Migrating content created with Blossom 0.5

Your Rating: ☆☆☆☆☆ Results: ★★★☆☆ 114 rates

Migrating content created with Blossom 0.5

This guide is for early adopters that have content created with version 0.5 and want to upgrade it. Read on to find out why this might be necessary and how to do it.

How the content model differs between Blossom 0.5 and 1.0+

As some of you know Blossom 0.5 used a placeholder paragraph instead of the ParagraphRenderer mechanism it uses today. In 0.5 Blossom had a single paragraph called 'blossom' that when rendered looked at a nodeData named 'handlerPath' and invoked your Spring controller. The nodeData was set to the path that your controller was mapped to.

In 1.0 this was changed and for each controller annotated with @Paragraph a proper paragraph is registered with Magnolia.

Why should you migrate your content?

The new mechanism has many benefits, for instance localization which is done per paragraph now work. Previously you could set the resource bundle name only on the placeholder paragraph and that would take effect on all your blossom paragraphs.

How the migration works

Migrating your content means changing the template of all paragraphs created with version 0.5 to the new correct template name.

It is highly recommended that you start Spring from your module using the recommended module startup method introduced in 1.1 (BlossomModuleSupport). If you're starting Spring from web.xml then it will be running when your install tasks run and you will need an extra restart after all this.

In the process of upgrading to Blossom 1.2.2 I created an install task that automatically migrates my content on install. All you have to do is add it to your modules VersionHandler like this:

```
register(DeltaBuilder.update("1.44", "").addTask(new MigrateBlossomPlaceholderParagraphs()));
```

This means that when my module updates to version 1.44 all content is scanned and migrated where necessary.

If you have customized the name of the placeholder paragraph and/or the name of the handlerPath-nodeData you can specify this using the second constructor.

My install task looks like this

It is suggested that you try this a few times on a local development repository before really just running it in production, even though we have tested this quite a few times.

```
package se.issi.stellata.module;

import info.magnolia.cms.beans.config.ContentRepository;
import info.magnolia.cms.core.Content;
import info.magnolia.cms.util.ContentUtil;
import info.magnolia.module.InstallContext;
import info.magnolia.module.delta.AllChildrenNodesOperation;
import info.magnolia.module.delta.TaskExecutionException;
import org.apache.commons.lang.StringUtils;

import javax.jcr.RepositoryException;

/**
 * @author Åke Argéus
 */
public class MigrateBlossomPlaceholderParagraphs extends AllChildrenNodesOperation {

        private final String placeholderParagraph;
        private final String handlerPath;

        public MigrateBlossomPlaceholderParagraphs() {
                super("Migrate blossom-0.5 paragraphs", "Migrate blossom-0.5 paragraphs", ContentRepository.
WEBSITE, "/", ContentUtil.EXCLUDE_META_DATA_CONTENT_FILTER);
                this.placeholderParagraph = "blossom";
                this.handlerPath = "handlerPath";
        }

        public MigrateBlossomPlaceholderParagraphs(String placeholderParagraph, String handlerPath) {
                super("Migrate blossom-0.5 paragraphs", "Migrate blossom-0.5 paragraphs", ContentRepository.
WEBSITE, "/", ContentUtil.EXCLUDE_META_DATA_CONTENT_FILTER);
                this.placeholderParagraph = placeholderParagraph;
                this.handlerPath = handlerPath;
        }

        protected void doExecute(final InstallContext ctx) throws RepositoryException, TaskExecutionException {
                log.info("Starting migration of blossom-0.5 paragraphs");
                final Content parentNode = getParentNode(ctx);

                try {
                        ContentUtil.visit(parentNode, new ContentUtil.Visitor() {
                                public void visit(Content subNode) throws Exception {
                                        operateOnChildNode(subNode, ctx);
                                }
                        }, ContentUtil.EXCLUDE_META_DATA_CONTENT_FILTER);
                } catch (Exception e) {
                        throw new TaskExecutionException("Could not visit subnode", e);
                }
                log.info("Finished migration of blossom-0.5 paragraphs");
        }

        @Override
        protected void operateOnChildNode(Content subNode, InstallContext ctx) throws RepositoryException,
TaskExecutionException {
                if (subNode.getMetaData().getTemplate().equals(placeholderParagraph)) {
                        log.info("Migrating away from placeholder paragraph on paragraph at:" + subNode.
getHandle());
                        String template = StringUtils.replaceChars(StringUtils.strip(subNode.getNodeData
(handlerPath).getString(), "/"), "/#", "_");
                        subNode.getMetaData().setTemplate(template);
                        subNode.deleteNodeData(handlerPath);
                        subNode.save();
                }
        }

}
```