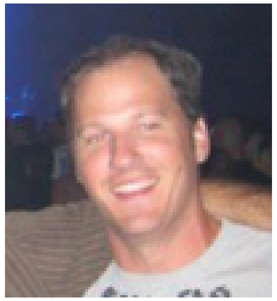


James Falkner

Blog

Documents

Recent Bloggers

Olaf Kock

43 Posts

December 1, 2012

Ronald Sarayudej

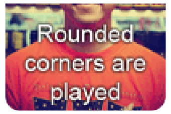
Yet Another Liferay JSON Service Example

« Back

June 10, 2011 By James Falkner

The past couple of years I've seen many examples of using Liferay's built-in JSON services in various ways. The architecture and syntax of this feature has undergone several refinements in the past few versions, so the documentation/examples you can find via a website search are usually slightly wrong and misleading. Recently I saw this thread come alive and decided to sit down and make a non-trival read/write example of using Liferay's JSON services work.

Bear in mind that this applies to Liferay 6.0.x (I am using 6.0.6 CE in the examples).



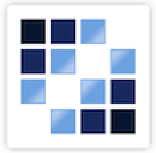
145 Posts

November 29, 2012

Kenneth Dong

1 Posts

November 27, 2012



Support Portal Admin

46 Posts

November 26, 2012

Jonas Yuan

53 Posts

November 24, 2012

Jorge Ferrer

46 Posts

November 22, 2012

Martin Yan

1 Posts

November 21, 2012

In Liferay 6.1 there are new interfaces coming like (such as RESTful interfaces), ability to do automatic serialization and improved method argument passing, and there are also existing "heavy lifting" web service interfaces like SOAP endpoints that one can use. So this is not the only way to do things. But it is great for prototyping and getting things to work quickly without dragging a bunch of dependencies and debugging hard-to-understand wire protocols. I hope this example is still relevant!

The only dependency I am using here is Apache Commons HTTPClient. I also decided to write it in Java (as opposed to Ray's earlier example on 5.2 in PHP).

Couple of things to be aware of:

- By default, access is through Liferay's tunnel-web web app. So the proper full URL in a default Liferay 6.0.6 install is `http://localhost:8080/tunnel-web/secure/json`.
- Since it is a "secure" (authenticated) interface we need to provide a username and password. This is done using HTTP Basic Authentication, which of course is not appropriate for a production environment, since the password is unencrypted (it is instead base64-encoded following HTTP Basic Authentication). The default username/password is "test/test".
- There's no error checking whatsoever here. You should add it for a real world scenario.

First Example

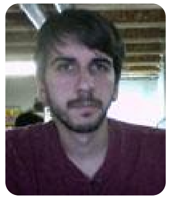
So, here's the first example. A simple "Hello World" that does the same thing as Ray's example, only using Liferay 6 and written in simple Java. It simply access the "Country" service and returns a list of country entities known to Liferay.

```
import org.apache.http.HttpHost;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.AuthCache;
```

[Joshua Asbury](#)

19 Posts

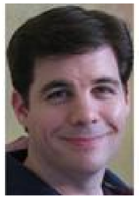
November 21, 2012



[Stephen Kostas](#)

1 Posts

November 13, 2012



[Neil Griffin](#)

30 Posts

October 31, 2012

```
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.protocol.ClientContext;
import org.apache.http.impl.auth.BasicScheme;
import org.apache.http.impl.client.BasicAuthCache;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.BasicHttpContext;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
```

If you compile and run this (you'll have to download the Apache Commons HTTPClient libraries and put them on the classpath and/or add them as dependencies in your IDE), then you should see something like this on your output screen (this is the returned content from the HTTP POST):

```
[{"countryId":20,"idd":"093","name":"Afghanistan","active":true,
"a2":"AF","number":"4","a3":"AFG"},{"countryId":21,"idd":"355","
name":"Albania","active":true,"a2":"AL","number":"8","a3":"ALB"}
,

.....many more countries listed

{"countryId":227,"idd":"263","name":"Zimbabwe","active":true,"a2
":"ZW","number":"716","a3":"ZWE"}]
```

Notice where I specify the username/password using Apache HTTPClient APIs. This should be easily translatable to your favorite client (or if you are using [curl](#) or some other RESTful client test console such as [rest-client](#) and you want to specify the authentication header manually, use an Authorization header with a value of `Basic dGVzdDp0ZXN0Cg==`)

Also note the serviceClassName parameter. This name (`com.lifery.portal.service.CountryServiceUtil`) specifies the service name (and maps to an actual class). Not all services have remote service endpoints (for example, there's no `com.liferay.portlet.social.service.SocialActivityServiceUtil` for creating new activity stream items. I wish there were).

