FAST

Annalist

(“keeper of records”)

Graham Klyne

e-Research Centre, University of Oxford

Oxford e-Research Centre

Digital Economy
Transforming Business and Society

Queen Mary
University of London

EPSRC
Engineering and Physical Sciences
Research Council

The University of
Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Acknowledgements

OeRC: FAST project (digital music)

- Kevin Page, David Weigl, David De Roure, Terhi Nurmikko-Fuller, Alan Chamberlain, Steve Benford, *et al*

JISC RDS: CREAM project (active metadata)

- Cerys Willoughby, Simon Coles, Colin Bird, Iris Garrelfs, Athanasios Velios, Mike Mineter, *et al*

Wf4Ever (research objects, workflows)

- Jun Zhao, Kevin Page, David De Roure, Stian Soiland-Reyes, Sean Bechhofer, Khalid Belhajjame, Carole Goble, Daniel Garijo, Oscar Corcho, *et al*

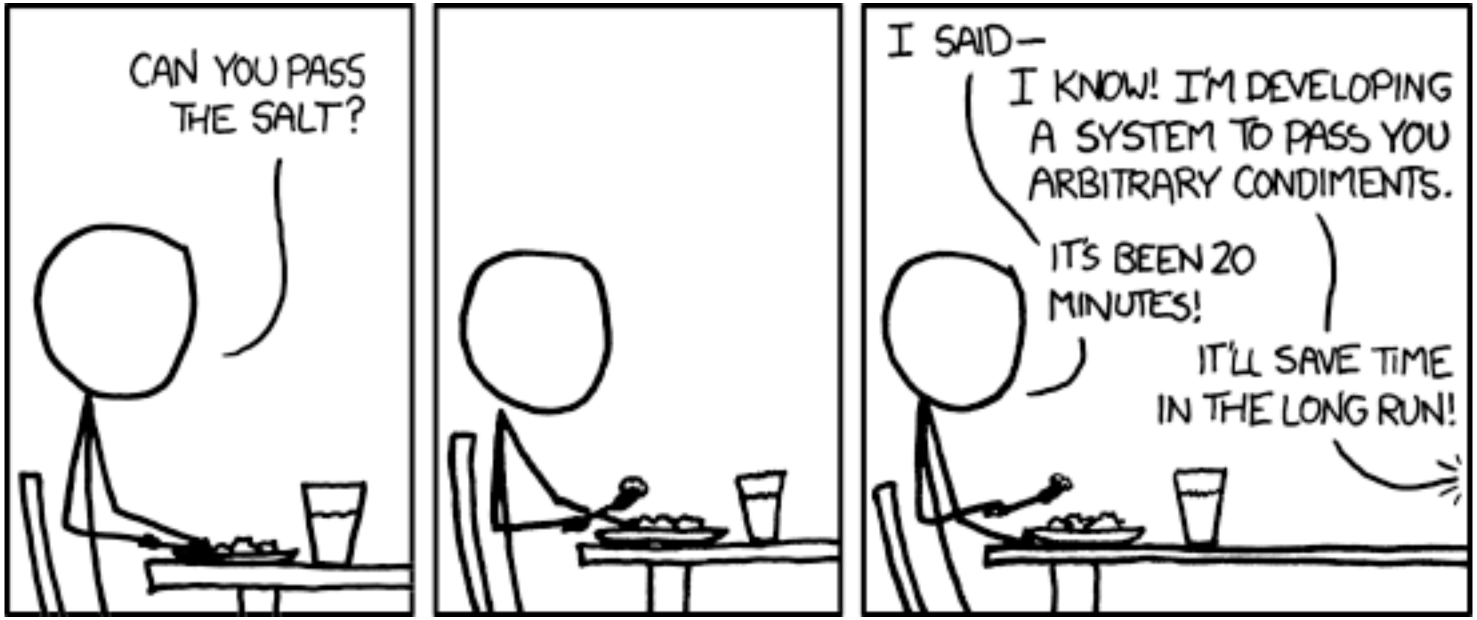
OeRC: Claros project (classical art)

- Donna Kurtz, Robert Kummer, Reinhard Förtsch, *et al*

Oxford (Zoology): IBRG (image bioinformatics)

- David Shotton, Jun Zhao, Alistair Miles, Helen White-Cooper, *et al*

Origins


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

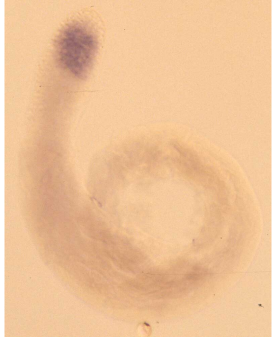
https://xkcd.com/974/

Goal

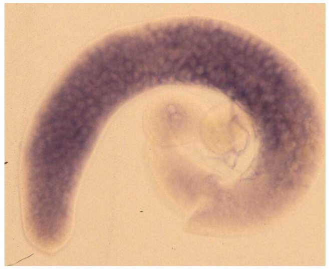
To make it quick and easy for individuals and small teams to create and share linked data on the web

Example: Fly-TED

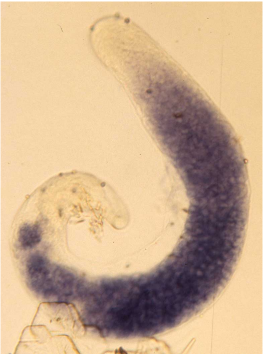
- To an expert observer these images clearly show gene expression at different stages of spermatogenesis
- Each image corresponds to a different combination of gene and a strain of *Drosophila melanogaster* (fruit fly)
- These *in situ* hybridization images are the final result of a complex experimental process
- Reproducibility and interpretation require that the preparatory steps are recorded along with the images and annotations



