Fig. 1

1. interface/serverSoftware?

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

A while ago Mark Taylor and I stumbled into two use cases that may call for a slight addition to VOResource. These are, roughtly:

Use case “global”: There’s a bug in DaCHS 1.1 Mark is working around in TOPCAT. Can he drop the workaround since there’s no DaCHS 1.1 around any more?

Use case “local”: A client offers extra features when a specific server software is on the other end and hence needs to know what’s on the other side.

(cf. Fig. 1)

2. A Possible Solution

We could put a new element in vr:Interface, perhaps like this:

```
<interface role="std" xsi:type="vs:ParamHTTP">
  <accessURL use="base">http://victor:8080/tap</accessURL>
  <serverSoftware>DaCHS/1.2</serverSoftware>
  <serverSoftware>python/2.7.16</serverSoftware>
</interface>
```

In RegTAP, these would end up in res_detail rows with a key of capability/interface/serverSoftware.

Pulling this from VOSI capabilities would in principle satisfy the local, keeping it in res_details pretty much the global use case.

3. But: RFC 2616, Sect. 14.38

For those not in the numbers business: That’s the HTTP 1.1 standard.

```
14.38 Server

   The Server response-header field contains information about the
   software used by the origin server to handle the request.
```

People who don’t care to put something sensible in the Server: header probably won’t bother to do it in their capabilities.

So, we don’t need serverSoftware for the local use case.

There’s a caveat, though: many VO servers run behind HTTPS terminators and other things that may swallow or distort the Server header (though proxies aren’t supposed to touch server). Hence, perhaps the server header isn’t quite enough for the local use case yet.

4. But: Stale Records?

It’s unlikely that server operators will push out new registry records when they update their software. In particular, I don’t think I’d even want to cause a full reharvest from within DaCHS after an update.

Hence, serverSoftware in RegTAP will always be really stale.

So much for the global use case.

Conventional all-VO querying will still work fine for determining server software for all protocols but SCS (Hints: DDC adoption! SCS update!).

5. On the Other Hand:

Census of TAP services, 2019-09-26 using

```
for svc_rec in pyvo.registry.search(servicetype="tap"):
    res = request.urlopen(svc_rec.access_url+"/capabilities")
    server_software = res.info()["server"]
```

The full program is available in an attachment of this PDF.

Top results:

```
Apache-Coyote/1.1                        56
Apache                                   26
TwistedWeb/14.0.2                        24
TwistedWeb/16.6.0                        17
Microsoft-IIS/10.0                        6
None                                      6
Apache/2.2.15 (Scientific Linux)          2
nginx/1.12.2                              2
```

That’s not useful for VO clients. . .

See PDF attachment(s): tapstats.py

6. So?

I've fixed DaCHS; in Heidelberg, it already produces:

```
Server: DaCHS/1.0.1 twistedWeb/16.6.0
```

But will that help?

A serverSoftware element would at least be robust against active HTTP components.

But then: should we work around sloppy practices and broken components in VOResource?