Serviio Online Resource plugin implementation Guide

Doc version 1.6

Serviio version 1.4

Table of Contents

1 Preface

1.1 Scope

Serviio server includes a Groovy-based plugin system enabling to retrieve relevant information from Atom/RSS based feeds or generic web resources (e.g. web pages) that don't provide the required information in itself. This guide is to help developers write these plugins.

1.2 Intended Audience

This document should be used by third-party developers who wish to write Online Resources plugins for Serviio media server.

1.3 Revision history

Date	Version	Description
21/09/11	1.0	Initial version for Serviio 0.6
29/11/11	1.1	Added Web Resource plugin instructions
17/04/12	1.2	Updated Groovy to version 1.8.6, added decrypting shared methods, added UserAgent
22/10/12	1.3	Added getVersion() method
01/03/13	1.4	Added getExtractItemsTimeout() method
06/08/13	1.5	Added cacheKey on WebResourceItem
05/12/13	1.6	Updated description of cacheKey on WebResourceItem; added getOnlineFeedExpiryInterval() shared method

2 How do the plugins work

Serviio Online Resources plugins (from now on only “plugins”) include functionality that extracts information about content that Serviio requires for playback.

There are two types of online resources that may require plugin for playback:

- **Rss/Atom feed** - if the Rss/Atom feed XML doesn't include content URL (down to the media file level), but only URL to web resource that describes/advertises/plays the content
- **Web Resource** – a plugin is always needed for processing of generic Web Resource (e.g. web page including links to content files). Using this plugin you will be able to process almost any online resource.

The work needed to retrieve the URL will most likely be different from case to case and may involve making HTTP requests, parsing XML/HTML documents, etc.

In case of RTMP sources, this guide might help you work out the content URL for librtmp/RTMPDump: http://stream-recorder.com/forum/tutorial-simply-use-rtmpsrv-and-example-t6325.html?p=18142#post18142

2.1 RSS / Atom feeds

Below is an example of a RSS feed that doesn't require a plugin. Serviio can find the content URL in the `media:content` or `enclosure` or other elements.

```
<item>
      <title>DPP_0281</title>
      <pubDate>Tue, 26 Apr 2011 14:33:16 -0700</pubDate>
      <media:content url="http://farm6.static.flickr.com/5104/5658712239_b4c541f6ea_o.jpg"
type="image/jpeg" height="3456" width="5184" />
      media:title>DPP_0281</media:title>
      <media:thumbnail url="http://farm6.static.flickr.com/5104/5658712239_c66a553652_s.jpg"
height="75" width="75" />
      <enclosure url="http://farm6.static.flickr.com/5104/5658712239_b4c541f6ea_o.jpg"
type="image/jpeg" />
</item>
```

Other feeds might only point to a web page where the content can be played. Below is an example of such a feed item (Atom syntax).

```
<entry>
      <title type="text">Cherry's Cash Dilemmas</title>
      <category term="Factual" />
      <category term="Money" />
      <category term="TV" />
      <link rel="alternate"
href="http://www.bbc.co.uk/iplayer/episode/b013h51z/Cherrys_Cash_Dilemmas/" type="text/html"
title="Cherry's Cash Dilemmas">
            <media:content>
                  <media:thumbnail
url="http://node2.bbcimg.co.uk/iplayer/images/episode/b013h51z_150_84.jpg" width="150"
height="84" />
            </media:content>
      </link>
      <link rel="self" href="http://feeds.bbc.co.uk/iplayer/episode/b013h51z"
type="application/atom+xml" title="Cherry's Cash Dilemmas" />
      <link rel="related" href="http://www.bbc.co.uk/programmes/b013h51z" type="text/html"
title="Cherry's Cash Dilemmas" />
</entry>
```

There are no URLs pointing to the actual content files (with the exception of thumbnail) so a plugin will be necessary to retrieve that information.

3 Implementing plugins

The plugins are short implementations written in Groovy (http://groovy.codehaus.org). Groovy is a JVM language and runs in the same Java runtime environment as Serviio itself. Serviio bundles Groovy 1.8.3 so all the expected functionality is at your disposal. Plugins also need classes from serviio.jar, so make sure it's on your classpath if you want to test your plugins outside Serviio.

Once copied to Serviio's plugin folder, Serviio will attempt to compile the .groovy plugin file and, if successful, registers the plugin with the system.

