Galaxy

Internals, organization, control flow

www.getgalaxy.org

James, Dan, Greg

The Plan

1. What's in the galaxy-central repository?
2. Galaxy web application architecture
3. Control flow in the Galaxy web application
4. Tools in the age of the toolshed
5. Galaxy Workflows
6. Galaxy data organization

0. The right ways to be involved with Galaxy

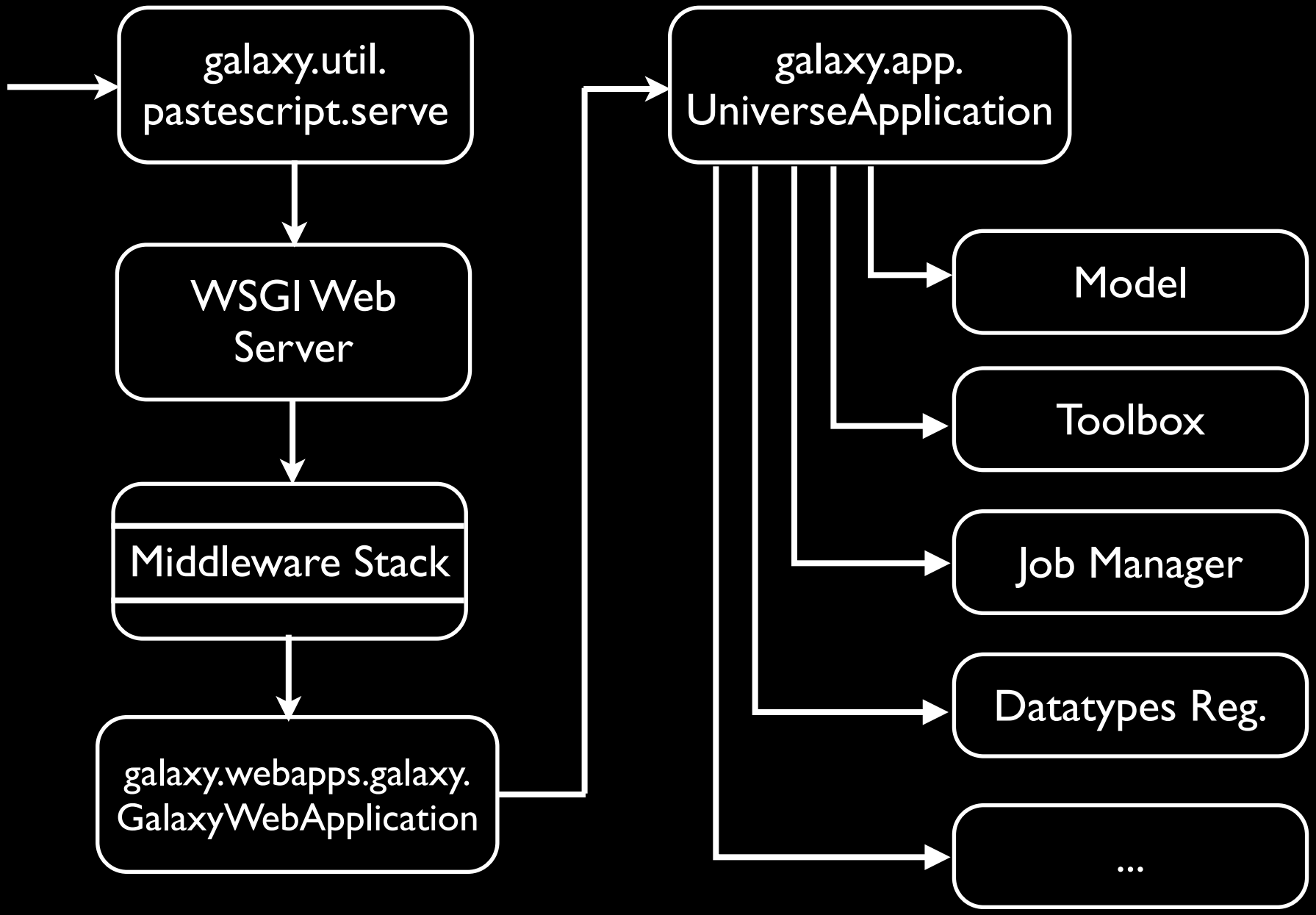
