details.
22
23  You should have received a copy of the GNU General Public License
24  along with this program; if not, write to the Free Software
25  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
26
27  For specific licensing details and examples of how this software
28  can be used to implement integrations for your enterprise, visit
29  http://www.OpenEai.org/licensing.
30  */
31
32  package org.openeai.implementations.services.egs.commands;
33
34  // Core Java
35  import java.util.ArrayList;
36  import javax.jms.*;
37
38  // JDOM
39  import org.jdom.*;
40  import org.jdom.output.XMLOutputter;
41
42  // General OpenEAI foundation
43  import org.openeai.config.*;
44  import org.openeai.xml.*;
45  import org.openeai.jms.consumer.commands.*;
46  import org.openeai.jms.producer.*;
47  import org.openeai.layouts.EnterpriseLayoutException;
48  import org.openeai.moa.EnterpriseObjectSyncException;
49
50  // Objects and message objects from OpenEAI implementations
51  import org.any_openeai_enterprise.moa.objects.resources.v1_0.Greetee;
52  import org.any_openeai_enterprise.moa.jmsobjects.coreapplication.v1_0.Greeting;
53
54  /**
55   * This class implements the message support of the EnterpriseGreetingService.
56   * <P>
57   * Specifically, this command handles
```

```
 58   * org.any-openeai-enterprise.CoreApplication.Greeting.Generate-Request message
 59   * and replies with an
 60   * org.any-openeai-enterprise.CoreApplication.Greeting.Response-Reply.
 61   * <P>
 62   * <B>1. org.any-openeai-enterprise.CoreApplication.Greeting.Generate-Request</
 63   * <P>
 64   * Gets the value of the Greetee/FullName and concatenates, 'Hello', the
 65   * value of the FullName, and '!' to generate the Greeting.
 66   * <P>
 67   * <B>Configuration Parameters:</B>
 68   * <P>
 69   * This command expects exactly one properties object in the command
 70   * configuration named 'GreetingRequestCommandProperties' with the following
 71   * properties. The properties object may have any name, because it is retrieved
 72   * by type.
 73   * <P>
 74   * <TABLE BORDER=2 CELLPADDING=5 CELLSPACING=2>
 75   * <TR>
 76   * <TH>Name</TH>
 77   * <TH>Required</TH>
 78   * <TH>Description</TH>
 79   * </TR>
 80   * <TR HALIGN="left" VALIGN="top">
 81   * <TD>responseDocumentUri</TD>
 82   * <TD>yes</TD>
 83   * <TD>URI for retrieving the primed
 84   * org.any-openeai-enterprise.CoreMessaging.Greeting.Response-Reply document</T
 85   * </TR>
 86   * <TR HALIGN="left" VALIGN="top">
 87   * <TD>defaultGreetingText</TD>
 88   * <TD>yes</TD>
 89   * <TD>Default text to use for a greeting, of no other input is provided.</TD>
 90   * </TR>
 91   * </TABLE>
 92   * <P>
 93   * <B>Error Messages:</B>
 94   * <P>
 95   * <TABLE BORDER=2 CELLPADDING=5 CELLSPACING=2>
 96   * <TR>
 97   * <TH>Code</TH>
 98   * <TH>Type</TH>
 99   * <TH>Description</TH>
100   * <TH>Explanation</TH>
101   * </TR>
102   * <TR HALIGN="left" VALIGN="top">
103   * <TD><NOBR>OpenEAI-1001</NOBR></TD>
104   * <TD>application</TD>
105   * <TD>Unsupported message object: [unsupported message object name]. This
106   * 'Greeting')</TD>
107   * <TD>Somehow the wrong message object name is getting placed into the
108   * message by the sending application or it is sending the wrong message
109   * entirely.</TD>
110   * </TR>
111   * <TR HALIGN="left" VALIGN="top">
112   * <TD><NOBR>OpenEAI-1002</NOBR></TD>
113   * <TD>application</TD>
114   * <TD>Unsupported message action: [unsupported message action name]. This
115   * command only supports '[supported message action name(s)]'. (in this case
116   * 'generate')</TD>
117   * <TD>Somehow the wrong message action name is getting placed into the
```

```
118   * message by the sending application or it is sending the wrong message
119   * entirely. This command only supports Generate-Request messages.</TD>
120   * </TR>
121   * <TR HALIGN="left" VALIGN="top">
122   * <TD><NOBR>OpenEAI-1015</NOBR></TD>
123   * <TD>application</TD>
124   * <TD>Invalid generate element found in the Generate-Request message. This
125   * command expects a Greetee.</TD>
126   * <TD>Somehow the wrong generate object is getting placed into the
127   * message by the sending application. A Greetee object is expected.</TD>
128   * </TR>
129   * <TR HALIGN="left" VALIGN="top">
130   * <TD><NOBR>EnterpriseGreetingService-1001</NOBR></TD>
131   * <TD>application</TD>
132   * <TD>An error occurred building the Greetee object from the Greetee element
133   * in the Generate-Request. The exception is: [EnterpriseLayoutException
