StringInterpolator Document Rendering Library

Table of Contents

This module (`incode-module-docrendering-stringinterpolator`) provides an implementation of the Document subdomain module's `Renderer` interface using the stringinterpolator library module.

API

The module provides three different implementations of `Renderer`:

- `RendererForStringInterpolator` which implements `RendererFromCharsToChars`
  - useful for interpolating document names or simple text (eg an HTML email)
- `RendererForStringInterpolatorCaptureUrl` which implements `RendererFromCharsToBytes`
  - interpolates to a URL, then uses the `UrlDownloaderService` (SPI service, discussed below) to download the URL and return its content as a `byte[]`. For example, could be used to download a PDF or Word document.
- `RendererForStringInterpolatorPreviewAndCaptureUrl` which implements `RendererFromCharsToBytesWithPreviewToUrl`
  - this is very similar to the previous renderer, but also provides access to the intermediary URL, thereby allowing the Document module to provide an action to preview the document.

These classes can be used as the `Renderer` implementation for a Document `RenderingStrategy`. Subclasses of the `RenderingStrategyFSAbstract` fixture script can be used to create such an entity, eg:

- `RenderingStrategyFSForStringInterpolator`

```java
public class RenderingStrategyFSForStringInterpolator extends
RenderingStrategyFSAbstract {
    public static final String REF = "SI";
    protected void execute(ExecutionContext executionContext) {
        upsertRenderingStrategy(
                REF,
                "String interpolate",
                DocumentNature.CHARACTERS,
                DocumentNature.CHARACTERS,
                RendererForStringInterpolator.class, executionContext);
    }
}
```

- `RenderingStrategyFSForStringInterpolatorCaptureUrl`

```
public class RenderingStrategyFSForStringInterpolatorCaptureUrl extends
RenderingStrategyFSAbstract {
    public static final String REF = "SINC";
    protected void execute(ExecutionContext executionContext) {
        upsertRenderingStrategy(
                REF,
                "String interpolate URL for Capture (no preview)",
                DocumentNature.CHARACTERS,
                DocumentNature.BYTES,
                RendererForStringInterpolatorCaptureUrl.class, executionContext);
    }
}
```

- `RenderingStrategyFSForStringInterpolatorPreviewAndCaptureUrl`

```
public class RenderingStrategyFSForStringInterpolatorPreviewAndCaptureUrl extends
RenderingStrategyFSAbstract {
    public static final String REF = "SIPC";
    @Override
    protected void execute(ExecutionContext executionContext) {
        upsertRenderingStrategy(
                REF,
                "String interpolate URL for Preview and Capture",
                DocumentNature.CHARACTERS,
                DocumentNature.BYTES,
                RendererForStringInterpolatorPreviewAndCaptureUrl.class,
executionContext);
    }
}
```

The document subdomain module also allows `RenderingStrategy`s to be created from the UI; it will "discover" all `Renderer` implementations from the classpath.

SPI

The `RendererForStringInterpolatorCaptureUrl` and `RendererForStringInterpolatorPreviewAndCaptureUrl` renderer implementations both rely on the `UrlDownloaderService` to convert the interpolated URL string into an array of bytes. This interface is defined as:

```
public interface UrlDownloaderService {
    public byte[] download(URL url) throws IOException;
    boolean canDownload(URL url);
}
```

The application consuming this module can provide its own implementation of this service, or may be able to use one of the implementations provided out-of-the-box by the module discussed below).

SPI Implementations

The module provides two implementations of the `UrlDownloadService`. The renderer will use the first service where `canDownload(…)` returns `true`.

UrlDownloaderUsingNtlmCredentials

(As its name suggests), the `UrlDownloaderUsingNtlmCredentials` implementation of the service is able to access a URL and provide NTLM (Active Directory) credentials.

It requires the following configuration options:

- `incode.module.docrendering.stringinterpolator.UrlDownloaderUsingNtlmCredentials.user`
- `incode.module.docrendering.stringinterpolator.UrlDownloaderUsingNtlmCredentials.password`
- `incode.module.docrendering.stringinterpolator.UrlDownloaderUsingNtlmCredentials.host`

The user should be provided in the format `DOMAIN/username` eg `ACME/jbloggs`.

If the configuration options are not provided, then the service is effectively ignored.

This service has a priority of 100 (`@DomainService#menuOrder="100"`).

UrlDownloaderService.SimpleUsingGuava (fallback)

The `UrlDownloaderService.SimpleUsingGuava` implementation of the service simply uses Guava to download from the URL, providing no credentials. It is used as a fallback if no other service is able to download.

How to configure/use

Classpath

Update your classpath by adding this dependency in your dom project's `pom.xml`:

```
<dependency>
    <groupId>org.incode.example.docrendering</groupId>
    <artifactId>incode-example-docrendering-stringinterpolator-dom</artifactId>
    <version>1.16.1</version>
</dependency>
```

Check for later releases by searching Maven Central Repo.

For instructions on how to use the latest `-SNAPSHOT`, see the contributors guide.

Bootstrapping

In the `AppManifest`, update its `getModules()` method, eg:

```
@Override
public List<Class<?>> getModules() {
    return Arrays.asList(
            ...
            org.incode.example.docrendering.stringinterpolator.dom
.StringInterpolatorDocRenderingModule.class,
    );
}
```

Known issues

None known at this time.

Dependencies

Maven can report modules dependencies using:

```
mvn dependency:list -o -pl modules/dom/docrendering-stringinterpolator/impl -D
excludeTransitive=true
```

which, excluding Apache Isis itself, returns these compile/runtime dependencies:

```
org.apache.httpcomponents:httpclient:jar:4.5.2
```

From the Incode Platform it uses:

- base library module
- stringinterpolator library module
- document example subdomain module