Added by Mark Horton, last edited by Mark Horton on Jul 17, 2008

Integrating General Interface With Tempo

General Interface is a Javascript based IDE that allows you to easily build dynamic web apps. We integrated GI into Intalio's Designer which created a new way to create task forms. The task forms needed to interact with Tempo so we had to see how GI would integrate with Tempo. This integration had 3 main areas:

1. Implementing the server components.
2. Interfacing with the Tempo WSDL operations.
3. A special case for handling attachments.

Server Components

Tempo already has support for alternate form types. So the server components we needed to implement were pretty straight forward. GI already has a framework for communicating with WSDLs and building web apps, so we didn't have a lot of work to do on the server side.

The 2 main items implemented were **ComponentManager** and **DeployServlet**. **ComponentManager** handles deployment and knows what's been deployed (available to users). **DeployServlet** bootstraps the **ComponentManager** (via web.xml <load-on-startup>1</load-on-startup>) and acts as glue between the different pieces.

The **ComponentManager** class should implement org.intalio.tempo.deployment.spi.ComponentManager. For General Interface I didn't need to do anything special. The main thing I needed was to be able to store and retrieve the GI application path. For this the **_giapps** Map object was used.

ComponentManager.java

```
/**
 * Copyright (c) 2008 Intalio inc.
 *
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 * Intalio inc. - initial API and implementation
 */
package com.intalio.bpms.gi;

import static com.intalio.bpms.gi.LocalizedMessages._;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.intalio.tempo.deployment.ComponentId;
import org.intalio.tempo.deployment.DeploymentMessage;
import org.intalio.tempo.deployment.spi.ComponentManagerResult;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * @author <a href="http://www.intalio.com">Intalio Inc.</a>
 */
public class ComponentManager implements org.intalio.tempo.deployment.spi.ComponentManager {

    protected static final Logger LOG = LoggerFactory.getLogger(ComponentManager.class);

    private Map<ComponentId, ApplicationInfo> _giapps = Collections
        .synchronizedMap(new HashMap<ComponentId, ApplicationInfo>());

    public void activate(ComponentId id, File path) {
        LOG.debug(_("Activate: {0} Path: {1}", id, path));

        ApplicationInfo info = new ApplicationInfo();

        try {
            info.appPath = path.getCanonicalPath();
        } catch (IOException ex) {
            LOG.error(_("Error setting appPath.", ex));
        }

        _giapps.put(id, info);
    }

    public void deactivate(ComponentId id) {
        LOG.debug(_("Deactivate: {0}", id));
        _giapps.remove(id);
    }
    public ComponentManagerResult deploy(ComponentId id, File path) {
        LOG.debug(_("Deploy: {0} Path: {1}", id, path));
        return new ComponentManagerResult(new ArrayList<DeploymentMessage>());
    }

    public String getComponentManagerName() {
        return "gi";
    }

    public void start(ComponentId id) {
        LOG.debug(_("Start: {0}", id));
    }

    public void stop(ComponentId id) {
        LOG.debug(_("Stop: {0}", id));
    }

    public void undeploy(ComponentId id, List<String> resources) {
        LOG.debug(_("Undeploy: {0}", id));
    }

    public ApplicationInfo getGIAppInfo(ComponentId id) {
        return _giapps.get(id);
    }
}
```

The same goes for **DeployServlet**. I didn't need anything special so just used the most basic implementation.