3.1 RSS / Atom feed plugins

A Feed plugin has to extend class `org.serviio.library.online.FeedItemUrlExtractor`. This is an abstract class and it provides some shared functionality (see 3.3).

Plugin itself can live in any package, but a default (empty) package is recommended.

3.1.1 Methods to be implemented

These are the methods that have to be implemened for each plugin.

```
boolean extractorMatches(URL feedUrl)
```

Input parameters:

`feedUrl` – URL of the whole feed, as entered by the user

Description:

Called once for the whole feed, it returns `true` if the feed's items can be processed by this plugin. For each feed which needs a plugin Serviio tries to match all available plugins to the feed's URL by calling this method and uses the first plugin that returns `true`. Use of regular expression is recommended.

```
String getExtractorName()
```

Description:

Returns the name of this plugin. Is mostly used for logging and reporting purposes.

```
int getVersion()
```

Description:

Returns the version of this plugin. Defaults to “1” if the method is not implemented.

```
ContentURLContainer extractUrl(Map<String,URL> links, PreferredQuality
requestedQuality)
```

Input parameters:

links – a map of links as found in the feed item XML document. Keys are values of `rel` attribute of `link` elements. There are also special key names:

- default – for a `link` element without `rel` attribute
- thumbnail – includes URL of the item's thumbnail, if available

requestedQuality – includes value (HIGH, MEDUIM, LOW) of enumeration `org.serviio.library.online.PreferredQuality`. It should be taken into consideration if the online service offers multiple quality-based renditions of the content.

Description:

Performs the actual extraction of content information using the provided information. It returns an instance of `org.serviio.library.online.ContentURLContainer`. These are the properties of the class:

- `String` **contentUrl** – URL of the feed item's content; mandatory
- `String` **thumbnailUrl** – URL of the feed item's thumbnail; optional
- `org.serviio.library.metadata.MediaFileType` **fileType** – file type of the feed item; default is VIDEO
- `Date` **expiresOn** – a date the feed item expires on. It can mean the item itself expires or the item's contentUrl expires; the whole feed will be parsed again on the earliest expiry date of all feed items; optional
- `boolean` **expiresImmediately** – if true Serviio will extract the URL again when the play request is received to get URL that is valid for the whole playback duration. Note this is related to the content URL only, the feed item itself should still be valid and available; optional
- `String` **cacheKey** – a unique identifier of the content (i.e. this item with this quality) used as a key to technical metadata cache; required if either `expiresOn` and/or `expiresImmediately` is provided
- `boolean` **live** – identifies the content as a live stream; optional (default is `false`)
- `String` **userAgent** – specifies a particular User-Agent HPPT header to use when retrieving the content

These can be set either via a setter method (e.g. `setContentUrl()`) or via named constructor, e.g. `new ContentURLContainer(contentUrl: someContentUrl, thumbnailUrl: someThumbnailUrl)`.

If the object cannot be constructed the method should return `null` or throw an exception.

3.2 Web Resource plugins

A Web Resouce plugin has to extend class

`org.serviio.library.online.WebResourceUrlExtractor`. This is an abstract class and it provides some shared functionality (see 3.3).

Plugin itself can live in any package, but a default (empty) package is recommended.

3.2.1 Methods to be implemented

These are the methods that have to be implemened for each plugin.

```
boolean extractorMatches(URL resourceUrl)
```

Input parameters:

`resourceUrl` – URL of the web resource, as entered by the user

Description:

Called once for the whole web resource, it returns `true` if the resource can be processed by this plugin. For each web resource Serviio tries to match all available plugins to the resource's URL by calling this method and uses the first plugin that returns `true`. Use of regular expression is recommended.

```
String getExtractorName()
```

Description:

Returns the name of this plugin. Is mostly used for logging and reporting purposes.

```
int getVersion()
```

Description:

Returns the version of this plugin. Defaults to "1" if the method is not implemented.

```
int getExtractItemsTimeout()
```

Description:

Returns the number of seconds after which the plugin's `extractItems()` thread will be terminated (or marked as terminated). Defaults to 30 if the method is not implemented.

```
WebResourceContainer extractItems(URL resourceUrl, int maxItemsToRetrieve)
```

Input parameters:

`resourceUrl` – URL of the resource to be extracted. The plugin will have to get the contents on the URL itself.

