

2015-12-10 · Chris Mair



workshop

for new users

AlpineBits?

- interface specification
- based on XML and HTTP(S)
- open (specs are CC-By-ND), everybody is free to implement it
- all XML messages validate against OTA (= tourism industry standard schema for XML)
- AlpineBit Alliance: group of 14 local SMEs

!AlpineBits

- AlpineBits is **not** a software package
- AlpineBits is **not** a company
- AlpineBits is **not** a database, service or website
- you **cannot** “get a ready made product” called AlpineBits, but you **can** teach your software how to “speak” AlpineBits

why the name?

- yes, it used to be an acronym
- **A** Lightweight **P**latform for **IN**terchanging **E**ssential **B**ooking **I**nformation and **T**ravel **S**pecifics
- everybody thought this was stupid, so it was dropped after the first release in 2011
- (I'm still offended ;)

resources (part 1)

- <http://www.alpinebits.org/>
- get the AlpineBits 2015-07 zip (**get it now!**)
 - the specification document (PDF, 65 pages, color-print it to anger your boss ;)
 - XML message sample files
 - XML-schema (XSD) & RNG-schema files

client-server

- understand what we mean when we say:
 - **client** (the software than initiate the comunication with a request)
 - **server** (the software that answers with a response)
- for example: a backend of a website running on a webserver can be an AlpineBits client!

exchange what data?

- FreeRooms (= notify about room availability)
- GuestRequests (= get quote or booking requests)
- SimplePackages (= notify about package availability)
- Inventory (= notify about room category info)
- RatePlans (= notify about rate plans)

my usage?

- think a moment where AlpineBits will fit into your system. will you be a server, a client or both? what kind of data do you need to exchange?
- if you're lost, as a starting point, watch the video on the FAQ site (<http://www.alpinebits.org/faq/>)
- once again: AlpineBits is just an interface specification - it will not magically send you data or accept data from you (it is not a service!), you still need to agree with your counterpart

getting started with the implementation

- skim chapter 1
- read chapter 2
- realize what you've just read: you need to be able to send (or receive) HTTPS POST-requests with content-type multipart/form-data using basic authentication
- how are you going to do that?

really, how are you going to do that?

- AlpineBits is (of course) language agnostic
- on <https://github.com/alpinebits/code-snippets> we collect code snippets (currently there is something in C# and in PHP, contributions welcome)
- try to see what your environment offers (e.g. **libcurl** for PHP or **Apache HttpComponents** for Java), use existing APIs if possible, don't reinvent the wheel!

a basic client

- parameters: action and request
- simple case: query the server version, just need:
 - `action=getVersion`
- you could do this with just a HTML form, but we're going to use PHP and **libcurl**
- emphasis on **libcurl** (I'm not interested in PHP)

libcurl client code 1

```
# $endpoint is the server's https-url
$cu = curl_init($endpoint);
curl_setopt($cu, CURLOPT_RETURNTRANSFER, true);
curl_setopt($cu, CURLOPT_POST, true);
curl_setopt($cu, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
# note https is mandatory!
curl_setopt($cu, CURLOPT_PROTOCOLS, CURLPROTO_HTTPS);
# $authstr contains the credentials in the form "chris:secret"
curl_setopt($cu, CURLOPT_USERPWD, $authstr);
curl_setopt($cu, CURLOPT_HTTPHEADER,
    array("X-AlpineBits-ClientProtocolVersion: 2015-07",
        "X-AlpineBits-ClientID: testclient.php v. 1.0"
    )
);
```

libcurl client code 2

```
$data = array( "action" => "getVersion");

curl_setopt($cu, CURLOPT_POSTFIELDS, $data);

$output = curl_exec($cu);
$info = curl_getinfo($cu);
curl_close($cu);

if ($info["http_code"] != 200) {
    echo "Oops: http status code " . $info["http_code"] . "\n";
}
echo "I said:\n";
print_r($data);
echo "Server said:\n";
echo $output;
```

libcurl client outcome

```
I said:  
Array  
(  
    [action] => getVersion  
)  
Server said:  
OK:2015-07
```

getVersion, what else?