DeployServlet.java

```
/**
 * Copyright (c) 2008 Intalio inc.
 *
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 * Intalio inc. - initial API and implementation
 */
package com.intalio.bpms.gi;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;

import org.intalio.tempo.deployment.utils.DeploymentServiceRegister;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * @author <a href="http://www.intalio.com">Intalio Inc.</a>
 */
public class DeployServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected static final Logger LOG = LoggerFactory.getLogger(DeployServlet.class);

    protected final String CONFIG_DIR = System.getProperty("com.intalio.bpms.configDirectory");

    public static final String GI_PROPERTY_FILE = "file:${org.intalio.tempo.configDirectory}/gi.properties";

    protected String propertyFile = GI_PROPERTY_FILE;

    protected ComponentManager _manager;

    protected Configuration _configuration;

    protected DeploymentServiceRegister _register;

    protected static DeployServlet INSTANCE;

    public DeployServlet() {
        super();

        synchronized (DeployServlet.class) {
            if (INSTANCE != null) {
                LOG.warn("GI.DeployServlet.INSTANCE already initialized");
            }

            INSTANCE = this;
        }
    }

    public static DeployServlet getInstance() {
        synchronized (DeployServlet.class) {
            return INSTANCE;
        }
    }

    @Override
    public void init() throws ServletException {
        loadConfiguration();

        _manager = new ComponentManager();
        _register = new DeploymentServiceRegister(_manager);
        _register.init();
    }
    @Override
    public void destroy() {
        super.destroy();

        if (_register != null) {
            _register.destroy();
        }
    }

    void loadConfiguration() {
        _configuration = new Configuration();
    }

    public ComponentManager getComponentManager() {
        return _manager;
    }

    public Configuration getConfiguration() {
        return _configuration;
    }
}
```

Interfacing with Tempo's WSDL Operations

General Interface has some nice features that allow you to easily map data to and from WSDL operations via a web browser. We decided to use this ability to help us create and manage our tasks with Tempo. By communicating directly with Tempo from the web browser we remove the need for server side code to handle this. Depending on what you are doing this may or may not be beneficial. There is probably a performance benefit when commuincating directly.

An alternative would be to create servlets and have the browser submit data to the servlet, and then the servlet invoke the WSDL operations, and then return the results to the browser.

The main operations we needed to use involve creating a task, completing a task, changing task state, adding attachments, etc... The three Tempo WSDLs we used and the operations we invoked are briefly described below.

1) Task Management Service WSDL

- Get Task - `getTask()` - This is invoked when the task is first loaded in the web page. Task related data, if any, will be returned and can be used to populate fields in the web page. Task state and other task metadata will also be returned.
- Initialize Process - `initProcess()` - This is invoked to create a new task process.
- Save - `setOutput()` - After a process is started, this can be invoked to save the task data. Invocations of Get task will then return the saved data.
- Dismiss task - `complete()` - Dismiss the task.
- Get attachments - `getAttachments()` - Gets a list of attachments and their metadata associated with a particular task.
- Add Attachment - `addAttachment()` - Adds the given attachment metadata to the task. Add must also be invoked on Task Attachment Service to add the actual file.
- Remove Attachment - `removeAttachment()` - Remove the given attachment metadata from the task. Delete must also be invoked on Task Attachment Service to remove the actual file.

2) Task Manager Process WSDL

- Claim Task - `claimTask()` - This sets the task state to CLAIMED.
- Revoke Task - `revokeTask()` - This will un-claim a task by changing its state to READY.
- Complete Task - `completeTask()` - Completes a task and sets its task state to COMPLETED.

3) Task Attachment Service WSDL

- Add - `add()` - Adds the the actual file to the file system. Add Attachment should also be invoked on Task Management Service to add the associated metadata.
- Delete - `delete()` - This deletes the actual file from the file system. Remove Attachment should also be invoked on Task Management Service to remove the associated metadata.

Handling Attachment Files

Attachment files represent a special case for General Interface because GI is designed to be used within a web browser. Local files are off limits to Javascript due to security restrictions. So there is no easy way to submit the file directly to a WSDL operation using GI.

To work around this we created a regular HTML form and submitted the file to a servlet. The servlet then accepted the file, base64 encoded it, and submitted it to the Task Attachment Service WSDL via the add() operation. It then also invokes the addAttachment() operation on Task Management Service to add the file metatadata.