134   * message].</TD>
135   * <TD>There was an error building the Greetee XmlEnterpriseObject from the
136   * Greetee element in the message. Verify that the Greetee object sent in
137   * has allowable values.</TD>
138   * </TR>
139   * <TR HALIGN="left" VALIGN="top">
140   * <TD><NOBR>EnterpriseGreetingService-1002</NOBR></TD>
141   * <TD>application</TD>
142   * <TD>An error occurred setting the value of the greeting text. The exception
143   * is: [EnterpriseFieldException message].</TD>
144   * <TD>There was an error setting the value of the greeting text. Verify that
145   * the command is generating a value allowed by the enterprise objects document
146   * used for greeting.</TD>
147   * </TR>
148   * <TR HALIGN="left" VALIGN="top">
149   * <TD><NOBR>EnterpriseGreetingService-1003</NOBR></TD>
150   * <TD>application</TD>
151   * <TD>An error occurred retrieving a pub/sub producer to use to publish the
152   * Greeting.Create-Sync message. The exception is: [JMSException message].</TD>
153   * <TD>Verify that the ProducerPool is correctly configured and that the
154   * JMS Provider is operating and the producers are able to connect to it.</TD>
155   * </TR>
156   * <TR HALIGN="left" VALIGN="top">
157   * <TD><NOBR>EnterpriseGreetingService-1004</NOBR></TD>
158   * <TD>application</TD>
159   * <TD>An error occurred building the greeting element from the greeting object
160   * The exception is: [EnterpriseLayoutException message].</TD>
161   * <TD>Verify that the command is generating a value allowed by the enterprise
162   * objects document used for greeting.</TD>
163   * </TR>
164   * <TR HALIGN="left" VALIGN="top">
165   * <TD><NOBR>EnterpriseGreetingService-1005</NOBR></TD>
166   * <TD>application</TD>
167   * <TD>An error occurred publishing the Greeting.Create-Sync message to
168   * communicate this create action to the rest of the enterprise. The
169   * Generate-Request cannot be processed. The exception is:
170   * [EnterpriseLayoutException message].</TD>
171   * <TD>Verify that the command is generating a value allowed by the enterprise
172   * objects document used for greeting.</TD>
173   * </TR>
174   * </TABLE>
175   * <P>
176   * @author      Steve Wheat (steve@openeai.org)
177   * @version     1.0 - 24 September 2003
```

```
178   */
179  public class GreetingRequestCommand
180    extends RequestCommandImpl implements RequestCommand {
181
182    private Document m_responseDoc = null;  // the primed XML response document
183    private ProducerPool m_pubSubProducerPool = null; // for publishing syncs
184    private String m_defaultGreetingText = null; // default greeting text
185
186    /**
187     * @param CommandConfig
188     * @throws InstantiationException
189     * <P>
190     * This constructor initializes the command using a CommandConfig object. It
191     * invokes the constructor of the ancestor, RequestCommandImpl, and then
192     * retrieves one PropertyConfig object from AppConfig by name and gets and
193     * sets the command properties using that PropertyConfig object. This means
194     * that this command must have one PropertyConfig object in its configuration
195     * named 'GreetingRequestCommandProperties'. This constructor also initialize
196     * the response document and provide document used in replies. It also gets a
197     * pool of PubSubProducers from AppConfig for the command to use in publishin
198     * sync messages. The command excpects a producer pool named
199     * 'EgsSubSubProducerPool'. Then the constuctor gets the value of the
200     * defaultGreetingText property and sets the value of the defaultGreetingText
201     * to use if no Greetee information is received in a request.
202     */
203    public GreetingRequestCommand(CommandConfig cConfig) throws
204      InstantiationException {
205      super(cConfig);
206
207      // Get and set the general properties for this command.
208      PropertyConfig pConfig = new PropertyConfig();
209      try {
210        pConfig = (PropertyConfig)getAppConfig()
211          .getObject("GreetingRequestCommandProperties");
212      }
213      catch (EnterpriseConfigurationObjectException eoce) {
214        String errMsg = "Error retrieving a PropertyConfig object from " +
215          "AppConfig: The exception is: " + eoce.getMessage();
216        logger.fatal("[GreetingRequestCommand] " + errMsg);
217        throw new InstantiationException(errMsg);
218      }
219      setProperties(pConfig.getProperties());
220
221      // Initialize response document.