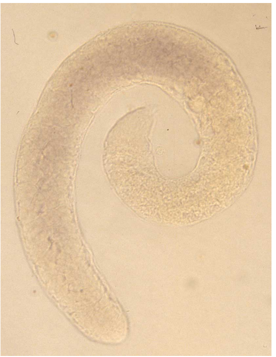
CG2247 wt



CG2247 topi



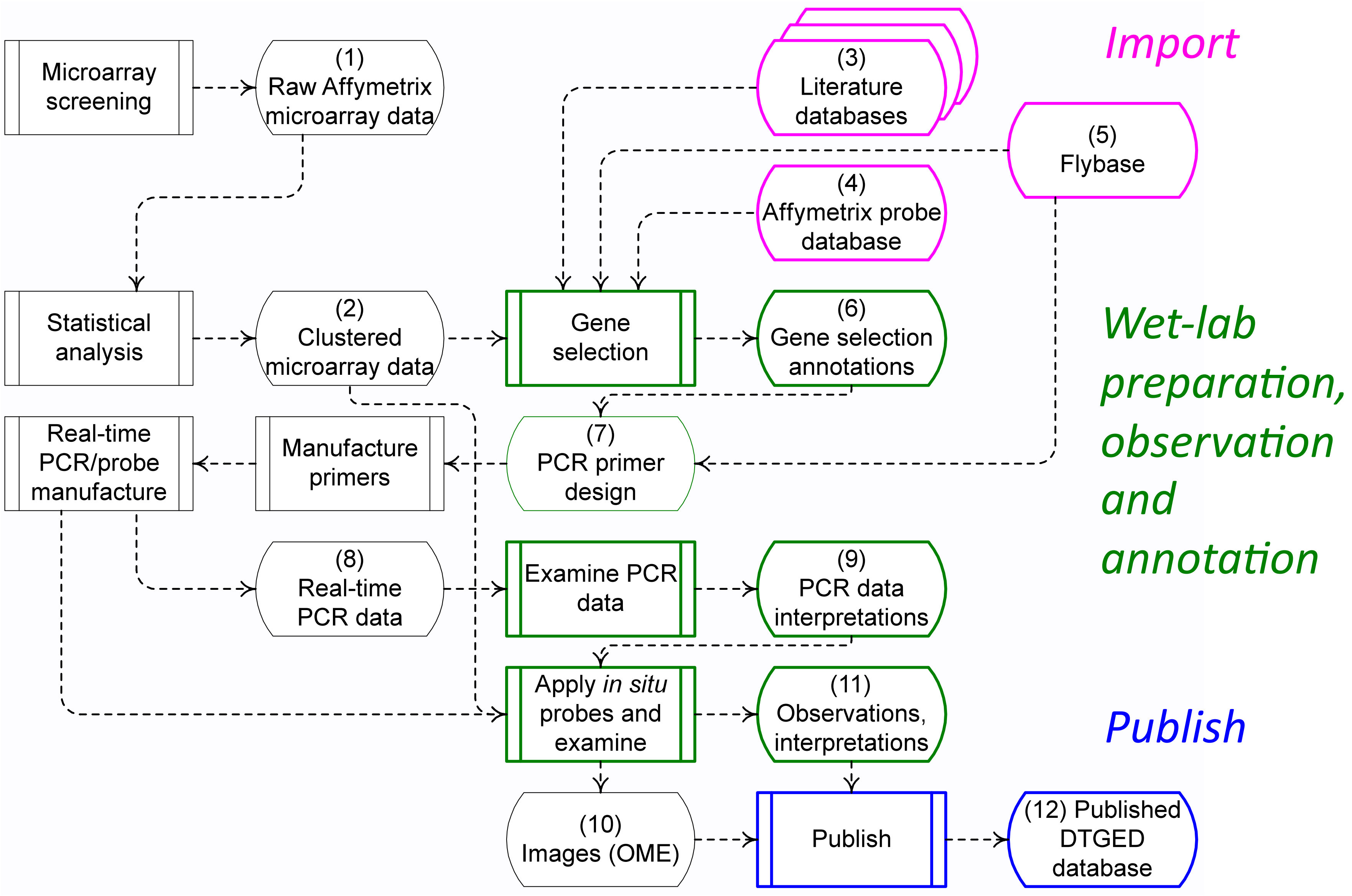
CG12907 aly



CG12907 topi

Images: Dr Helen White-Cooper

Example: Fly-TED data flows


Microarray screening
(1) Raw Affymetrix microarray data
(3) Literature databases
Import
(5) Flybase
(4) Affymetrix probe database
Statistical analysis
(2) Clustered microarray data
Gene selection
(6) Gene selection annotations
Wet-lab preparation, observation and annotation
Real-time PCR/probe manufacture
Manufacture primers
(7) PCR primer design
(8) Real-time PCR data
Examine PCR data
(9) PCR data interpretations
Apply in situ probes and examine
(11) Observations, interpretations
Publish
(10) Images (OME)
Publish
(12) Published DTGED database

