```
/*
 * Created on 11.01.2006
 * $Id: //products/main/pkg/JRENDER/qatest_src/ixos/qa/render/Template.java#1 $
 *
 */
package ixos.qa.render;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

import junit.framework.TestCase;

import ixos.base.log4j.Log;
import ixos.base.utils.StreamUtils;
import ixos.base.utils.TestUtils;
import ixos.render.service.EmailDescriptor;
import ixos.render.service.RenderContentSource;
import ixos.render.service.RenderContentTarget;
import ixos.render.service.RenderException;
import ixos.render.service.RenderJob;
import ixos.render.service.RenderOptions;
import ixos.render.service.RenderService;
import ixos.render.service.RenderServiceFactory;
import ixos.render.service.RenderState;
import ixos.render.service.Utils;

/**
 * <!-- purpose -->
 *
 * <p>
 * <b>
 * <br>$Id: //products/main/pkg/JRENDER/qatest_src/ixos/qa/render/Template.java#1 $
 * </b>
 *
 * @author initial author: andip 11.01.2006
 *
 * @version
 * version: $Revision: #1 $
 * <br>last change: $DateTime: 2006/01/13 13:43:56 $
 * <br>last author: $Author: andip $
 * <br>Perforce changelist: $Change: 380877 $
 *
 *
 * @since <!-- logical version introducing this class -->
 *
 */
public class Template extends TestCase
{
    private final transient static Log.Context LC = new Log.Context();
```

```
protected String logDir = System.getProperty("java.io.tmpdir", ".");

public Template(String name)
{
    super(name);

    this.logDir = TestUtils.initLogging("qatest_src/configstore.properties",
        Template.class.getName(), name);
}

public void dummy()
{
    final String MN = "";
    LC.important(MN, "ENTER...");
    LC.important(MN, "...LEAVE");
}

//---------------------------------
// test methods
//---------------------------------

// for it to be JUnit-Testcase this method must be named testCompleteCycle()
public void doTestCompleteCycle()
{
    final String MN = "testCompleteCycle";
    LC.important(MN, "ENTER...");

    //-------------------------
    // Obtain a RenderService
    RenderService rs = obtainRenderService();

    //-------------------------
    // Create a RenderJob
    RenderJob job = createRenderJob(rs);

    //------------------------------
    // Add the render sources
    InputStream inputStream = new ByteArrayInputStream(MN.getBytes());
    RenderContentSource rcs = addStreamSource(job, inputStream, MN, "text/plain");
    try
    {
        rcs.setUrlForm("http://inside.ixos.de/uweke.jpg");
    }
    catch (RenderException e1)
    {
        String msg = "Cannot set the form URL";
        LC.error(MN, msg, e1);
        throw new RuntimeException(msg, e1);
    }

    File bushSaddam = new File("\\\\mucfs02\\eng_el\\Projects\\Callisto\\Documents\\formats\\bushSaddam.jpg");
    addStreamSource(job, bushSaddam, "bushSaddam.jpg", "image/jpeg");
```

```
//addFileSource(job, "\\\\mucfs02\\eng_el\\Projects\\Callisto\\Documents\\formats\\harryNazi.jpg", "image/jpeg");

addUrlSource(job, "http://inside.ixos.de/uweke.jpg", "image/jpeg");
//addAlUrlSource(job, url, contentType);



//--------------------------------
// Set the render target
RenderContentTarget rct = job.setTargetContent();
rct.setStream();


//------------------------------
// set the target content type
job.setTargetContentType("application/pdf");


//--------------------------------
// set the render options
RenderOptions options = job.createRenderOptions();
options.setProperty(RenderOptions.OPT_PAGE_SIZE, "A5");
options.setProperty(RenderOptions.OPT_PAGE_SIZE_SCALE_CONTENT,
    String.valueOf(true));
//options.setProperty(RenderOptions.OPT_SS_PROFILE, "profileName");


//--------------------------------
// add the job
try
{
    rs.addJob(job);
}
catch (RenderException e)
{
    String msg = "Cannot add the job '"+job+"'";
    LC.error(MN, msg, e);
    throw new RuntimeException(msg, e);
}

//--------------------------------
// start the job
try
{
    rs.startJob(job);
}
catch (RenderException e)
{
    String msg = "Cannot start the job";
    LC.error(MN, msg, e);
    throw new RuntimeException(msg, e);
}
```

```java
//---------------------------------
// wait for the job to be finished
RenderState jobState = null;
try
{
    jobState = pollJobState(rs, job);
}
catch (RenderException e)
{