IRC: irc.freenode.net #galaxyproject

Trello: https://trello.com/b/75c1kASa/galaxy-development

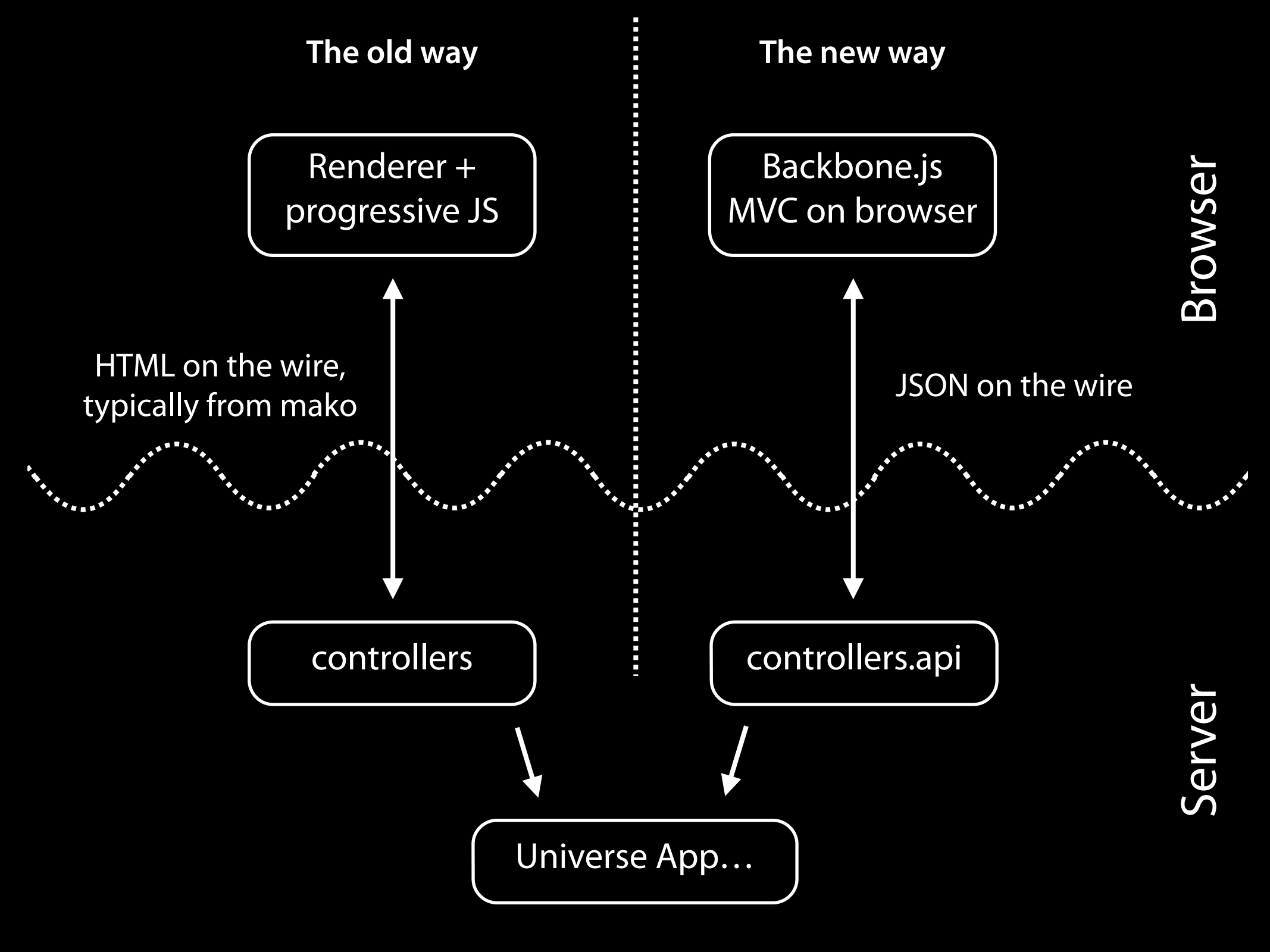
1. Getting around the Galaxy repository

(lets look at the code)

2. Galaxy application architecture


galaxy.util.
pastescript.serve
WSGI Web
Server
Middleware Stack
galaxy.webapps.galaxy.
GalaxyWebApplication
galaxy.app.
UniverseApplication
Model
Toolbox
Job Manager
Datatypes Reg.
...