Requirements

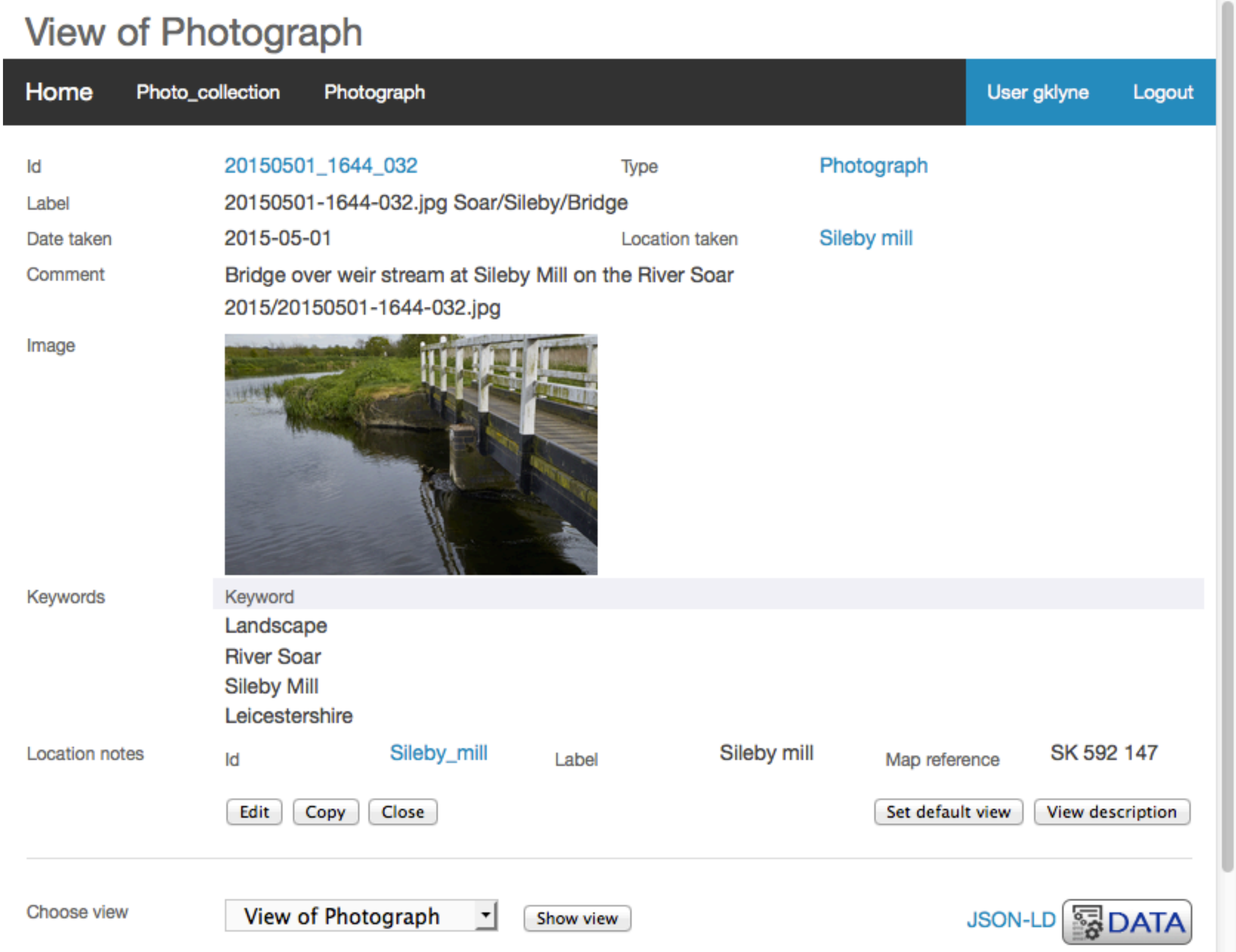
R1: Ease of use: quickly create a simple collection
R2: Ease of use: no programming or HTML coding
R3: Ease of use: no knowledge of RDF and/or OWL
R4: Flexibility: choice of RDF vocabulary used
R5: Flexibility: define or adapt structure of data
R6: Sharability: including online access and offline copying
R7: Remixability: linkable, use domain vocabularies
R8: Portability: move data between systems; not centralized
R9: Sustainable software: use unmodified software
R10: Sustainable of data: standard, easily used format
R11: Exposed data: accessible to independent software
R12: Offline working

Survey

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Callimachus	✘	✘	✘	✔	?	?	✔	✘	✧	✔	✧	?
Semantic MediaWiki	✧	✔	✔	✔	✧	✘	✧	?	✔	✘	✧	✘
Wikidata	✘	✔	✔	✔	✧	✘	✔	✘	✔	✘	✧	✘
Protege	✘	✔	✘	✔	✧	✔	✔	✔	✔	✔	✔	✔
Figshare	✔	✔	✔	✘	✘	✘	✘	✘	✔	✘	✔	✘
ResearchSpace	✔	✔	✔	✘	✘	✘	✔	✘	✔	✧	?	✘
Histcross/Segrada	✔	✔	✔	✧	?	✘	✘	✘	✧	✘	✘	✔
Spreadsheet	✔	✔	✔	✔	✔	✧	✘	✔	✔	✧	✔	✔
Rightfield	✔	✔	✧	✔	✧	✧	✧	✔	✔	✧	✔	✔
Desktop database	✘	✧	✘	✔	✧	✘	✘	✔	✔	✘	✘	✔
CMS	✘	✔	✔	✔	✧	✘	✧	✘	✔	✘	✧	✘
ELN	✔	✔	✔	✧	✧	✧	✘	✘	✔	✘	✧	✘

Annalist	✔	✔	✧	✔	✧	✔	✔	✔	✔	✔	✔	✔

✔	Yes	✘	No	✧	Partial	?	Unknown



View of Photograph

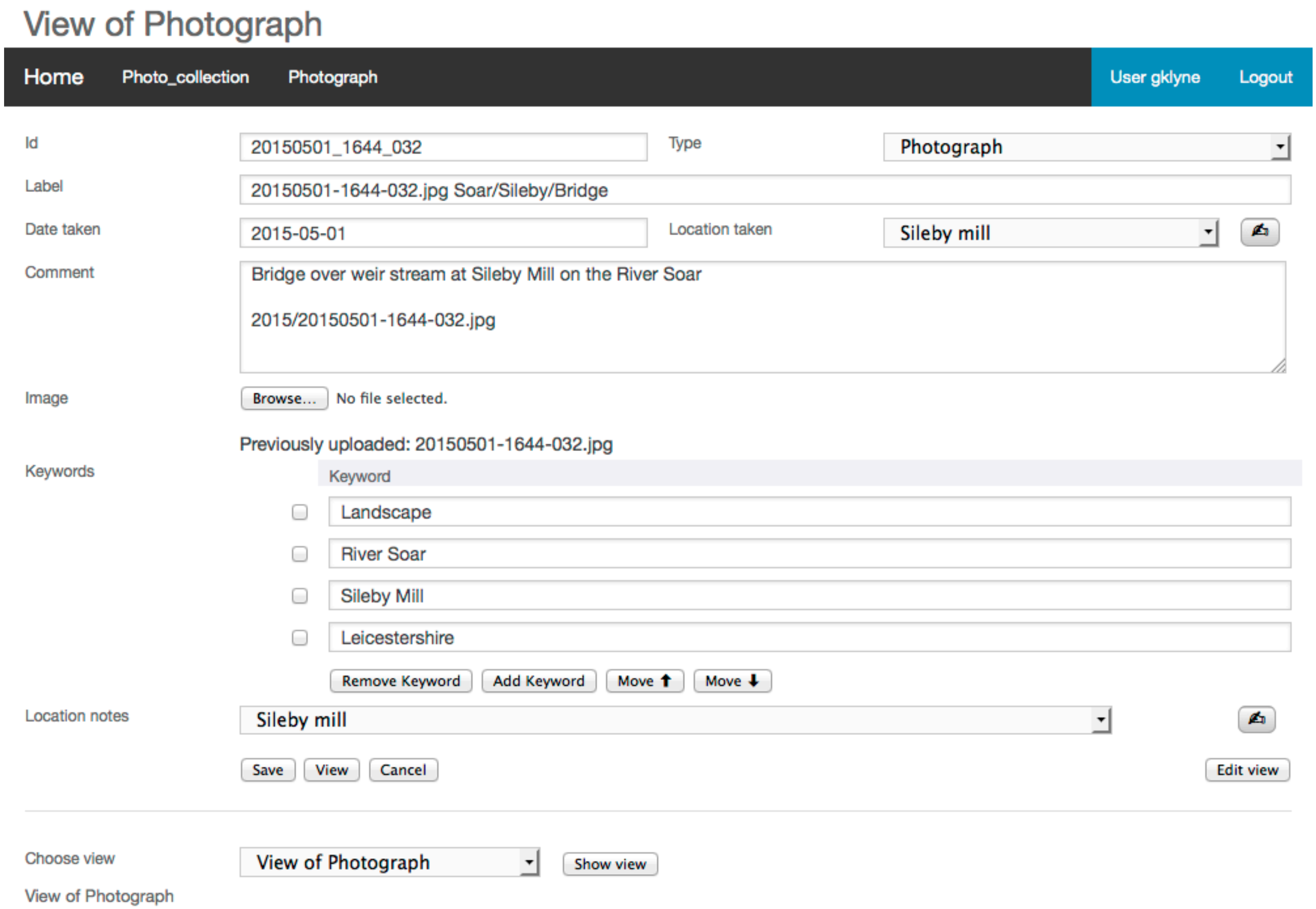
Id	20150501_1644_032	Type	Photograph
Label	20150501-1644-032.jpg Soar/Sileby/Bridge		
Date taken	2015-05-01	Location taken	Sileby mill
Comment	Bridge over weir stream at Sileby Mill on the River Soar 2015/20150501-1644-032.jpg		
Image			