Here's the servlet we created:

AttachmentServlet.java

```
/**
 * Copyright (c) 2008 Intalio inc.
 *
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 * Intalio inc. - initial API and implementation
 */
```

```
package com.intalio.bpms.gi;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.namespace.QName;

import org.apache.axiom.om.OMAbstractFactory;
import org.apache.axiom.om.OMElement;
import org.apache.axiom.om.OMFactory;
import org.apache.axiom.om.OMNamespace;
import org.apache.axis2.AxisFault;
import org.apache.axis2.addressing.EndpointReference;
import org.apache.axis2.client.Options;
import org.apache.axis2.client.ServiceClient;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * @author <a href="http://www.intalio.com">Intalio Inc.</a>
 */
public class AttachmentServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected static final Logger LOG = LoggerFactory.getLogger(RouterServlet.class);

    public AttachmentServlet() {
        super();
    }

    @SuppressWarnings("unchecked")
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        String shp = req.getScheme() + "://" + req.getServerName() + ":" + req.getServerPort();

        PrintWriter out = resp.getWriter();

        if (ServletFileUpload.isMultipartContent(req)) {
            HashMap<String, FileItem> fieldMap = new HashMap<String, FileItem>();

            try {
                FileItemFactory factory = new DiskFileItemFactory();
                ServletFileUpload upload = new ServletFileUpload(factory);
                upload.setSizeMax(1100000); // 1MB limit (plus a little extra)

                // get the fields and convert to a hashmap
                List items = upload.parseRequest(req);
                for (Object object : items) {
                    FileItem item = (FileItem) object;
                    fieldMap.put(item.getFieldName(), item);
                }

                if (fieldMap.size() > 0) {
                    OMElement tasResponse = callTasAddAttachment(shp, fieldMap);
                    if (tasResponse == null) {
                        LOG.info("null tasResponse");
                    } else {
                        out.println(tasResponse);
                        OMElement tmsResponse = callTmsAddAttachment(shp, fieldMap, tasResponse);
                        out.println(tmsResponse);
                    }
                }
            } catch (FileUploadException ex) {
                LOG.error(ex.getMessage());
            }
        }

        out.println("ok");
        out.close();
    }

    protected OMElement callTasAddAttachment(String shp, HashMap<String, FileItem> fieldMap)
            throws AxisFault {
        FileItem fileItem = fieldMap.get("attachmentFile");

        String participantToken = getFieldString(fieldMap, "participantToken");
        String users = getFieldString(fieldMap, "authorizedUsers");
        String roles = getFieldString(fieldMap, "authorizedRoles");
        String type = getFieldString(fieldMap, "attachmentType");
        String text = getFieldString(fieldMap, "attachmentText");
        String title = getFieldString(fieldMap, "attachmentName");

        String mimeType = "text/plain";
        String fileName = title;

        boolean isText = ("text".equalsIgnoreCase(type));
        if (!isText) {
            if (fileItem == null) {
                LOG.info("null attachmentFile FileItem");
                return null;
            }

            mimeType = fileItem.getContentType();
            fileName = fileItem.getName();
        }

        if (title.equals("")) {
            LOG.info("empty title");
            return null;
        }

        if (fileName.equals("")) {
            LOG.info("empty fileName");
            return null;
        }

        OMFactory fac = OMAbstractFactory.getOMFactory();
        OMNamespace ns = fac.createOMNamespace(
            "http://www.intalio.com/BPMS/Workflow/TaskAttachmentService/", "tns");

        OMElement request = fac.createOMElement("addRequest", ns);

        OMElement authCredentials = fac.createOMElement("authCredentials", ns);
        addTextElement(fac, ns, authCredentials, "participantToken", participantToken);

        if (!users.equals("")) {
            OMElement authroizedUsers = fac.createOMElement("authorizedUsers", ns);
            addTextElement(fac, ns, authroizedUsers, "user", users);
            authCredentials.addChild(authroizedUsers);