<<thread>>
Job Manager
Job Handlers
Job Runner


The old way
The new way
Renderer + progressive JS
Backbone.js MVC on browser
Browser
HTML on the wire, typically from mako
JSON on the wire
controllers
controllers.api
Server
Universe App…

The old way

User stuff (prefs, etc)

Tool forms

Reports

Tool shed

**Many of these have an API but it is not yet used by the UI*

The new way

Visualizations

History

Tool menu

Most grids

In between

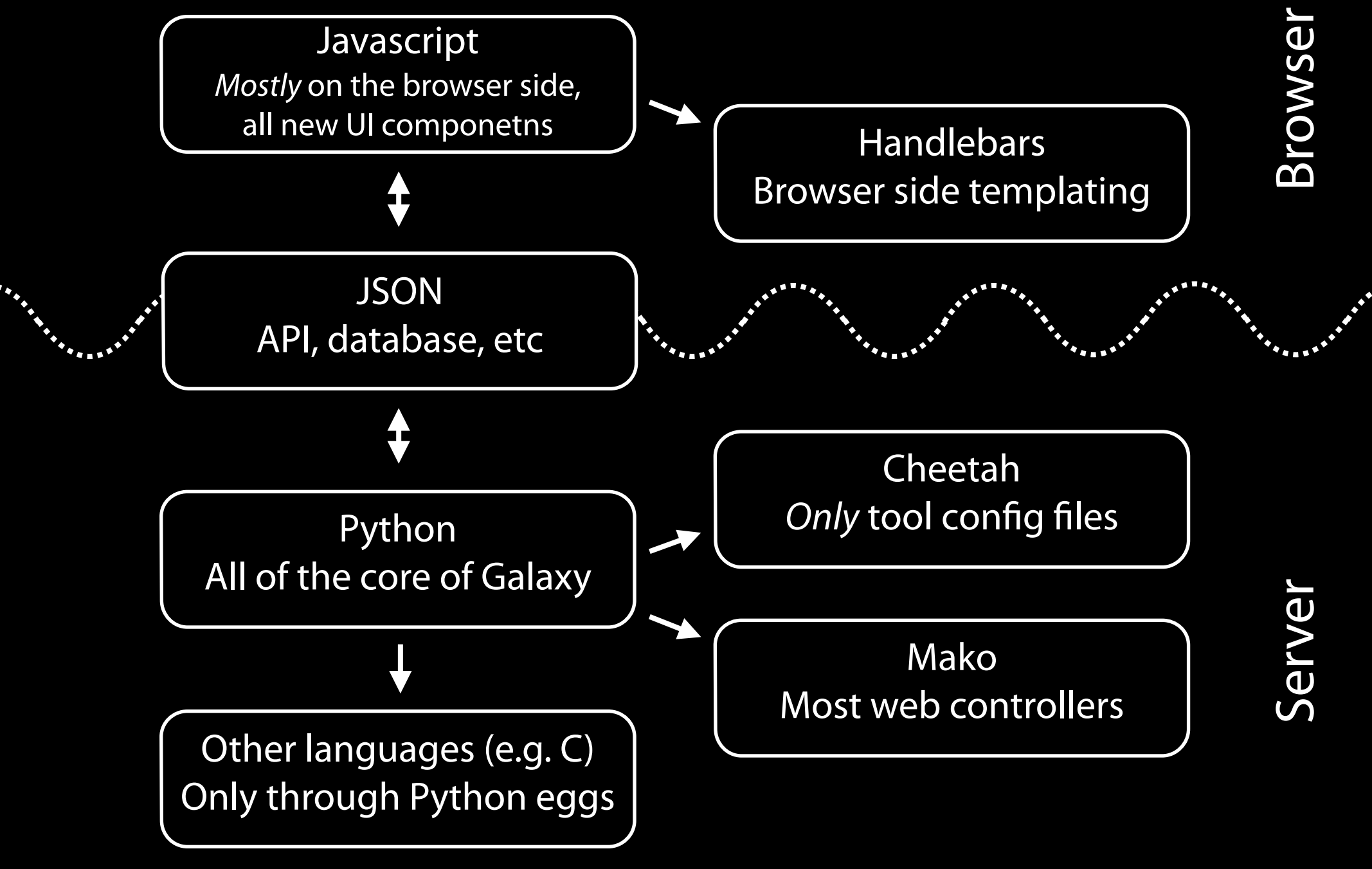
Workflows

Data Libraries

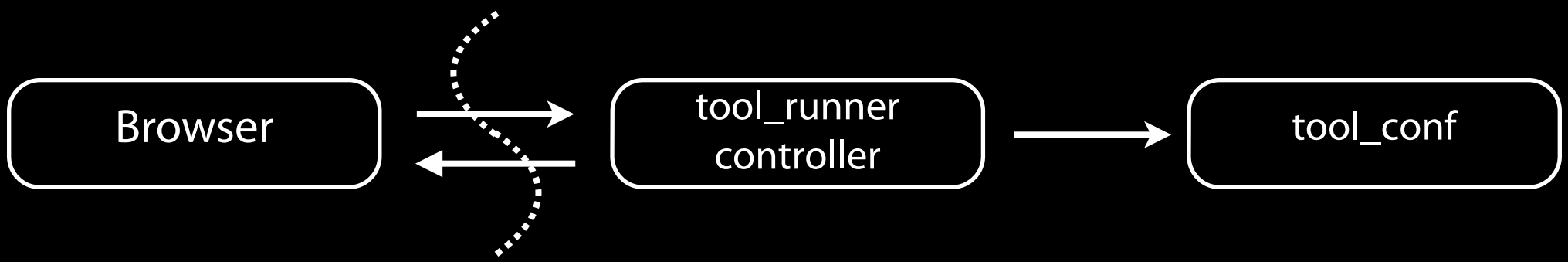
Example of the new way: Tool menu generation

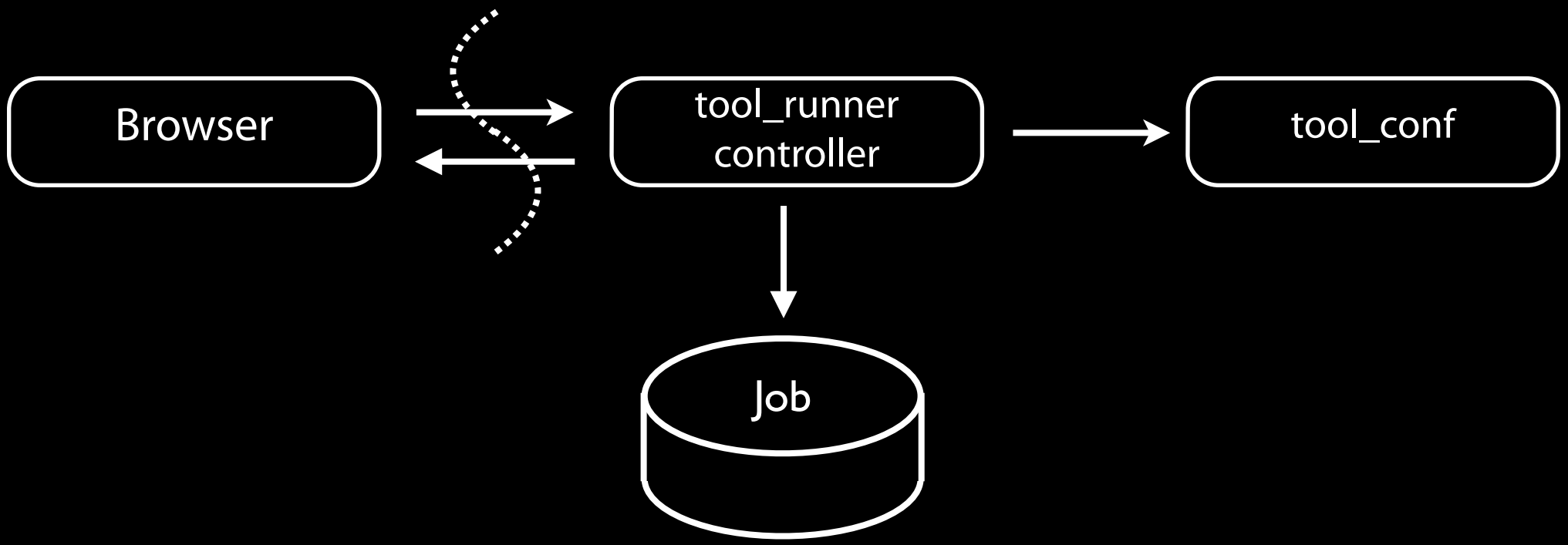
Example of the new way: Tool form generation