Keywords

Keyword
Landscape
River Soar
Sileby Mill
Leicestershire

Location notes	Id	Sileby_mill	Label	Sileby mill	Map reference	SK 592 147

Choose view: View of Photograph



View of Photograph

Id: 20150501_1644_032

Type: Photograph

Label: 20150501-1644-032.jpg Soar/Sileby/Bridge

Date taken: 2015-05-01

Location taken: Sileby mill

Comment: Bridge over weir stream at Sileby Mill on the River Soar

2015/20150501-1644-032.jpg

Image: No file selected.

Previously uploaded: 20150501-1644-032.jpg

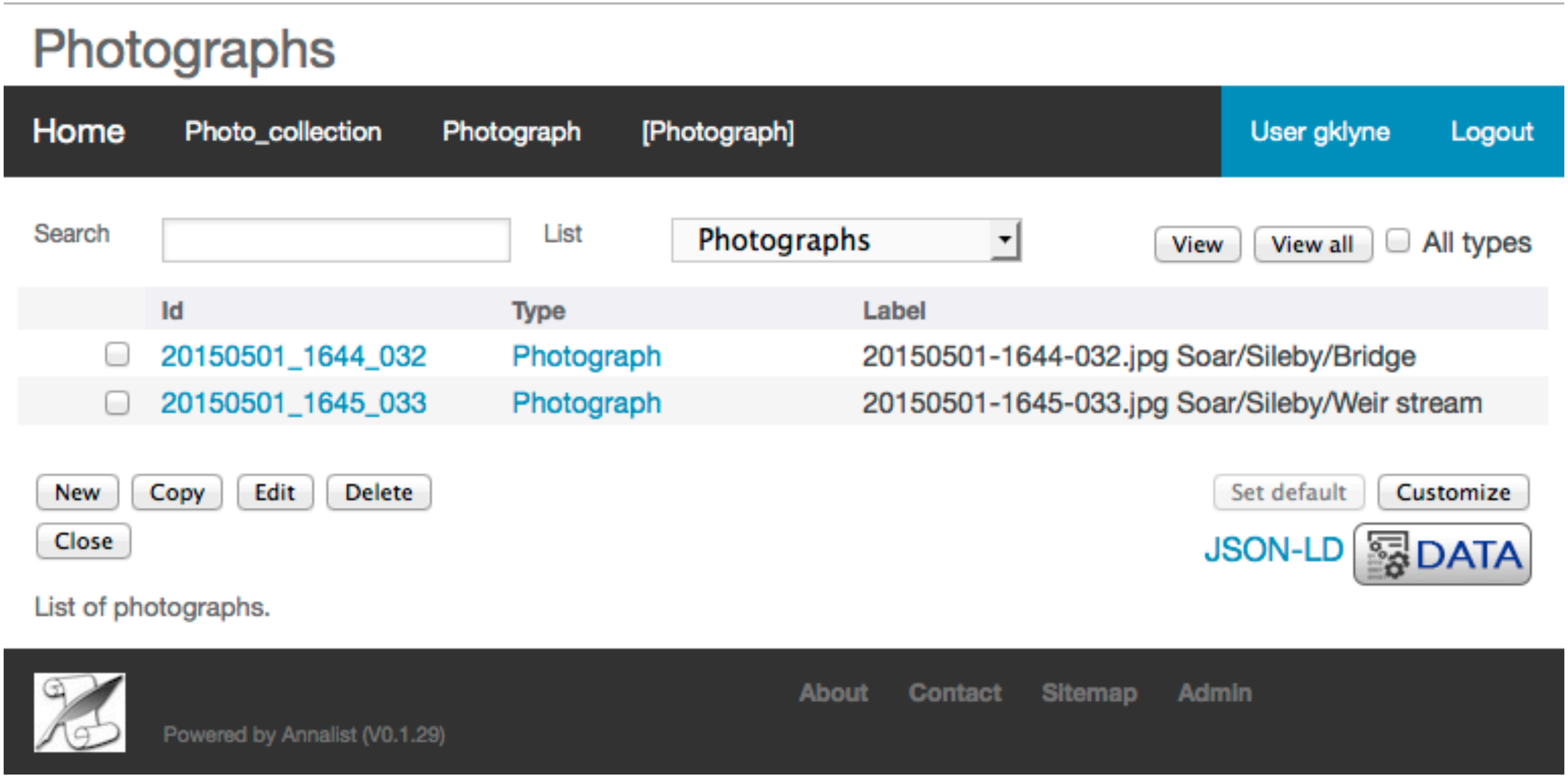
Keywords

	Keyword
□	Landscape
□	River Soar
□	Sileby Mill
□	Leicestershire

Location notes: Sileby mill

Choose view: View of Photograph

View of Photograph



Photographs

	Id	Type	Label
☐	20150501_1644_032	Photograph	20150501-1644-032.jpg Soar/Sileby/Bridge
☐	20150501_1645_033	Photograph	20150501-1645-033.jpg Soar/Sileby/Weir stream

List of photographs.