The parameters are encoded into the body of the HTTP POST request using the Apache Commons HTTPClient's `UrlEncodedFormEntity` utility. This is the same as Ray's `$a->addPostData` examples in PHP.

Second Example

Ok, now that the easy one works (it does work for you, right?), let's move on to something trickier using the Web Content system. Notice that this system used to be called "Journal" so all the APIs refer to the Journal Service since the actual APIs were not changed in the interest of compatibility. This second example calls the JournalArticle service to retrieve a sample article using the default install's groupId of 10156 and the articleId of 10455. These numbers are automatically generated during initial startup the first time and may be different for you. If they are you'll need to change them. You can find them through the Control Panel by going to Communities -> Actions -> Edit to find the groupId, and pick any articleId from Web Content.

This example calls a specific method by identifying it using its formal parameters and passes the values for the parameters.

```
    public static void journal() throws Exception {
        HttpHost targetHost = new HttpHost("localhost", 8080, "h
ttp");
        DefaultHttpClient httpclient = new DefaultHttpClient();
        httpclient.getCredentialsProvider().setCredentials(
                new AuthScope(targetHost.getHostName(), targetHo
st.getPort()),
                new UsernamePasswordCredentials("test", "test"))
;

        // Create AuthCache instance
        AuthCache authCache = new BasicAuthCache();
        // Generate BASIC scheme object and add it to the local
        // auth cache
        BasicScheme basicAuth = new BasicScheme();
        authCache.put(targetHost, basicAuth);

        // Add AuthCache to the execution context
        BasicHttpContext ctx = new BasicHttpContext();
```

Notice here that the "setup" is exactly the same as before, only the parameters are different. Also note that the list of parameter names (`serviceParameters`) starts and ends with brackets. It's an array of Strings! So don't forget the brackets.

The getArticle method returns a JSON-encoded article from Liferay's web content system, so if this example works for you, you should get this on your output stream:

```
{"urlTitle":"welcome","indexable":true,"statusDate":"12876000930
00","type":"general","smallImageId":10458,"articleId":"10455","v
ersion":1,"id":10456,"title":"Welcome","description":"","userId"
:10134,"userName":" ","smallImage":false,"createDate":"128760009
3000","displayDate":"1201824000000","smallImageURL":"","expirati
onDate":"","status":0,"statusByUserName":" ","reviewDate":"","mo
difiedDate":"1287600093000","content":...
```

Third Example

Ok, now that you a trivial and almost-trivial example, let's do something more interesting. Let's add (and remove) a Journal Article (this is what the initial thread asked about anyway).

Here are two methods: addArticle and removeArticle. They use a hard-coded 60000 for articleId to make removal easy. There are a ton of parameters for addArticle (29 to be exact), and since there are multiple addArticle methods in the JournalArticleServiceUtil class we have to specify a serviceParameterTypes list to tell Liferay which service API we wish to invoke. So the parameter list gets nasty and I didn't do a very good job of coding it to look nice. You can do that though.

```
  public static void addArticle() throws Exception {
        HttpHost targetHost = new HttpHost("localhost", 8080, "h
ttp");
        DefaultHttpClient httpclient = new DefaultHttpClient();
        httpclient.getCredentialsProvider().setCredentials(
                new AuthScope(targetHost.getHostName(), targetHo
st.getPort()),
                new UsernamePasswordCredentials("test", "test"))
;

        // Create AuthCache instance
        AuthCache authCache = new BasicAuthCache();
        // Generate BASIC scheme object and add it to the local
        // auth cache
        BasicScheme basicAuth = new BasicScheme();
        authCache.put(targetHost, basicAuth);

        // Add AuthCache to the execution context
        BasicHttpContext ctx = new BasicHttpContext();
```

More Notes:

- I specified "" (blank) strings for the structureId and templateId. This means that the article has no structure and no template, so it will just be interpreted as raw

HTML (as though you had created a new Web Content Article and didn't specify a structure or template).

- The content parameter is XML with the content inside of a CDATA block. If there were an associated structure for this content, the content would look different. That's for an advanced reader to explore.
- The timestamps use Java's Calendar method to set the display date and other dates.
- The articleId is hard-coded to 60000. If you wish to auto-generate specify `autoArticleId` to be `true`.
- The `serviceContext` parameter is special and tricky and not well documented. The "content" of the parameter is a JSON-serialized instance of the `com.liferay.portal.service.ServiceContext` class. For other services, it might require a more complex serialized instance of the ServiceContext class. For example, instead of `{}` (which, which decoded, results in a simple new instance of the ServiceContext class with `null`/empty/blank/0 for all of its properties), one might have something like `{scopeGroupId:themeDisplay.getScopeGroupId()}` (I took this from this example). Anyway, for this example, an empty `{}` works.