- read chapter 3 - the “housekeeping actions” are:
 - getVersion
 - getCapabilities
- try getCapabilities...
- note: it is a client's responsibility to check for server capabilities before trying to use them!

easy, isn't it?

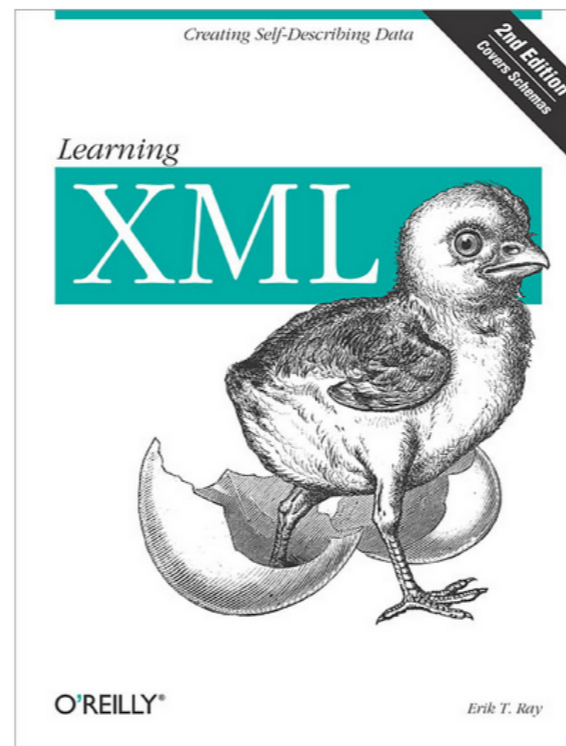
- ask your counterpart to create a test account for you (I'm using one as we demo this right now!)
- start using https (almost) immediately, avoid going into production without testing https
- implement and test error handling - do not underestimate this - implement sensible logging (when your counterpart calls and ask about an error she got from your server, you will look like a fool without some logging at hand)
- AlpineBits 2015-07 has added the support for gzip-compressed requests (not on the housekeeping, though)

great, now let's talk

- read the intro to chapter 4 (the color-table)
- read the section you're interested in, for example “FreeRooms”
- let's try to get our client to notify the server about a room availability! we need to send:
 - action = OTA_HotelAvailNotif:FreeRooms
 - request = [an XML document]

XML, you say?

- if XML ist new to you, spend some time understanding it (as a developer), you might want to get a book such as this:



a sample message

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_HotelAvailNotifRQ [...]>
  <UniqueID Type="16" ID="1" Instance="CompleteSet"/>
  <AvailStatusMessages HotelCode="2286">
    <AvailStatusMessage BookingLimit="10" BookingLimitMessageType="SetLimit">
      <StatusApplicationControl Start="2016-08-01" End="2016-08-10"
        InvTypeCode="double" InvCode="101S" />
    </AvailStatusMessage>
    <AvailStatusMessage BookingLimit="1" BookingLimitMessageType="SetLimit">
      <StatusApplicationControl Start="2016-08-21" End="2016-08-30"
        InvTypeCode="double" InvCode="101S" />
    </AvailStatusMessage>
  </AvailStatusMessages>
</OTA_HotelAvailNotifRQ>
```

let's send it!

just fill the right parameters...

