OMERO JSON

RIAD GOZIM

Outline

- JSON basics
- OmeroWeb - Webclient, Webgateway, Webadmin
- OmeroWeb - JSON API

JSON - the basics

- JSON - Javascript Object Notation
- Basic type support
  - Number (signed decimal)
  - String (unicode)
  - Boolean
  - Array
  - Object
  - null

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

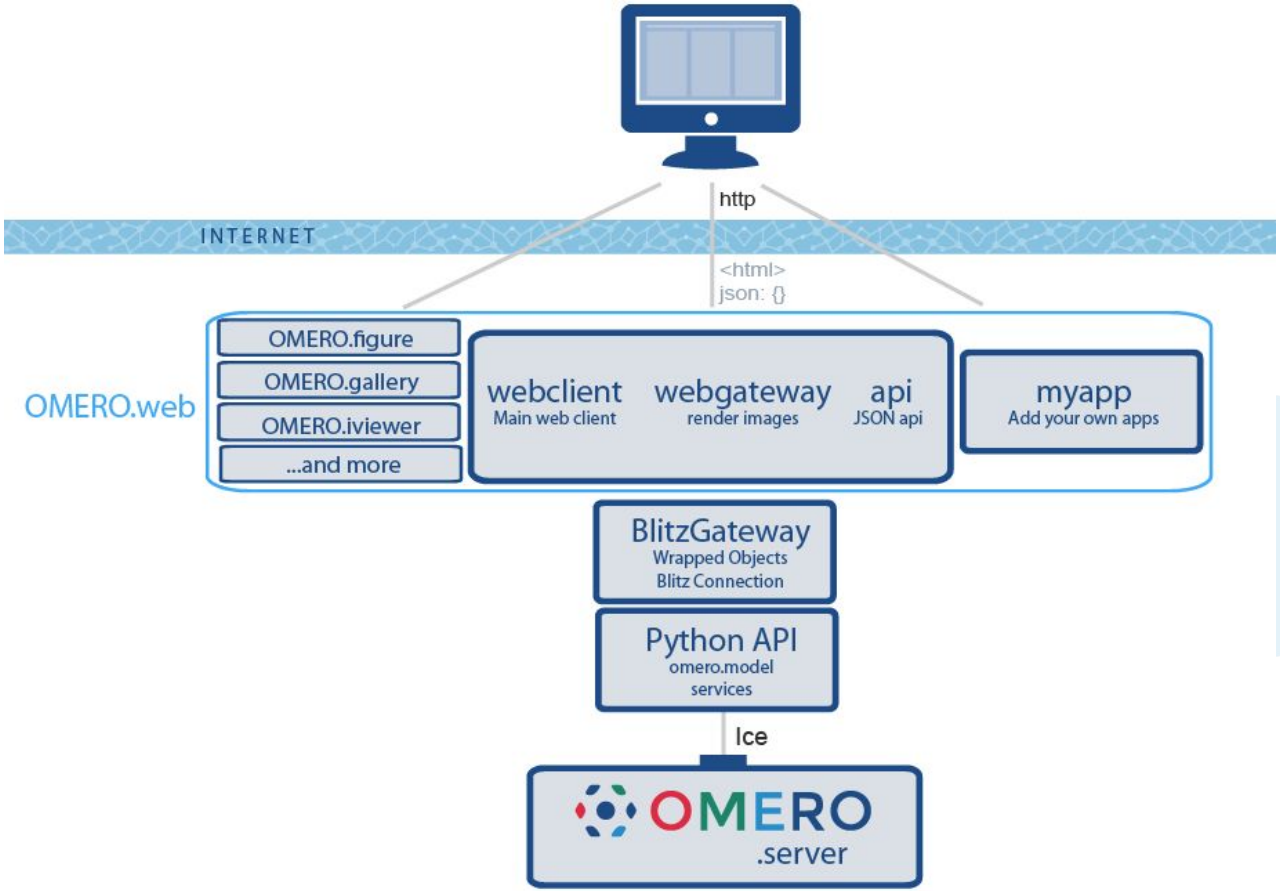
JSON - use cases

- Data interchange
  - Server - Client
- Configuration
- Schema
  - Currently in draft 6