If `addArticle` works, you should get in return a serialized version of the new article:

```
{"urlTitle":"test-json-article","indexable":true,"statusDate":""
,"type":"general","smallImageId":14540,"articleId":"60000","vers
ion":1,"id":14538,"title":"Test JSON Article","description":"Tes
t JSON Description","userId":10168,"userName":"Test Test","small
Image":false,"createDate":"1307729885333","displayDate":"1310221
080000","smallImageURL":"","expirationDate":"1310912280000","sta
tus":2,"statusByUserName":"","reviewDate":"1310912280000","modif
iedDate":"1307729885333","content":"<?xml version='1.0' encoding
='UTF-8'?><root available-locales=\"en_US\" default-locale=\"en_
US\"><static-content language-id=\"en_US\"><![CDATA[<p>\n\ttest
content<\/p>]]><\/static-content><\/root>","templateId":"","grou
pId":10156,"resourcePrimKey":14539,"structureId":"","statusByUse
rId":0,"companyId":10131,"uuid":"d2a09ad8-43d5-476b-bcb9-3a16214
09835"}
```

Since `deleteArticle` returns `void`, you won't get anything (But the article should be gone, which you can verify in the GUI, or via a database browser).

Good luck!

Tweet 2 Like 3 +1 1 Digg

BlinkList del.icio.us Newsvine Technorati

12641 Views, 15 Comments

Showing 15 Comments

Frans Thamura
6/11/11 9:28 AM

we have a code, cimande www.sf.net/projects/cimande

we want to become rest server (json) for liferay.

can share code? and is it using liferay ide?

Jay Patel
6/12/11 11:42 PM

Very well explained....Thanks James

Jignesh Vachhani
6/14/11 2:47 AM

Nice Article James !!! Thanks !!!

Sohui Gu
6/17/11 7:29 PM

Great job. Thanks.

filip Lagerlov
8/24/11 4:59 AM

Yes but how do u make a json request to a custom service?

James Falkner
8/25/11 1:31 PM

There is some additional work in Liferay 6.0 that needs to be done to enable remote JSON services for your custom service. I'll research that. In the meantime I believe in 6.1 this will be done for you using the code from http://issues.liferay.com/browse/LPS-14359?page=com.atlassian.jira.plugin.system.issuetabpanels%3Aall-tabpanel

filip Lagerlov
8/25/11 11:44 PM

Thanks, i figured it out how to make it work in liferay 6 tomcat, u need to move the projectname-portal-service.jar file "generated by ant build-service" and put it inside the tomcat \bundles\tomcat-6.0.29\lib\ext,

filip Lagerlov
8/25/11 11:45 PM

all creds to this post

http://www.liferay.com/zh/community/forums/-/message_boards/message/7779777

Hitoshi Ozawa
11/28/11 1:42 PM

Your example 3 would not work in 6.1 because they deleted addArticle interface.

Rajeev Kumar
4/15/12 5:03 AM

hi hitoshi,
I am using liferay-portal-6.1.10-ee-ga1 and trying example 3 and as you stated not working, is there work around for it, this is something urgent, pls respond.

James Falkner
4/16/12 8:35 AM

Check out http://www.liferay.com/community/forums/-/message_boards/message/13489024 I have re-done the example 3 for Liferay 6.1.

python shi
8/5/12 8:20 PM

The Apache Commons HTTPClient libraries if it is commons-httpclient-3.0.1.jar? why in my ide, prompt constructor HttpHost(String, int, String) is undefined of HttpHost, and other class also undefined.

James Falkner
8/6/12 7:03 AM

"python shi" : There were many changes in the Apache HTTPClient libraries from 3.x to 4.x - I used 4.x as it's the latest. It should be possible to port my samples to use 3.x. But the errors you are getting are expected (due to the API changes in httpclient).

python shi
8/7/12 3:01 AM

OK, thanks, I use the 4.2.1, it is ok.

mikel basabe
9/6/12 2:37 AM

Hi everybody,

First of all, I would like to congratulate James on this post. It's really useful.

But I have a doubt about how you can call "/assetentry/get-entries" method which needs a "com.liferay.portlet.asset.service.persistence.AssetEntryQuery" object as parameter. Any idea?