Powered by Annalist (V0.1.29)

Data:

http://demo.annalist.net/annalist/c/Photo_collection/

Tutorial:

http://annalist.net/documents/tutorial/annalist-tutorial.html

Design philosophy

- Data first, structure later
- Minimize impediments to data entry
- JSON-LD as "view source" for linked data
- Annalist as a part in a wider linked data ecosystem
- Open source, open development

The message that data can tell is not always clear at the outset, but may emerge though the process of observation and collection.

Technical design

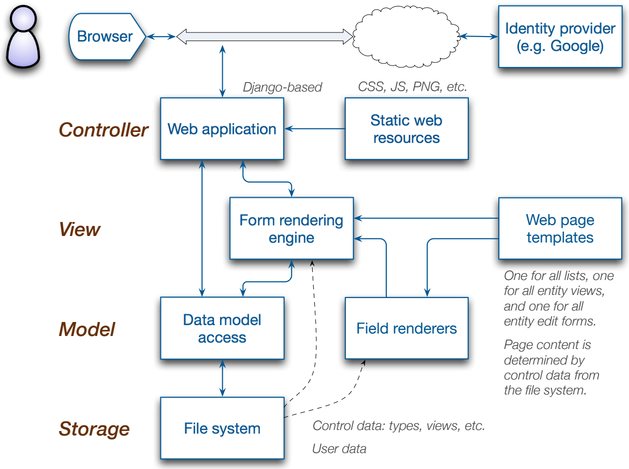
Web server application

Data stored as JSON-LD files

- Can be published by any HTTP server

Customizable form generator

- Definition as JSON-LD, also managed by Annalist
- One definition used for view and edit forms

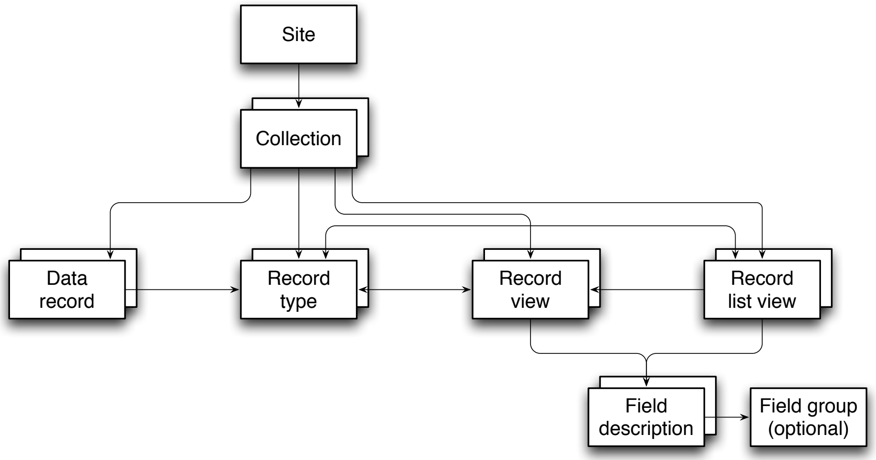


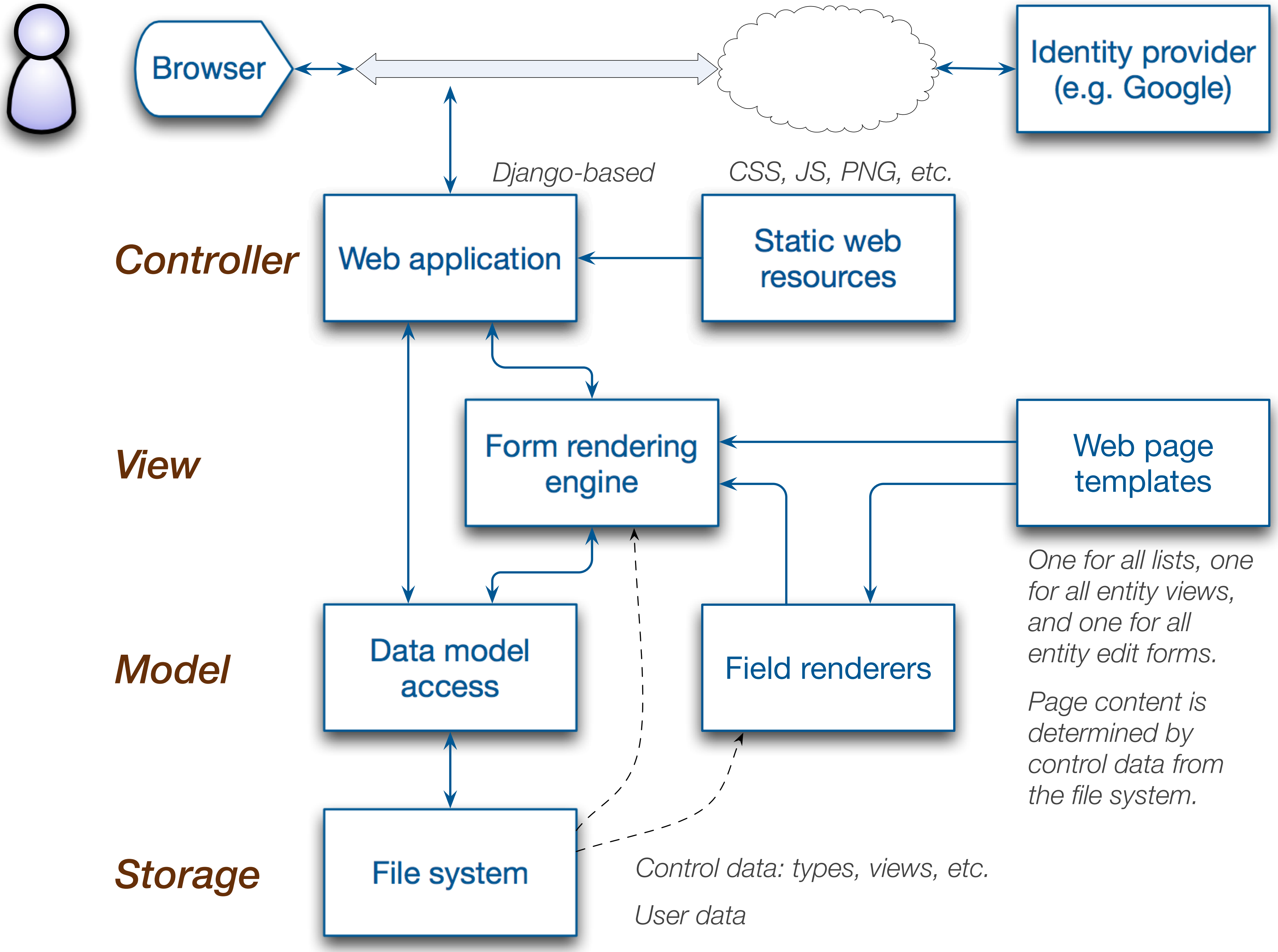
One for all lists, one for all entity views, and one for all entity edit forms.