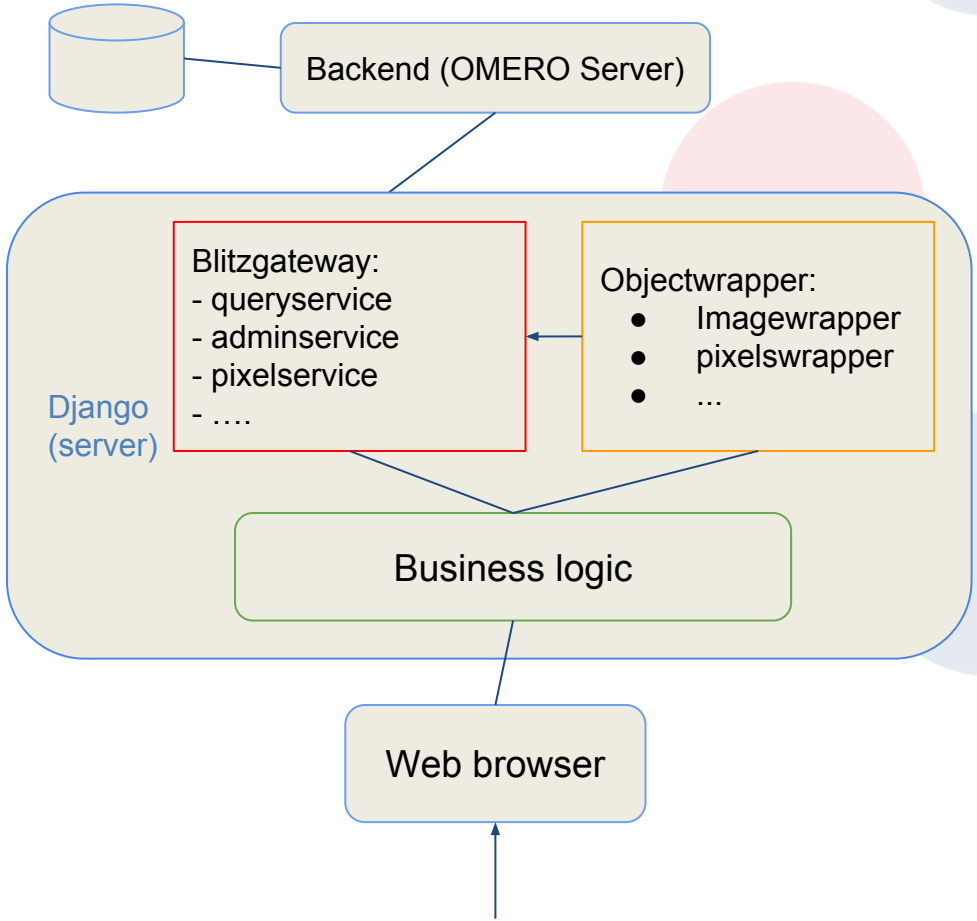
JSON - pros

- It’s everywhere
- Human readable
- Easy to modify

OMERO Web - quick recap



OMERO Webclient



1. Navigate to OMERO Web on browser
2. Django Server loads ***view*** for the given **URL** and sends it to the browser in a template (HTML)
3. HTML on browser loads in ***static*** files, inc. Javascript and CSS
4. Javascript code triggers various AJAX requests to the Django Server
5. Django serializes response data as JSON
6. Browser handles JSON response

OMERO Webclient

Scenario 1

1. Django *View* populates HTML fields and passes down required data for the page
2. Browser displays ‘pre-rendered’ HTML
3. Button on HTML page triggers Ajax request, transmitting and receiving data in JSON

