Integrating Web Sequence Diagrams with other services

Don Brown from Atlassian provided a plugin for Confluence.

Carl Gould of Inductive Automation sent in a plugin for Trac. See the README file, and download here.

Here is a Macro for XWiki.

See below for command line interfaces in Python, Ruby, and Java.

Web pages

You can include diagrams in any web page, and they automatically update when the page changes. Just copy these two lines, and put your diagram source text between them.

```
<div class=wsd wsd_style="default"><pre>

</pre></div><script type="text/javascript" src="http://www.websequencediagrams.com/service.js"></script>
```

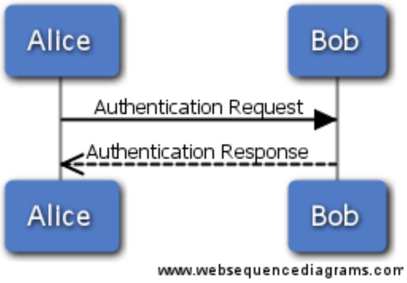
Do not remove the <pre> tags. They are required to preserve line-breaks on some browsers.

Example

```
<div class=wsd wsd_style="modern-blue" ><pre>

Alice->Bob: Authentication Request
Bob-->Alice: Authentication Response

</pre></div><script type="text/javascript" src="http://www.websequencediagrams.com/service.js"></script>
```



You can set wsd_style to any of these styles:

- default
- earth
- modern-blue
- mscgen
- omegapple
- qsd
- rose
- roundgreen
- napkin

Limitations

Due to limitations in some browsers, the diagram description must be under 2 KB after compression. If the diagram is too large, then no image will appear, but the text will be displayed.

Integrating with other services

You can access the websequencediagrams server programmatically using a HTTP POST request. Send the POST request to http://www.websequencediagrams.com/index.php. The request must contain "style" parameter, containing the name of the style to use. The "message" parameter contains the text of the diagram. In response to the POST request, websequencediagrams.com will return a string similar to the following:

```
{img: "?img=mscKTO107", page: 0, numPages: 1, errors: []}
```

The response is a JSON encoding of the following fields:

- img: When appended to "http://www.websequencediagrams.com/", the result is the URL of the image.
- page: always 0
- numPages: always 1
- errors: an array of errors in the source text, if there were any.

The image may be deleted from the server immediately after it is accessed. To access the image again, you will have to regenerate it using a separate POST request.

Python API

Here is an example implementation in python.

```
import urllib
import re

def getSequenceDiagram( text, outputFile, style = 'default' ):
    request = {}
    request["message"] = text
    request["style"] = style

    url = urllib.urlencode(request)

    f = urllib.urlopen("http://www.websequencediagrams.com/", url)
    line = f.readline()
    f.close()

    expr = re.compile("(\?img=[a-zA-Z0-9]+)")
    m = expr.search(line)

    if m == None:
        print "Invalid response from server."
        return False
```

Java API

Here is an example implementation in Java.

```
import java.io.*;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;

public class Main {

    public void getSequenceDiagram( String text, String outFile,
            String style )
    {

        try {
            //Build parameter string
            String data = "style=" + style + "&message=" +
                URLEncoder.encode( text, "UTF-8" );

            // Send the request
            URL url = new URL("http://www.websequencediagrams.com");
            URLConnection conn = url.openConnection();
```

Ruby API

Here is an example implementation in Ruby. Source: http://github.com/EmmanuelOga/websequencediagrams/tree/master

```
#! /usr/bin/env ruby
require 'net/http'
require 'uri'
require 'open-uri'

if ARGV[0] && $stdout.tty?
  if File.exist?(ARGV[0])
    text = File.read(ARGV[0])
  else
    text = open(ARGV[0], 'Accept' => 'text/plain').read
  end
else
  text = readlines.join
end

response = Net::HTTP.post_form(URI.parse('http://www.websequencediagrams.com/ind

if response.body =~ /img: "(.+)"/

  url = "http://www.websequencediagrams.com/#{ $1 }"
```