Page content is determined by control data from the file system.

Control data: types, views, etc.

User data





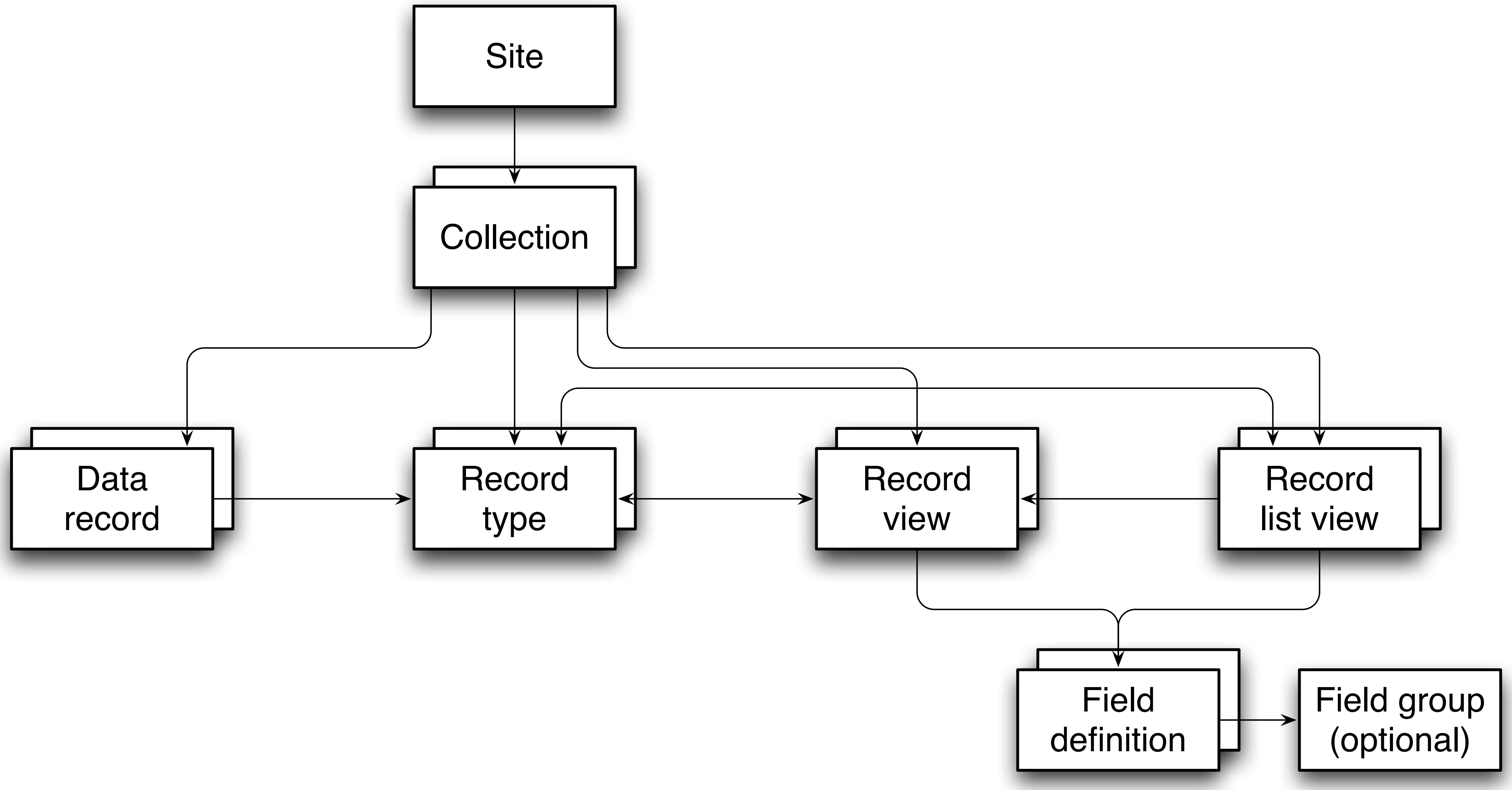
One for all lists, one for all entity views, and one for all entity edit forms.

Page content is determined by control data from the file system.