222      XmlDocumentReader xmlReader = new XmlDocumentReader();
223      try {
224        logger.debug("[GreetingRequestCommand] " +
225          "responseDocumentUri: " + getProperties()
226          .getProperty("responseDocumentUri"));
227        m_responseDoc = xmlReader.initializeDocument(getProperties()
228          .getProperty("responseDocumentUri"), getOutboundXmlValidation());
229        if (m_responseDoc == null) {
230          String errMsg = "Missing 'responseDocumentUri' " +
231            "property in the deployment descriptor.  Can't continue.";
232          logger.fatal("[GreetingRequestCommand] " + errMsg);
233          throw new InstantiationException(errMsg);
234        }
235      }
236      catch (XmlDocumentReaderException e) {
237        logger.fatal("[GreetingRequestCommand] Error initializing" +
```

```
238           " the primed documents.");
239         e.printStackTrace();
240         throw new InstantiationException(e.getMessage());
241       }
242
243       // Initialize a pub/sub producer pool.
244       try {
245         m_pubSubProducerPool = (ProducerPool)getAppConfig().
246           getObject("EgsPubSubProducerPool");
247       }
248       catch (EnterpriseConfigurationObjectException eoce) {
249         String errMsg = "Error retrieving a ProducerPool object " +
250           "from AppConfig. The exception is: " + eoce.getMessage();
251         logger.fatal("[GreetingRequestCommand] " + errMsg);
252         throw new InstantiationException(errMsg);
253       }
254
255       // Get the defaultGreetingText property.
256       String defaultGreetingText = getProperties()
257         .getProperty("defaultGreetingText");
258       if (defaultGreetingText == null || defaultGreetingText.equals("")) {
259         throw new InstantiationException("Missing 'defaultGreetingText' " +
260           "property in the deployment descriptor. Can't continue.");
261       }
262       setDefaultGreetingText(defaultGreetingText);
263       logger.info("[GreetingRequestCommand] defaultGreetingText is '" +
264         getDefaultGreetingText() + "'.");
265
266       logger.info("[GreetingRequestCommand] instantiated successfully.");
267     }
268
269     /**
270      * @param int, the number of the message processed by the consumer.
271      * @param Message, the message for the command to process.
272      * @throws CommandException, with details of the error processing the message
273      * <P>
274      * This method makes a local copy of the response and provide documents to
275      * use in the reply to the request. Then it converts the JMS message to an XM
276      * document, retrieves the text portion of the message, clears the message
277      * body in preparation for the reply, gets the ControlArea from the XML
278      * document, and verifies that message object of the message is a
279      * Greeting. If the message object is not a Greeting, the command replies wit
280      * an error to the client. If the message object is a Greeting, it processes
281      * the message. The comments above contain a detailed description of the
282      * processing logic for supported message actions for a Greeting. That logic
283      * is implemented in this method.
284      */
285     public final Message execute(int messageNumber, Message aMessage)
286       throws CommandException {
287       // Make a local copy of the response document to use in the replies.
288       Document localResponseDoc = (Document)m_responseDoc.clone();
289
290       // Convert the JMS Message to an XML Document
291       Document inDoc = null;
292       try {
293         inDoc = initializeInput(messageNumber, aMessage);
294       }
295       catch (Exception e) {
296         String errMsg = "Exception occurred processing input message in " +
297           "org.openeai.jms.consumer.commands.Command.  Exception: " +
```

```
298           e.getMessage();
299         throw new CommandException(errMsg);
300       }
301
302       // Retrieve text portion of message.
303       TextMessage msg = (TextMessage)aMessage;
304       try {
305         // Clear the message body for the reply, so we do not
306         // have to do it later.
307         msg.clearBody();
308       }
309       catch (Exception e) {
310         String errMsg = "Error clearing the message body.";
311         throw new CommandException(errMsg + ". The exception is: " +
312           e.getMessage());
313       }
314
315       // Get the ControlArea from XML document.
316       Element eControlArea = getControlArea(inDoc.getRootElement());
317
318       // Get messageAction and messageObject attributes from the
319       // ControlArea element.
320       String msgAction = eControlArea.getAttribute("messageAction").getValue();
321       String msgObject = eControlArea.getAttribute("messageObject").getValue();
322
323       // Verify that the message object we are dealing with is a Greeting; if not
324       // reply with an error.