```
$data = array(  
    "action" => "OTA_HotelAvailNotif:FreeRooms",  
    "request" => file_get_contents("xml/FreeRooms-OTA_HotelAvailNotifRQ.xml")  
);
```

... and get a nice success message back

```
<?xml version="1.0" encoding="UTF-8"?>  
<OTA_HotelAvailNotifRS [...]>  
    <Success/>  
</OTA_HotelAvailNotifRS>
```

easy, isn't it?

- well... you used the file from the samples in the zip file, right?
- in real life you need to construct the XML message you wish to send from your database system or parse the XML message you receive
- it really helps if you understand XML concepts: **wellformed** documents, **parser** APIs (SAX, DOM), **schemas**, **valid** documents, possibly **XPath**...

some tools: xmllint

- find a tool that suits your environment to:
 - check a document for wellformedness
 - validate a document against the schemas in the zip file
- Linux and OS X comes with a command line tool called “xmllint”...

xmllint in action

oops: not wellformed:

```
$ xmllint --noout xml/FreeRooms-OTA_HotelAvailNotifRQ.xml
xml/FreeRooms-OTA_HotelAvailNotifRQ.xml:30: parser error : Opening and ending tag mismatch:
AvailStatusMessages line 20 and vailStatusMessages
    </vailStatusMessages>
      ^
```

oops: not valid against the AlpineBits XSD:


```
$ xmllint --noout --schema schemas/alpinebits.xsd xml/FreeRooms-OTA_HotelAvailNotifRQ.xml
xml/FreeRooms-OTA_HotelAvailNotifRQ.xml:16: element OTA_HotelAvailNotifRQ: Schemas validity
error : Element '{http://www.opentravel.org/OTA/2003/05}OTA_HotelAvailNotifRQ': Missing child
element(s). Expected is ( {http://www.opentravel.org/OTA/2003/05}AvailStatusMessages ).
xml/FreeRooms-OTA_HotelAvailNotifRQ.xml fails to validate
```

don't waste time

- sending not wellformed or invalid documents wastes everybody's time
- test the code that constructs your XML messages extensively, don't rely on you counterpart to find validation errors in your document!
- test corner cases and read the specs carefully

more tools: testing machine

<http://alpinebits.testingmachine.eu/validator>

AlpineBits message validation service 

Validate your message by file upload!

This message was successfully validated!	
AlpineBits version	201507
file name	FreeRooms-OTA_HotelAvailNotifRQ.xml
file size (bytes)	1094
validates against OTA 2015A XSD (only AlpineBits relevant root elements are considered)	yes
validates against AlpineBits 2015-07 XSD	yes
validates against AlpineBits 2015-07 RNG	yes

Select another message to validate.

Version:

File: No file selected.

validates against:

- OTA
- AlpineBits XSD
- AlpineBits RNG

what was OTA, again?

- <http://www.opentravel.org/>
- AlpineBits strives to be a subset of OTA, that is all document that validate against the AlpineBits schema(s) should validate against OTA, but not viceversa
- if you want to navigate the (vast) OTA schema you might want to use this tool:
 - <http://adriatic.pilotfish-net.com/ota-modelviewer/>

experiments

- what happens in our FreemRooms-client when:
 - we use wrong credentials
 - the parameter values are wrong
 - the document is not well formed / not valid
 - we send rooms for an unknown hotel
 - we send a start date > end date

layers

- realize that there can be errors at any level (low level, authentication, before the XML is even processed, problems with the XML, problems with the meaning of the data)
- test error catching in all layers
- test
- test more ;)

Oook, but I need a server...

- handling the multipart/form-data POST requests should be easier on the server, since web developers are used to do it all the time ;)
- it really depends on your language and framework
- the github code snippets directory has a sketch of the first part of a server, that I can demo, but won't even bother copying into the slides, since it's too PHP-specific anyway... ;)

great news, or not?

- not so fast... handling the POST is quite easy, but:
 - think about parsing the XML data
 - how do you map it into your (existing) database?
 - generally you have to implement (almost) everything for a given action, because you cannot foresee what the client will send

now have a look at

- the specification for:
 - **FreeRooms** (= notify about room availability)
 - **GuestRequests** (= get quote or booking requests)
 - **Inventory** (= notify about room category info)
- (no slides here, I'll give an overview using the AlpineBits document ;)

resources (part 2)

- we use the forum / mailing list at
<https://groups.google.com/forum/#!forum/alpinebits>
to discuss about AlpineBits
- the code repository is at
<https://github.com/alpinebits>



the end