    String msg = "Cannot poll the job state";
    LC.error(MN, msg, e);
    throw new RuntimeException(msg, e);
}

//-------------------------------
// get the protocol
InputStream protocolStream = null;
try
{
    protocolStream = rs.getProtocol(job);
}
catch (RenderException e)
{
    String msg = "Cannot load the protocol.";
    LC.error(MN, msg, e);
    throw new RuntimeException(msg, e);
}
String protocolStr = null;
try
{
    protocolStr = Utils.convertInputStreamToString(protocolStream);
}
catch (Exception e)
{
    String msg = "Cannot convert the protocol into a String.";
    LC.error(MN, msg, e);
    throw new RuntimeException(msg, e);
}

// log the protocol
LC.important(MN, "Protocol: \r\n"+protocolStr);




if (jobState.equals(RenderState.ERROR))
{
    //--------- job finished with error-----------------
    // log the protocol and show error
    String msg = "RenderJob '"+job
        +"' finished with error. Protocoll: \r\n:"+protocolStr;
    LC.important(MN, msg);
    throw new RuntimeException(msg);
    }
    else if (jobState.equals(RenderState.FINISHED))
    {
        //----------- job finished without error ---------------
        // get the rendition
        RenderState state = null;
        try
        {
            state = rs.getRendition(job);
        }
        catch (RenderException e)
        {
            String msg = "Cannot get the rendition of job '"+job+"'";
            LC.error(MN, msg, e);
            throw new RuntimeException(msg, e);
        }
        if (! state.equals(jobState))
        {
            LC.warn(MN, "GetRendition returned another state: "+state
                +", jobState was: "+jobState);
        }
        InputStream renditionStream = rct.getInputStream();

        // write the rendition into a file for review
        File renditionFile = new File(this.logDir, MN+".pdf");
        try
        {
            FileOutputStream out = new FileOutputStream(renditionFile);
            StreamUtils.copyBytes(renditionStream, out, -1, 4096);
        }
        catch (IOException e)
        {
            String msg = "Cannot copy the rendition into the file "
                +renditionFile.getAbsolutePath();
            LC.error(MN, msg, e);
            throw new RuntimeException(msg, e);
        }

        String msg = "Check the file "+renditionFile.getAbsolutePath();
        LC.important(MN, msg);
        System.err.println(msg);
    }
    else
    {
        //--------- error wrong state ------------------
        String msg = "Unexpected job state '"+jobState+"'.";
        LC.error(MN, msg, null);
        throw new RuntimeException(msg);
    }

    LC.important(MN, "...LEAVE");
}
```

```
//------------------------------
// protected methods
//-----------------------------
protected RenderService obtainRenderService()
{
    final String MN = "obtainRenderService";
    LC.enter(MN);
    RenderServiceFactory rsf = RenderServiceFactory.getInstance();
    RenderService rs = null;
    try
    {
        rs = rsf.getRenderService(Constants.protocol, Constants.host,
            Constants.port, Constants.path);
    }
    catch (RenderException e)
    {
        String msg = "Cannot get RenderService";
        LC.error(MN, msg, e);
        throw new RuntimeException(msg, e);
    }
    LC.leave(MN, rs);
    return rs;
}
protected RenderJob createRenderJob(RenderService rs)
{
    final String MN = "createRenderJob";
    LC.enter(MN);

    RenderJob job = null;
    try
    {
        job = rs.createRenderJob();
    }
    catch (RenderException e)
    {
        String msg = "Cannot create a renderJob.";
        LC.error(MN, msg, e);
        throw new RuntimeException(msg, e);
    }

    LC.leave(MN, job);
    return job;
}
protected EmailDescriptor obtainEmailDescriptor(String subject, String msg)
{
    final String MN = "obtainEmailDescriptor";
    LC.enter(MN);

    EmailDescriptor ed = new EmailDescriptor(Constants.emailTo);
    ed.setFrom(Constants.emailFrom);
    ed.setSubject(subject);
    ed.setMessage(msg);
    ed.setCopySelf(Boolean.FALSE);
    LC.leave(MN, ed);
    return ed;
}
protected RenderContentSource addStreamSource(RenderJob job, InputStream stream,
    String name, String contentType)
{
    final String MN = "addStreamSource";
    if (LC.enterOn())