325       if (msgObject.equalsIgnoreCase("Greeting") == false) {
326         String errType = "application";
327         String errCode = "OpenEAI-1001";
328         String errDesc = "Unsupported message object: " + msgObject +
329           ". This command expects 'Greeting'.";
330         logger.fatal("[GreetingRequestCommand] " + errDesc);
331         logger.fatal("[GreetingRequestCommand] Message sent in is: \n" +
332           getMessageBody(inDoc));
333         ArrayList errors = new ArrayList();
334         errors.add(buildError(errType, errCode, errDesc));
335         String replyContents =
336           buildReplyDocumentWithErrors(eControlArea, localResponseDoc, errors);
337         return getMessage(msg, replyContents);
338       }
339
340       // Get a configured Greeting and Greetee from AppConfig.
341       Greeting greeting = new Greeting();
342       try {
343       greeting = (Greeting)getAppConfig().getObjectByType(greeting.getClass()
344         .getName());
345       }
346       catch (EnterpriseConfigurationObjectException eoce) {
347         logger.fatal("[GreetingRequestCommand] Error retrieving a Greeting " +
348           "object from AppConfig: The exception is: " + eoce.getMessage());
349       }
350       Greetee greetee = new Greetee();
351       try {
352         greetee = (Greetee)getAppConfig().getObjectByType(greetee.getClass()
353           .getName());
354       }
355       catch (EnterpriseConfigurationObjectException eoce) {
356         logger.fatal("[GreetingRequestCommand] Error retrieving a Greeting " +
357           "object from AppConfig: The exception is: " + eoce.getMessage());
```

```
358       }
359
360       // Handle a Generate-Request.
361       if (msgAction.equalsIgnoreCase("Generate")) {
362         logger.info("[GreetingRequestCommand] Handling an " +
363           "org.any-openeai-enterprise.CoreApplication.Greeting.Generate-Request"
364           " message.");
365         Element eGreetee = inDoc.getRootElement().getChild("DataArea")
366           .getChild("Greetee");
367
368         // Verify that Greetee element is not null; if it is, reply with an error
369         if (eGreetee == null) {
370           String errType = "application";
371           String errCode = "OpenEAI-1015";
372           String errDesc = "Invalid generate element found in the Generate-" +
373             "Request message. This command expects a Greetee.";
374           logger.fatal("[GreetingRequestCommand] " + errDesc);
375           logger.fatal("Message sent in is: \n" + getMessageBody(inDoc));
376           ArrayList errors = new ArrayList();
377           errors.add(buildError(errType, errCode, errDesc));
378           String replyContents =
379             buildReplyDocumentWithErrors(eControlArea, localResponseDoc, errors);
380           return getMessage(msg, replyContents);
381         }
382
383         // Now build a Greetee object from the Greetee element in the message.
384         try {
385           greetee.buildObjectFromInput(eGreetee);
386         }
387         catch (EnterpriseLayoutException ele) {
388           // There was an error building the Greetee object from a Greetee
389           // element.
390           String errType = "application";
391           String errCode = "EnterpriseGreetingService-1001";
392           String errDesc = "An error occurred building Greetee object from the "
393             "Greetee element in the Generate-Request message. The exception " +
394             "is: " + ele.getMessage();
395           logger.fatal("[GreeteeRequestCommand] " + errDesc);
396           logger.fatal("Message sent in is: \n" + getMessageBody(inDoc));
397           ArrayList errors = new ArrayList();
398           errors.add(buildError(errType, errCode, errDesc));
399           String replyContents =
400           buildReplyDocumentWithErrors(eControlArea, localResponseDoc, errors);
401           return getMessage(msg, replyContents);
402         }
403
404         // Generate the greeting.
405         try {
406           if (greetee.getFullName() != null && greetee.getFullName() != "") {
407             // There is a greetee with a name to greet. Prepare the greeting text
408             greeting.setText("Hello, " + greetee.getFullName() + "!");
409           }
410           else {
411             // There is no greetee with a name to greet. Use the default greeting
412             // text.
413             greeting.setText(getDefaultGreetingText());
414           }
415         }
416         catch (EnterpriseFieldException efe) {
417           // An error occurred setting the value of the greeting text. Log it and
```

```
418           // reply with an error.
