EmittingRSSFeeds

Emitting RSS Feeds

The ROME library (http://rome.dev.java.net) is a nice library that encapsulates the consumption and generation of RSS feeds in many formats. It can be used to create Tapestry components that generate RSS. Note that this code is not thoroughly tested, and was written against Rome 0.7. I (and Zillow) make no guarantees about how well this works...

RssFeedWriter component

This component encapsulates the entire feed. It should be the outermost component of the template being used to generate the feed.

```
package com.zillow.web.components.rss;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import org.apache.tapestry.AbstractComponent;
import org.apache.tapestry.IMarkupWriter;
import org.apache.tapestry.IRequestCycle;
import org.apache.tapestry.annotations.ComponentClass;
import org.apache.tapestry.annotations.InjectObject;
import org.apache.tapestry.annotations.Parameter;
import org.apache.tapestry.engine.NullWriter;
import org.apache.tapestry.markup.MarkupWriterSource;

import com.sun.syndication.feed.synd.SyndEntry;
import com.sun.syndication.feed.synd.SyndFeed;
import com.sun.syndication.feed.synd.SyndFeedImpl;
import com.sun.syndication.io.SyndFeedOutput;
import com.zillow.web.infrastructure.Property;

@ComponentClass( allowBody = true )
public abstract class RssFeedWriter extends AbstractComponent
{
        @InjectObject( "infrastructure:markupWriterSource" )
        public abstract MarkupWriterSource getMarkupWriterSource();

        @Parameter( defaultValue = "literal:rss_2.0" )
        public abstract String getFeedFormat();

        @Parameter( required = true )
        public abstract String getTitle();

        @Parameter( required = true )
        public abstract String getLink();

        @Parameter( required = true )
        public abstract String getDescription();

        @Parameter
        public abstract Date getPublishDate();

        @Property
        public abstract SyndFeed getFeed();
        public abstract void setFeed( SyndFeed feed );

        public static final String RSS_FEED_ATTRIBUTE = "rssFeed";

        @Property
        public abstract List< SyndEntry > getEntryList();
        public abstract void setEntryList( List< SyndEntry > entryList );

        @Override
        protected void renderComponent(IMarkupWriter writer, IRequestCycle cycle)
        {
                cycle.setAttribute( RSS_FEED_ATTRIBUTE, this );
                try
```

```
                {
                        SyndFeed feed = new SyndFeedImpl();
                        feed.setFeedType( getFeedFormat() );
                        feed.setTitle( getTitle() );
                        feed.setLink( getLink() );
                        feed.setDescription( getDescription() );
                        feed.setPublishedDate( getPublishDate() );

                        setEntryList( new ArrayList< SyndEntry >() );

                        renderBody( new NullWriter(), cycle );

                        cycle.setAttribute( RSS_FEED_ATTRIBUTE, null );

                        feed.setEntries( getEntryList() );

                        SyndFeedOutput output = new SyndFeedOutput();
                        writer.printRaw( output.outputString( feed ) );
                }
                catch ( Exception ex )
                {
                        throw new RuntimeException( ex );
                }
        }

        public void addEntry( SyndEntry entry )
        {
                getEntryList().add( entry );
        }

}
```

RssFeedEntry component

This component encapsulates each entry of the feed. The body of this component will be sent as the content of the entry.

```
package com.zillow.web.components.rss;

import java.io.PrintWriter;
import java.io.StringWriter;
import java.util.Date;

import org.apache.tapestry.AbstractComponent;
import org.apache.tapestry.IMarkupWriter;
import org.apache.tapestry.IRequestCycle;
import org.apache.tapestry.annotations.InjectObject;
import org.apache.tapestry.annotations.Parameter;
import org.apache.tapestry.markup.MarkupWriterSource;
import org.apache.tapestry.util.ContentType;

import com.sun.syndication.feed.synd.SyndContent;
import com.sun.syndication.feed.synd.SyndContentImpl;
import com.sun.syndication.feed.synd.SyndEntry;
import com.sun.syndication.feed.synd.SyndEntryImpl;

public abstract class RssFeedEntry extends AbstractComponent
{
        @InjectObject( "infrastructure:markupWriterSource" )
        public abstract MarkupWriterSource getMarkupWriterSource();

        @Parameter( required = true )
        public abstract String getTitle();

        @Parameter( required = true )
        public abstract String getLink();

        @Parameter( defaultValue = "literal:text/html" )
        public abstract String getContentType();

        @Parameter
        public abstract Date getPublishDate();

        @Override
        protected void renderComponent(IMarkupWriter writer, IRequestCycle cycle)
        {
                RssFeedWriter feed = (RssFeedWriter)cycle.getAttribute( RssFeedWriter.RSS_FEED_ATTRIBUTE );
                if ( feed == null )
                        throw new RuntimeException( "RssFeedEntry must be nested inside RssFeedWriter" );

                SyndEntry entry = new SyndEntryImpl();
                entry.setTitle( getTitle() );
                entry.setLink( getLink() );
                entry.setPublishedDate( getPublishDate() );
                SyndContent contents = new SyndContentImpl();
                contents.setType( getContentType() );

                StringWriter buffer = new StringWriter();
                PrintWriter printWriter = new PrintWriter( buffer );
                IMarkupWriter bufWriter = getMarkupWriterSource().newMarkupWriter(
printWriter,
new ContentType( getContentType() ) );
                renderBody( bufWriter, cycle );

                contents.setValue( buffer.toString() );
                entry.setDescription( contents );
                feed.addEntry( entry );
        }
}
```

How to use

Your template should contain only an RssFeedWriter component and its body, which must consist of one ore more RssFeedEntry components. Anything inside the RssFeedWriter, but not inside an RssFeedEntry, will be rendered but discarded.

Rss.html:

```
<span jwcid="@rss/RssFeedWriter" title="My Neato Feed" link="ognl:feedLink"
                        description="The best RSS Feed ever!"  publishDate="ognl:today">
        <span jwcid="@rss/RssFeedEntry" title="The First Entry" link="ognl:firstEntryLink" publishDate="ognl:
firstEntryDate" >
              This is the first entry.  It can contain other <span jwcid="@Insert" value="literal:components"
>components</span>.
        </span>
        <span jwcid="@rss/RssFeedEntry" title="The Second Entry" link="ognl:secondEntryLink" publishDate="ognl:
secondEntryDate" >
              This is the second entry.
        </span>
</span>
```

The only requirement for the page class is that you change the content type. Otherwise, it's just another page...

Rss.java:

```
public abstract class Rss extends BasePage
{
    @Override
    public ContentType getResponseContentType()
    {
        return new ContentType("text/xml");
    }

    // Everything else...
}
```