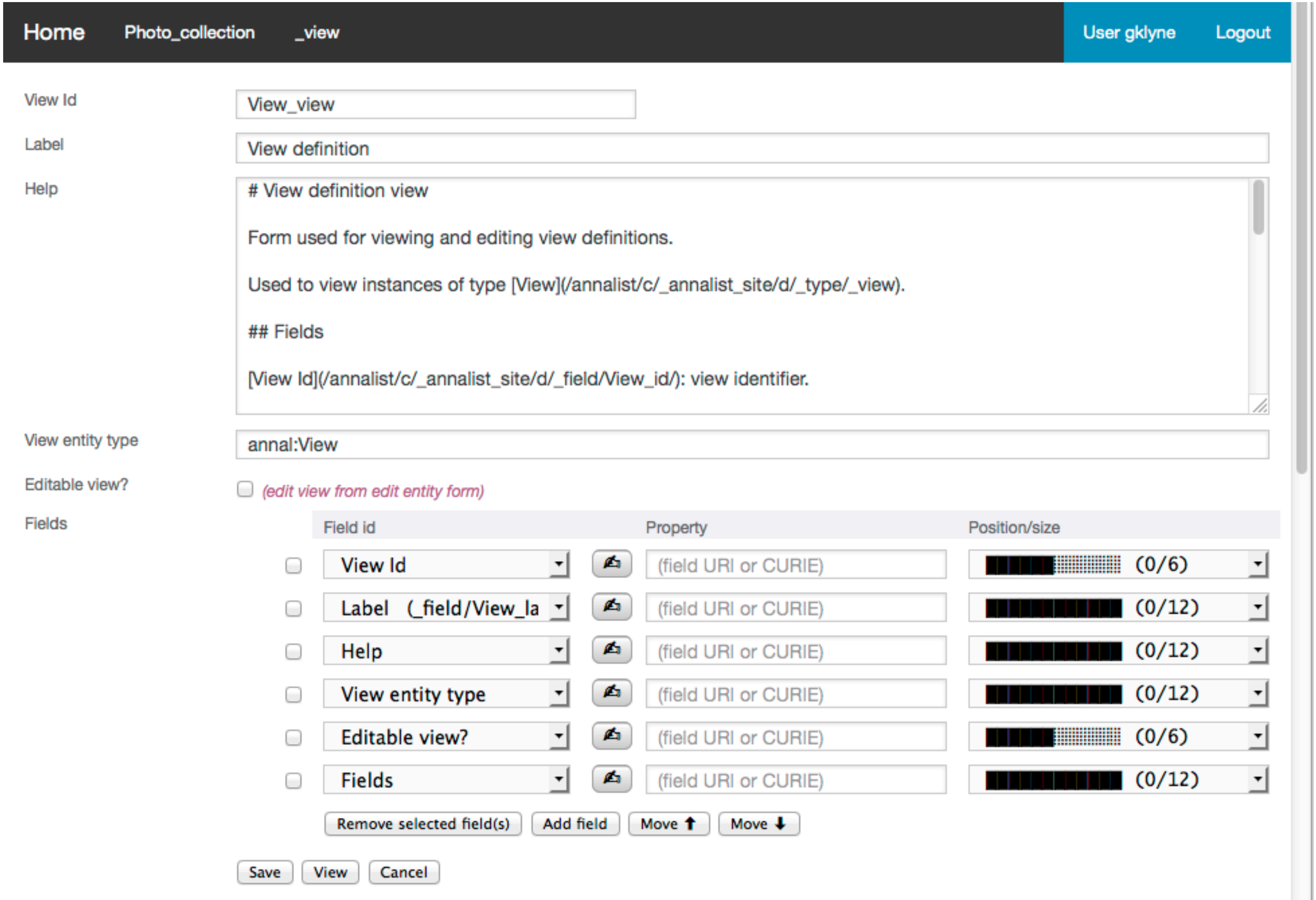
Control data: types, views, etc.

User data

Data model concepts



Configuration self-maintenance



View Id: View_view

Label: View definition

Help:

View definition view

Form used for viewing and editing view definitions.

Used to view instances of type [View](/annalist/c/_annalist_site/d/_type/_view).

Fields

[View Id](/annalist/c/_annalist_site/d/_field/View_id/): view identifier.

View entity type: annal:View

Editable view? □ *(edit view from edit entity form)*

Fields

	Field id	Property	Position/size
□	View Id	(field URI or CURIE)	(0/6)
□	Label (_field/View_la	(field URI or CURIE)	(0/12)
□	Help	(field URI or CURIE)	(0/12)
□	View entity type	(field URI or CURIE)	(0/12)
□	Editable view?	(field URI or CURIE)	(0/6)
□	Fields	(field URI or CURIE)	(0/12)

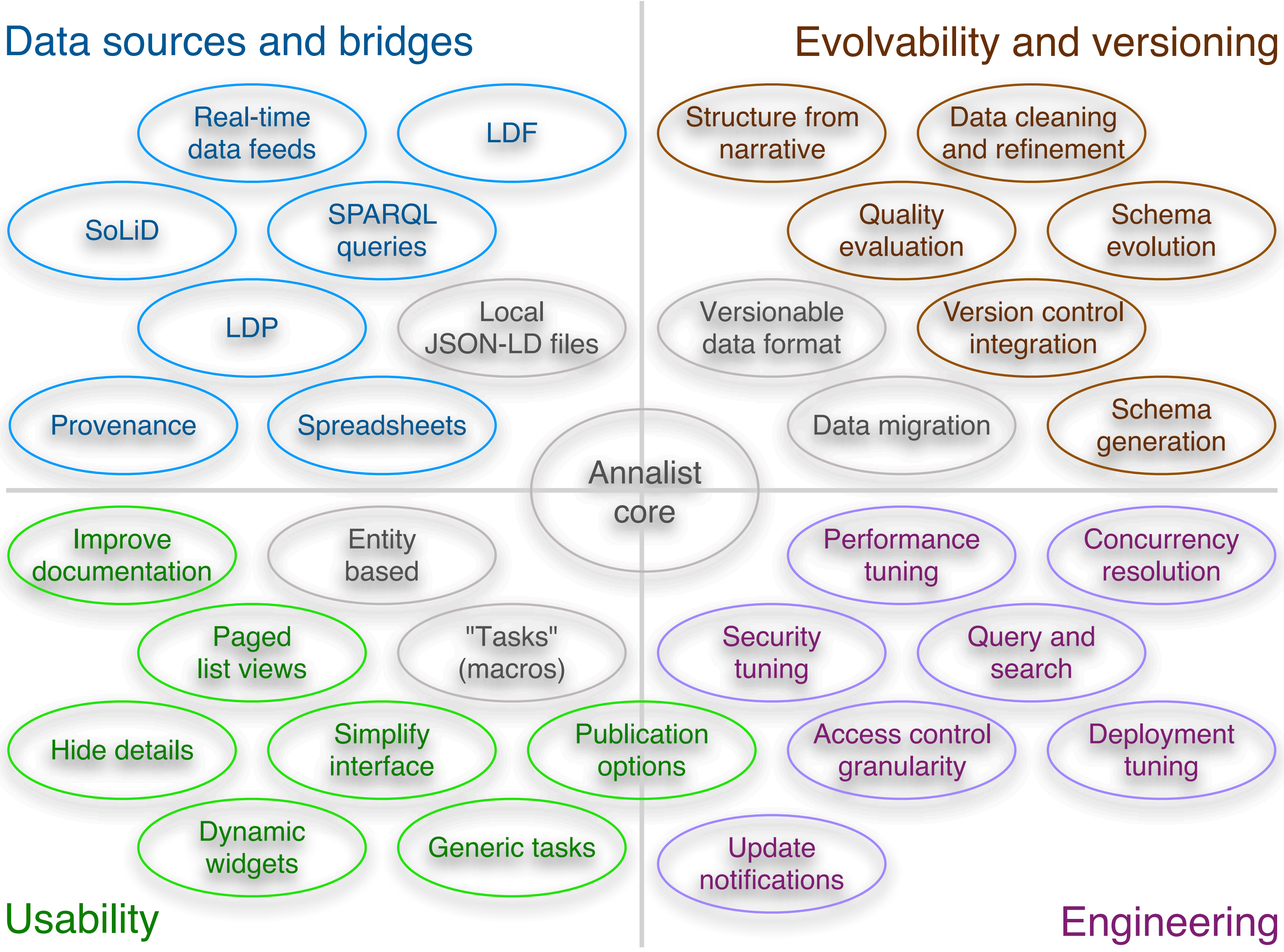
JSON-LD as “view source” for data

```
{ "@id":                "annal:display/View_view"
, "@type":              ["annal:View"]
, "@context":           ["../../coll_context.jsonld"]
, "annal:id":           "View_view"
, "annal:type_id":      "_view"
, "annal:uri":          "annal:display/View_view"
, "annal:record_type":  "annal:View"
, "rdfs:label":         "View definition"
, "rdfs:comment":       "# View definition view\r\n\r\nForm used for viewing ...
, "annal:open_view":    false
, "annal:view_fields":
  [ { "annal:field_id":               "_field/View_id"
    , "annal:field_placement":        "small:0,12;medium:0,6” }
  , { "annal:field_id":               "_field/View_label"
    , "annal:field_placement":        "small:0,12” }
  , { "annal:field_id":               "_field/View_comment"
    , "annal:field_placement":        "small:0,12” }
  , { "annal:field_id":               "_field/View_target_type"
    , "annal:field_placement":        "small:0,12” }
  , { "annal:field_id":               "_field/View_edit_view"
    , "annal:field_placement":        "small:0,12;medium:0,6” }
  , { "annal:field_id":               "_field/View_fields"
    , "annal:field_placement":        "small:0,12” }
  ]}
```