        }

        if (!roles.equals("")) {
            OMElement authroizedRoles = fac.createOMElement("authorizedRoles", ns);
            addTextElement(fac, ns, authroizedRoles, "role", roles);
            authCredentials.addChild(authroizedRoles);
        }

        request.addChild(authCredentials);

        OMElement attachmentMetadata = fac.createOMElement("attachmentMetadata", ns);
        addTextElement(fac, ns, attachmentMetadata, "mimeType", mimeType);
        addTextElement(fac, ns, attachmentMetadata, "filename", fileName);
        request.addChild(attachmentMetadata);

        if (isText) {
            addTextElement(fac, ns, request, "plaintext", text);
        } else {
            String payload = new String(Base64.encodeBase64(fileItem.get()));
            addTextElement(fac, ns, request, "payload", payload);
        }

        // invoke the service
        ServiceClient sc = new ServiceClient();
        Options opts = new Options();
        opts.setTo(new EndpointReference(shp + "/axis2/services/tas"));
        opts.setAction("add");

        sc.setOptions(opts);

        return sc.sendReceive(request);
    }

    protected OMElement callTmsAddAttachment(String shp, HashMap<String, FileItem> fieldMap,
            OMElement tasResponse) throws AxisFault {
        FileItem fileItem = fieldMap.get("attachmentFile");

        String taskId = getFieldString(fieldMap, "taskId");
        String participantToken = getFieldString(fieldMap, "participantToken");
        String type = getFieldString(fieldMap, "attachmentType");
        String title = getFieldString(fieldMap, "attachmentName");

        String mimeType = "text/plain";
        String fileName = title;

        boolean isText = ("text".equalsIgnoreCase(type));

        if (!isText) {
            if (fileItem == null) {
                LOG.info("null attachmentFile FileItem");
                return null;
            }

            mimeType = fileItem.getContentType();
            fileName = fileItem.getName();
        }

        // get the payloadUrl from the previous operation
        QName qname = new QName("http://www.intalio.com/BPMS/Workflow/TaskAttachmentService/",
            "url", "tas");
        OMElement element = tasResponse.getFirstChildWithName(qname);
        if (element == null) {
            LOG.info("null payloadUrl");
            return null;
        }

        String payloadUrl = element.getText();
        String description = "";
        String description = "";
        String creationDate = "";

        OMFactory fac = OMAbstractFactory.getOMFactory();
        OMNamespace ns = fac.createOMNamespace(
            "http://www.intalio.com/BPMS/Workflow/TaskManagementServices-20051109/", "tns");

        OMElement request = fac.createOMElement("addAttachmentRequest", ns);
        addTextElement(fac, ns, request, "taskId", taskId);

        OMElement attachment = fac.createOMElement("attachment", ns);
        OMElement attachmentMetadata = fac.createOMElement("attachmentMetadata", ns);
        addTextElement(fac, ns, attachmentMetadata, "mimeType", mimeType);
        addTextElement(fac, ns, attachmentMetadata, "fileName", fileName);
        addTextElement(fac, ns, attachmentMetadata, "title", title);
        addTextElement(fac, ns, attachmentMetadata, "description", description);
        addTextElement(fac, ns, attachmentMetadata, "creationDate", creationDate);
        attachment.addChild(attachmentMetadata);
        addTextElement(fac, ns, attachment, "payloadUrl", payloadUrl);
        request.addChild(attachment);

        addTextElement(fac, ns, request, "participantToken", participantToken);

        // invoke the service
        ServiceClient sc = new ServiceClient();
        Options opts = new Options();
        opts.setTo(new EndpointReference(shp + "/axis2/services/TaskManagementServices"));
        opts.setAction("addAttachment");
        sc.setOptions(opts);

        return sc.sendReceive(request);
    }

    protected void addTextElement(OMFactory fac, OMNamespace ns, OMElement parent, String name,
            String text) {
        OMElement child = fac.createOMElement(name, ns);
        child.setText(text);
        parent.addChild(child);
    }

    protected String getFieldString(HashMap<String, FileItem> fieldMap, String field) {
        FileItem item = fieldMap.get(field);
        if (item == null) return "";
        return item.getString();
    }
}
```

Comments