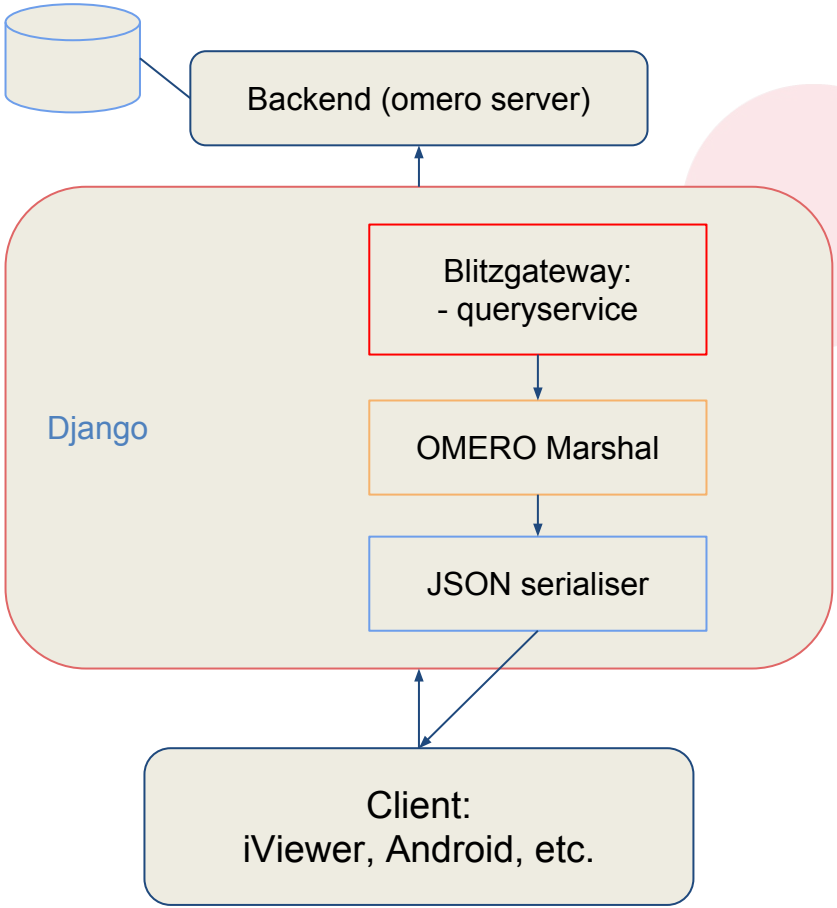
Scenario 2

1. Django *View* passdown HTML page with minimal processing
2. Browser loads accompanying Javascript with HTML
3. Browser calls to URL endpoints with Ajax to load data as a JSON response

OMERO Web

- Django applications
- ‘RPC style’ JSON endpoints
  - URLs as verbs
- Easy setup, difficult to reuse across platforms
- Tight coupling

OMERO REST API - REST



1. Client calls API **URL** [HTTP GET]
2. Django server receives request
3. Constructs SQL statement to run queryservice from Blitzgateway
4. Response from Blitz is *encoded / marshaled* to **Python** via OMERO Marshal
5. Result is converted to **JSON** from **Python** and sent to client

OMERO JSON API

- REST ‘style’ API
- Communicates via HTTP
- Nouns in URL - not verbs
  - /v0/m/plates (noun)
  - /getPlates (verb)

OMERO JSON API - why JSON & REST

- Scalable
- Versionable
- Independant
- Encapsulation

OMERO JSON API - supports

- GET
  - Single resources
  - Collection resources
- HATEOAS (Hypermedia as the Engine of Application State)
- DELETE (limited)
- PUT (limited)
- POST (limited)

OMERO JSON API - work in progress

- Collection resource updates and deletions
- Increase support for POST, PUT, DELETE
- Partial updates via PATCH
- Authorisation (403)
- Consistency

OMERO Web - JSON can do more

- OMERO Webclient - consistency
- Decouple Webclient
- Swagger - for REST API documentation
- Django REST framework?
- Reduce JSON API layers

Resources

Apigee free e-book on REST design

Postman - free REST client testing application

JSON API specification