419           String errType = "application";
420           String errCode = "EnterpriseGreetingService-1002";
421           String errDesc = "An error occurred setting the value of the " +
422             "greeting text. The exception is: " + efe.getMessage();
423           logger.fatal("[GreetingRequestCommand " + errDesc);
424           ArrayList errors = new ArrayList();
425           errors.add(buildError(errType, errCode, errDesc));
426           String replyContents =
427           buildReplyDocumentWithErrors(eControlArea, localResponseDoc, errors);
428           return getMessage(msg, replyContents);
429         }
430
431         // Set the TestId.
432         greeting.setTestId(greetee.getTestId());
433
434         // Get pub/sub producer to use in this transaction.
435         PubSubProducer pub = null;
436         try {
437           pub = (PubSubProducer)m_pubSubProducerPool.getProducer();
438         }
439         catch (JMSException jmse) {
440           // An error occurred retrieving a pub/sub producer to use to publish
441           // the Greeting.Create-Sync. Log it and reply with an error.
442           String errType = "application";
443           String errCode = "EnterpriseGreetingService-1003";
444           String errDesc = "An error occurred retrieving a pub/sub producer to "
445             "use to publish the Greeting.Create-Sync. The exception is: " +
446             jmse.getMessage();
447           logger.fatal("[GreetingRequestCommand " + errDesc);
448           ArrayList errors = new ArrayList();
449           errors.add(buildError(errType, errCode, errDesc));
450           String replyContents =
451           buildReplyDocumentWithErrors(eControlArea, localResponseDoc, errors);
452           return getMessage(msg, replyContents);
453         }
454
455         // Serialize the greeting and place it into the reply.
456         String replyContents = null;
457         try {
458           localResponseDoc.getRootElement().getChild("DataArea").removeContent();
459           localResponseDoc.getRootElement().getChild("DataArea").
460             addContent((Element)greeting.buildOutputFromObject());
461           replyContents = buildReplyDocument(eControlArea, localResponseDoc);
462         }
463         catch (EnterpriseLayoutException ele) {
464           // There was an error building the Greeting element from the Greeting
465           // object.
466           String errType = "application";
467           String errCode = "EnterpriseGreetingService-1004";
468           String errDesc = "Error building Greeting element from the Greeting " +
469             "object.  The exception is: " + ele.getMessage();
470           logger.fatal("[GreetingRequestCommand " + errDesc);
471           ArrayList errors = new ArrayList();
472           errors.add(buildError(errType, errCode, errDesc));
473           replyContents = buildReplyDocumentWithErrors(eControlArea,
474             localResponseDoc, errors);
475           return getMessage(msg, replyContents);
476         }
477
```

```
478         // Publish the create sync message.
479         try { greeting.createSync(pub); }
480         catch (EnterpriseObjectSyncException eose) {
481           // An error occurred publishing the Greeting.Create-Sync message.
482           // Log it and reply with an error.
483           String errType = "application";
484           String errCode = "EnterpriseGreetingService-1005";
485           String errDesc = "An error occurred publishing the Greeting.Create-" +
486             "Sync message to communicate the create action to the rest of the " +
487             "enterprise. The Generate-Request cannot be processed. " +
488             "The exception is: " + eose.getMessage();
489           logger.fatal("[GreetingRequestCommand " + errDesc);
490           ArrayList errors = new ArrayList();
491           errors.add(buildError(errType, errCode, errDesc));
492           replyContents = buildReplyDocumentWithErrors(eControlArea,
493             localResponseDoc, errors);
494           return getMessage(msg, replyContents);
495         }
496
497         // Return the response with status success.
498         return getMessage(msg, replyContents);
499       }
500       else {
501         // The messageAction is invalid; it is not a query.
502         String errType = "application";
503         String errCode = "OpenEAI-1002";
504         String errDesc = "Unsupported message action: " + msgAction + ". " +
505           "This command only supports 'generate'.";
506         logger.fatal("[GreetingRequestCommand] " + errDesc);
507         logger.fatal("Message sent in is: \n" + getMessageBody(inDoc));
508         ArrayList errors = new ArrayList();
509         errors.add(buildError(errType, errCode, errDesc));
510         String replyContents =
511         buildReplyDocumentWithErrors(eControlArea, localResponseDoc, errors);
512         return getMessage(msg, replyContents);
513       }
514     }
515
516     /**
517      * @param String, the default greeting text.
518      * <P>
519      * Sets the default greeting text.
520      */
521     private void setDefaultGreetingText(String defaultGreetingText) {
522       m_defaultGreetingText = defaultGreetingText;
523     }
524
525     /**
526      * @return String, the default greeting text.
527      * <P>
528      * Returns the default greeting text.
529      */
530     private String getDefaultGreetingText() {
531       return m_defaultGreetingText;
532     }
533   }
534
535
```