Progress to date

1. A viable tool to create and share linked data
2. Flexible to deal with diverse applications
3. Robust
    - even as a work-in-progress, I have never lost application data due to an Annalist software fault
4. At least approachable for users who are not familiar with RDF

Status and future work



Data sources and bridges

- Real-time data feeds
- LDF
- SoLiD
- SPARQL queries
- LDP
- Local JSON-LD files
- Provenance
- Spreadsheets

Evolvability and versioning

- Structure from narrative
- Data cleaning and refinement
- Quality evaluation
- Schema evolution
- Versionable data format
- Version control integration
- Data migration
- Schema generation

Annalist core

- Improve documentation
- Entity based
- Paged list views
- "Tasks" (macros)
- Hide details
- Simplify interface
- Publication options
- Dynamic widgets
- Generic tasks

Usability

- Performance tuning
- Concurrency resolution
- Security tuning
- Query and search
- Access control granularity
- Deployment tuning
- Update notifications

Engineering

Transition to community project

A public repository is a start:

- https://github.com/gklyne/annalist
- MIT licence

But there remain many things to do...

- Governance
- Supporting documentation
- Engage other developers
- Integration with complementary systems
- Application data definition “libraries”

Discussion

Some issues raised by reviewers:

- Evolution of data and schema
- Concurrent access conflicts
- Access control granularity
- Usability, evaluation

Evolvability

Recall: data first philosophy

Two aspects of evolution:

- adding structure to data (add schema)
- changing existing structure (schema change)

Types vs properties:

- using Annalist, types primarily affect resource naming (entity names)
- properties affect content (JSON keys)

Data migration

Focus on type and property URI changes

Adopting a guided approach for now

```
$ annalist-manager migratecollection Performance_defs Journal_defs

# Migration report from collection 'Performance_defs' to 'Journal_defs' #

  * Type Uploaded_audio, URI changed from 'coll:Uploaded_audio' to
   'coll:Uploaded_audio_test’
     Consider adding supertype 'coll:Uploaded_audio' to
         type 'Uploaded_audio' in collection 'Journal_defs’
      URI 'coll:Uploaded_audio' appears as entity type for view 'Uploaded_audio’
      URI 'coll:Uploaded_audio' appears as entity type for list 'Uploaded_audio’
      URI 'coll:Uploaded_audio' appears in selector for list 'Uploaded_audio’
      URI 'coll:Uploaded_audio' appears as entity type for group Uploaded_audio_m
      URI 'coll:Uploaded_audio' appears as entity type for group Uploaded_audio_r
  * Field Linked_audio, property URI changed from 'coll:audio_clip' to 'coll:linked_audio’
      Consider adding property alias for 'coll:audio_clip' to type
          Linked_audio in collection 'Journal_defs'
  * Field Web_resource, property URI changed from 'coll:web_resource' to 'coll:resource'
      Consider adding property alias for 'coll:web_resource' to type
          Web_resource in collection 'Journal_defs’
```

Concurrent access conflicts

Atomic updates to single entity

Design to detect update conflicts:

- detect changes while an edit is in progress
- cf. HTTP entity tag (ETag)
- not currently implemented

No consistency checks between entities

- storage model doesn’t care about consistency
- consider as aspect of data quality checks
- handle post-acquisition, as needed

Access control granularity

Currently:

- control applied per-collection
- permissions associated with authenticated user Id in Annalist "user permissions" record
- limited possibility to require different permissions for different record types

Possibilities:

- type-based permission requirements could offer finer granularity (but not to individual statement level)
- Generalize Annalist trust/permission model for RBAC

Would like:

- To devise way to use OpenID Connect (OAuth2) authentication with WebID permissions; e.g. to work with SoLiD servers

Usability, evaluation

No formal usability study (yet)

- What to test?
- Different user applications are ... different
- How to formally test flexibility?

Evolving interface through experience

- Incremental development, informed by “agile”
- Using Annalist in diverse applications
- Modifying user interface in response to problems experienced

Links

Paper

- https://github.com/gklyne/annalist/blob/develop/documents/publications/LDOW2016-paper/Annalist-paper-ACMSIG.pdf

Slides

- https://github.com/gklyne/annalist/blob/develop/documents/publications/LDOW2016-paper/annalist-presentation-ldow2016.pptx, .pdf

Demo site

- http://demo.annalist.net/

Tutorial

- http://annalist.net/documents/tutorial/annalist-tutorial.html

Software

- https://github.com/gklyne/annalist/