Copy of Design - Asynchronous REST API

Proposed Asynchronous Pattern

Polling and Notification (see UCAR reference)

Initial Request

Without any special client knowledge:

Client makes a request, e.g. "GET /some/resource"

Branch:

- Server requires asynchronous, responds with 4xx

- Server doesn't require asynchronous, blocks, and proceeds as normal.

With an asynchronous-aware client:

Client makes a request, e.g. "GET /some/resource" and accepts asynchronous response with HTTP headers:

- Accept-Asynchronous: polling
- Accept-Asynchronous: notify; Asynchronous-end-point: http://some-webhooks-uri/
- Accept-Asynchronous: response; Asynchronous-end-point: http://some-webhooks-uri/

If the server decides not to send an asynchronous response, blocks and proceeds as normal.

If the server wants to send an asynchronous response, server responses with status 202 (ACCEPTED).

Asynchronous response modes:

If the client sends "Accept-Asynchronous: polling", the server will provide a URL for the client to poll for request status. When done processing, the server will send a 301 to the request body.

If the client sends "Accept-Asynchronous: notify", when the server is done processing, the server will send a webhooks-style POST request to the provided endpoint with metadata about the response (response size, what request generated it, etc)

If the client sends "Accept-Asynchronous: response", when the server is done processing, the server will POST the generated response body to the provided endpoint.

Example (Get content; polling)

Client makes a request:

```
GET /some/object/fcr:content
  Accept-Asynchronous: polling
```

Server sends a 202 (Accepted) response with a location for the client to poll for status

```
HTTP/1.0 202
Location: /fcr:status/123456789
```

Client polls location for status

```
GET /fcr:status/123456789
```

Server should send a useful status response, in a serialization TBD. When done processing, server should send a redirect to the result.

```
HTTP/1.0 301
Location: /fcr:status/123456789/fcr:content
Expires: (+24 hours?)
```

Client can pick up the response:

```
GET /fcr:status/123456789/fcr:content
```

The server MAY cache the content response (if appropriate) for some length of time. The server MAY also expose the cached response at the original request endpoint (e.g. /some/object/fcr:content could response immediately instead of requiring asynchronous interactions)

Example (Get content; notify)

TODO

Example (Get content; response)

TODO

Use Cases

Batch Upload

Get Content (delivered somewhere else)

Delete Content (immediately, don't block)

Run fixity check

SPARQL Update

Execute LDPath Query

Export / Import

Backup / Restore

Discussion

1. References
    - http://www.infoq.com/news/2009/07/AsynchronousRest
    - http://www.tbray.org/ongoing/When/200x/2009/07/02/Slow-REST
    - http://www.adayinthelifeof.nl/2011/06/02/asynchronous-operations-in-rest/

    - https://community.jboss.org/message/823036#823036
    - http://www.unidata.ucar.edu/staff/edavis/notes/asynchHTTP-survey.html
2. Regarding async HTTP-API
    - "For any and all PUT/POST/DELETE operations, we return “202 In progress” and a new “Status” resource, which contains a 0-to-100 progress indicator, a target_uri for whatever’s being operated on, an op to identify the operation, and, when progress reaches 100, status and message fields to tell how the operation came out.
    - The idea is that this is designed to give a hook that implementors can make cheap to poll.
    - However, since most of the clients with which we are concerned will be machines and not browsers, we could use webhooks for the purpose.
3. JAX-RS-2.0 has the async notion built into its Client spec
    - Jersey reference implementation: https://jersey.java.net/documentation/latest/async.html