Displaying content (e.g. PDF, Excel, Word) in an IFRAME

Displaying content (e.g. PDF, excel, word) in an IFRAME

From time to time people ask this question, or similar or related ones, on the list. So, I'll try to summarize the answer I gave on the list...

Suppose we want to show a PDF (an Excel/ a Word) document embedded in a Wicket page. If you look into Wicket sources you will find the class **org.apache.wicket.markup.html.link.InlineFrame**. This class is suitable for displaying a Wicket page embedded inside an IFRAME contained in another Wicket page. So, let's clone and modify this class to make it suitable for displaying other types of contents (e.g. a generated PDF).

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package some.example;

import org.apache.wicket.IResourceListener;
import org.apache.wicket.markup.ComponentTag;
import org.apache.wicket.markup.html.WebMarkupContainer;
import org.apache.wicket.util.string.Strings;

/**
 * Implementation of an <a href="http://www.w3.org/TR/REC-html40/present/frames.html#h-16.5">inline
 * frame</a> component. Must be used with an iframe (&lt;iframe src...) element. The src attribute
 * will be generated. Its is suitable for displaying <em>generated contend</em> like PDF, EXCEL, WORD,
 * etc.
 *
 * @author Ernesto Reinaldo Barreiro
 *
 */

public class DocumentInlineFrame extends WebMarkupContainer implements IResourceListener
{
        private static final long serialVersionUID = 1L;


        private IResourceListener resourceListener;

        /**
         * Constructor receiving an IResourceListener..
         *
         * @param id
         * @param resourceListener
         */
        public DocumentInlineFrame(final String id, IResourceListener resourceListener)
        {
                super(id);
                this.resourceListener = resourceListener;
        }

        /**
         * Gets the url to use for this link.
         *
         * @return The URL that this link links to
         */
        protected CharSequence getURL()
```

```
        {
                return urlFor(IResourceListener.INTERFACE);
        }

        /**
         * Handles this frame's tag.
         *
         * @param tag
         *            the component tag
         * @see org.apache.wicket.Component#onComponentTag(ComponentTag)
         */
        @Override
        protected final void onComponentTag(final ComponentTag tag)
        {
                checkComponentTag(tag, "iframe");

                // Set href to link to this frame's frameRequested method
                CharSequence url = getURL();

                // generate the src attribute
                tag.put("src", Strings.replaceAll(url, "&", "&amp;"));

                super.onComponentTag(tag);
        }



        @Override
        protected boolean getStatelessHint()
        {
                return false;
        }

        public void onResourceRequested() {
                this.resourceListener.onResourceRequested();
        }
}
```

So, this class instead of implementing ILinkListener implements IResourceListener and generates an URL for the src attribute of IFRAME that points back to itself. The class receives an IResourceListener as an attribute on the constructor and delegates the production of the IFRAME contents to this IResourceListener. If you want to know which of the implementations of IResourceListener you need just open it with your favorite IDE and search for all classes implementing it (on Eclipse you can easily do that typing Ctr-T😉.

Now that we have our main class let's use it to display a PDF.

On the same package as class above I create the following class:

```
package some.example;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.wicket.markup.html.DynamicWebResource;

/**
 * @author Ernesto Reinaldo
 *
 */
public class MyPdfResource extends DynamicWebResource {

	private static final long serialVersionUID = 1L;

	static int BUFFER_SIZE = 10*1024;

	/**
	 *
	 */
	public MyPdfResource() {
	}

	/* (non-Javadoc)
	 * @see org.apache.wicket.markup.html.DynamicWebResource#getResourceState()
	 */
	@Override
	protected ResourceState getResourceState() {
		return new ResourceState() {

			@Override
			public String getContentType() {
				return "application/pdf";
			}

			@Override
			public byte[] getData() {
				try {
					return bytes(MyPdfResource.class.getResourceAsStream("test.pdf"));
				} catch (Exception e) {
					return null;
				}
			}
		};
	}

	public static  byte[] bytes(InputStream is) throws IOException {
		ByteArrayOutputStream out = new ByteArrayOutputStream();
		copy(is, out);
		return out.toByteArray();
	}

	public static void copy(InputStream is, OutputStream os) throws IOException {
		byte[] buf = new byte[BUFFER_SIZE];
		while (true) {
			int tam = is.read(buf);
			if (tam == -1) {
				return;
			}
			os.write(buf, 0, tam);
		}
	}
}
```

This class just reads a PDF named **test.pdf** from the same package **some.example**. So, to make the example work take you favorite PDF file and drop it on that folder and rename it to **test.pdf**.

Then lets show this PDF in a Wicket panel:

```
package some.example;
import org.apache.wicket.markup.html.panel.Panel;

public class MyPdfPanel extends Panel {

        private static final long serialVersionUID = 1L;

        public MyPdfPanel(String id) {
                super(id);

                setRenderBodyOnly(true);
                add(new DocumentInlineFrame("mypdf", new MyPdfResource()));
        }
}
```

and

```
<html>
    <wicket:panel>
        <iframe wicket:id="mypdf" width="90%" height="300px"></iframe>
    </wicket:panel>
</html>
```

If you now include previous panel in a Wicket page you will be able to see your test PDF embedded on that page!