    {
        LC.enter(MN, "job='"+job+"', stream='"+stream+"', contentType='"
            +contentType+"'");
    }
    RenderContentSource rcs = job.addSourceContent();
    try
    {
        rcs.setStream(stream, null, name, contentType);
    }
    catch (RenderException e)
    {
        String msg = "Cannot set the InputStream '"+stream
            +"' in content source '"
            +rcs+"'.";
        LC.error(MN, msg, e);
        throw new RuntimeException(msg, e);
    }
    return rcs;
}
protected RenderContentSource addStreamSource(RenderJob job, File file,
    String name, String contentType)
{
    final String MN = "addStreamSource";
    if (LC.enterOn())
    {
        LC.enter(MN, "job='"+job+"', file='"+file+"', contentType='"
            +contentType+"'");
    }
    FileInputStream fis = null;
    try
    {
        fis = new FileInputStream(file);
    }
    catch (FileNotFoundException e1)
    {
        String msg = "Cannot find the file "+file.getAbsolutePath();
        LC.error(MN, msg, e1);
        throw new RuntimeException(msg, e1);
    }
    RenderContentSource rcs = job.addSourceContent();
    try
    {
        rcs.setStream(fis, null, name, contentType);
    }
    catch (RenderException e)
    {
        String msg = "Cannot set the file '"+file.getAbsolutePath()
            +"' as stream source in content source '"
            +rcs+"'.";
        LC.error(MN, msg, e);
        throw new RuntimeException(msg, e);
    }
    return rcs;
}
protected RenderContentSource addFileSource(RenderJob job, String filename,
    String contentType)
{
    final String MN = "addFileSource";
    if (LC.enterOn())
    {
        LC.enter(MN, "job='"+job+"', filename='"+filename+"', contentType='"
            +contentType+"'");
    }
    RenderContentSource rcs = job.addSourceContent();
    try
    {
        rcs.setFile(filename, contentType);
    }
    catch (RenderException e)
    {
        String msg = "Cannot set the file '"+filename
            +"' in content source '"
            +rcs+"'.";
        LC.error(MN, msg, e);
        throw new RuntimeException(msg, e);
    }
    return rcs;
}
protected RenderContentSource addUrlSource(RenderJob job, String url,
    String contentType)
{
    final String MN = "addUrlSource";
    if (LC.enterOn())
    {
        LC.enter(MN, "job='"+job+"', url='"+url+"', contentType='"
            +contentType+"'");
    }
    RenderContentSource rcs = job.addSourceContent();
    try
    {
        rcs.setUrl(url, contentType);
    }
    catch (RenderException e)
    {
        String msg = "Cannot set the url '"+url
            +"' in content source '"
            +rcs+"'.";
        LC.error(MN, msg, e);
        throw new RuntimeException(msg, e);
    }
    return rcs;
}
protected RenderContentSource addAlUrlSource(RenderJob job, String url,
    String contentType)
{
    final String MN = "addAlUrlSource";
    if (LC.enterOn())
    {
        LC.enter(MN, "job='"+job+"', url='"+url+"', contentType='"
            +contentType+"'");
    }
    RenderContentSource rcs = job.addSourceContent();
    try
    {
        rcs.setArchivedDoc(url, contentType);
    }
    catch (RenderException e)
    {
        String msg = "Cannot set the ArchiveLink URL '"+url
            +"' in content source '"
            +rcs+"'.";
        LC.error(MN, msg, e);
        throw new RuntimeException(msg, e);
    }
    return rcs;
}

protected RenderState pollJobState(RenderService rs, RenderJob job)
throws RenderException
{
    final String MN = "pollJobState";
    if (LC.enterOn())
    {
        LC.enter(MN, "rs='"+rs+"', job='"+job+"'");
    }

    RenderState state = null;
    while (true)
    {
        try
        {
            LC.debug(MN, "Sleep a second...");
            Thread.sleep(1000);
            LC.debug(MN, "... woke up.");
        }
        catch (InterruptedException e)
        {
            LC.warn(MN, "Sleeping interrupted.");
            Thread.currentThread().interrupt();
        }
        state = rs.getState(job);

        if ((state.equals(RenderState.ERROR))
            || (state.equals(RenderState.FINISHED)))
        {
            break;
        }
    }

    if (LC.leaveOn())
    {
        LC.leave(MN, state);
    }
    return state;
}
}
```