`maxItemsToRetrieve` – Max. number of items the user prefers to get back or `-1` for unlimited. It is up to the plugin designer to decide how to limit the results (if at all).

Description:

Performs the extraction of basic information about the resource. It returns an instance of `org.serviio.library.online.WebResourceContainer`. These are the properties of the class:

- `String` **`title`** – title of the web resource; optional
- `String` **`thumbnailUrl`** – URL of the resource's thumbnail; optional
- `List<org.serviio.library.online.WebResourceItem>` **`items`** – list of extracted content items

`WebResourceItem` represents basic information about an item and has these properties:

- `String` **`title`** – title of the content item; mandatory
- `Date` **`releaseDate`** – release date of the content; optional
- `Map<String,String>` **`additionalInfo`** – a map of key – value pairs that can include information needed to retrieve content URL of the item (see `extractUrl()` method)
- `String` **`cacheKey`** – if present, the URL extracted on Serviio startup will be cached, and reused rather than re-extracted on subsequent feed refreshes, unless the item has explicitly expired due to the related `ContentURLContainer`'s `expiresOn` value or the feed is forcibly refreshed by the user

If the object cannot be constructed the method should return `null` or throw an exception.

```
ContentURLContainer extractUrl(WebResourceItem item, PreferredQuality
requestedQuality)
```

Input parameters:

`item` – an instance of `org.serviio.library.online.WebResourceItem`, as created in `extractItems()` method.

`requestedQuality` – includes value (HIGH, MEDUIM, LOW) of enumeration `org.serviio.library.online.PreferredQuality`. It should be taken into consideration if the online service offers multiple quality-based renditions of the content.

Description:

This method is called once for each item included in the created `WebResourceContainer`. Performs the actual extraction of content information using the provided item information. Use information from the provided `item`. It returns an instance of `org.serviio.library.online.ContentURLContainer`, whose properties are the same as in

3.1.1.

3.3 Available shared methods

These are the methods that plugins can use from the parent class:

```
void log(String message)
```

Input parameters:

message – Message to be logged

Description:

Writes the message to Serviio's log with DEBUG level.

```
String openURL(URL url, String userAgent)
```

Input parameters:

url – URL to retrieve

userAgent – User-Agent HTTP header to use

Description:

Opens the given URL using the provided User-Agent HTTP header. Returns content of the URL as a String.

```
String openURL(URL url, String userAgent, Map<String,String> cookies)
```

Input parameters:

url – URL to retrieve

userAgent – User-Agent HTTP header to use

cookies – Map of key – value pairs representing request cookies

Description:

Opens the given URL using the provided User-Agent HTTP header. Returns content of the URL as a String.

```
String generateMAC(String text, String hash, String algorithm)
```

Input parameters:

text – Text to be encrypted

hash – Hash to be used for encrypting

algorithm – Name of the MAC algorithm to use for encrypting, e.g. HmacMD5

Description:

Encrypts the provided text using a hash and Message Authentication Code (MAC) algorithm. List of standard algorithm names is at http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html#AppA

```
String decryptAES(String hexText, String key, String iv)
```

Input parameters:

hexText – Text to be decrypted, in the form of hexadecimal string, e.g. aabbcc

key – Key to use for decrypting

iv – Initialization vector

Description:

Decrypts the given hex string using AES Rijndael method.

```
byte[] decrypt(String text, String key, String method)
```

Input parameters:

text – Text (composed from bytes) to be decrypted

key – Key (composed from bytes) to use for decrypting

method – Decryption method to use, e.g. DES

Description:

Decrypts the given string using the key and method and returns the result as a array of bytes.

```
String decryptAsHex(String text, String key, String method)
```

Input parameters:

text – Text (composed from bytes) to be decrypted

key – Key (composed from bytes) to use for decrypting

method – Decryption method to use, e.g. DES

Description:

Decrypts the given string using the key and method and returns the result as a hexadecimal string (aabbcc).

```
String getFFmpegUserAgent()
```

Description:

Returns User-Agent HTTP header value the FFmpeg transcoder uses when accessing online content. It can be useful if the online service requires the same User-Agent to be used for extracting URLs and the actual content.

```
Integer getOnlineFeedExpiryInterval()
```

Description:

Returns user defined online feed expiry interval in hours.