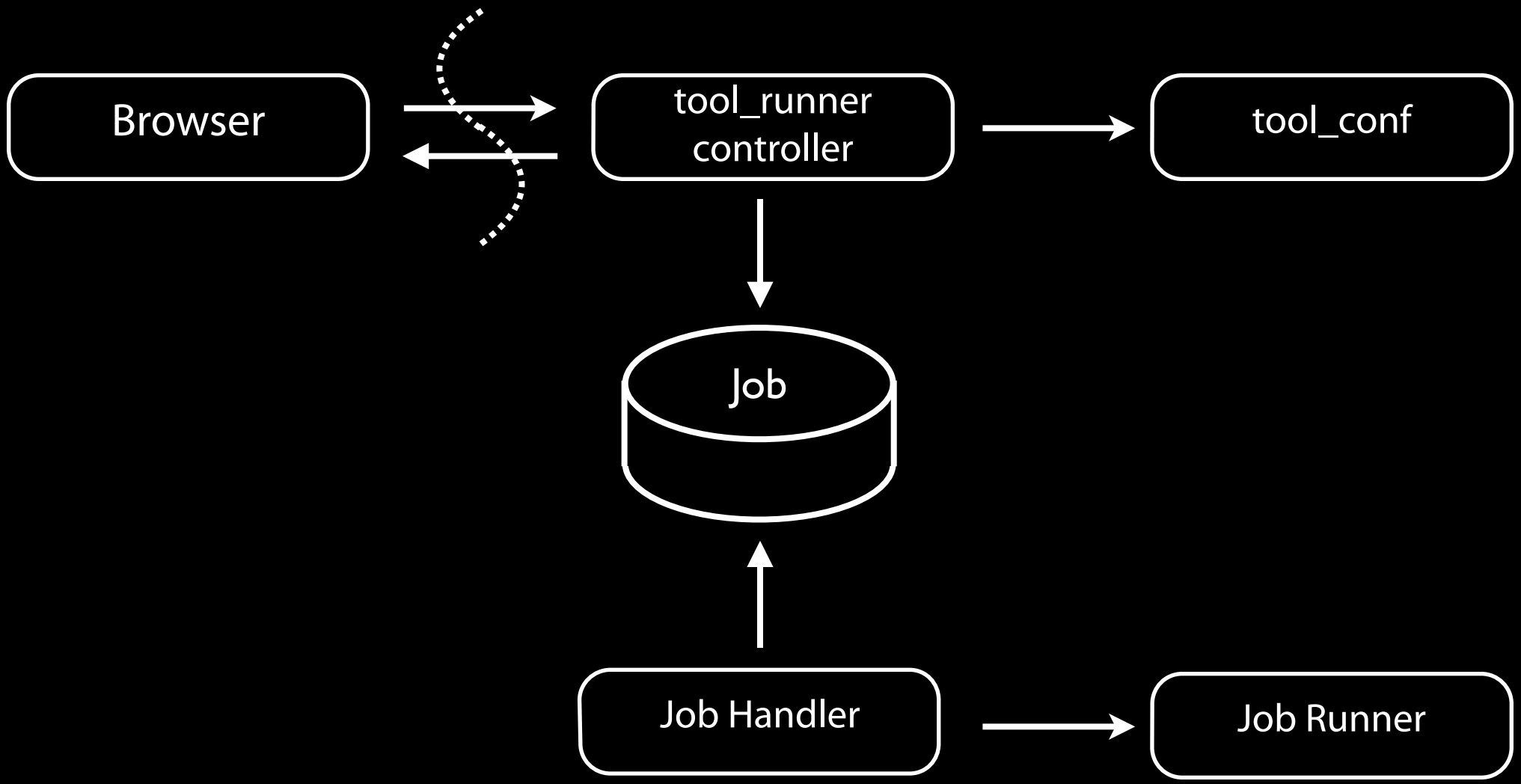
So many languages!



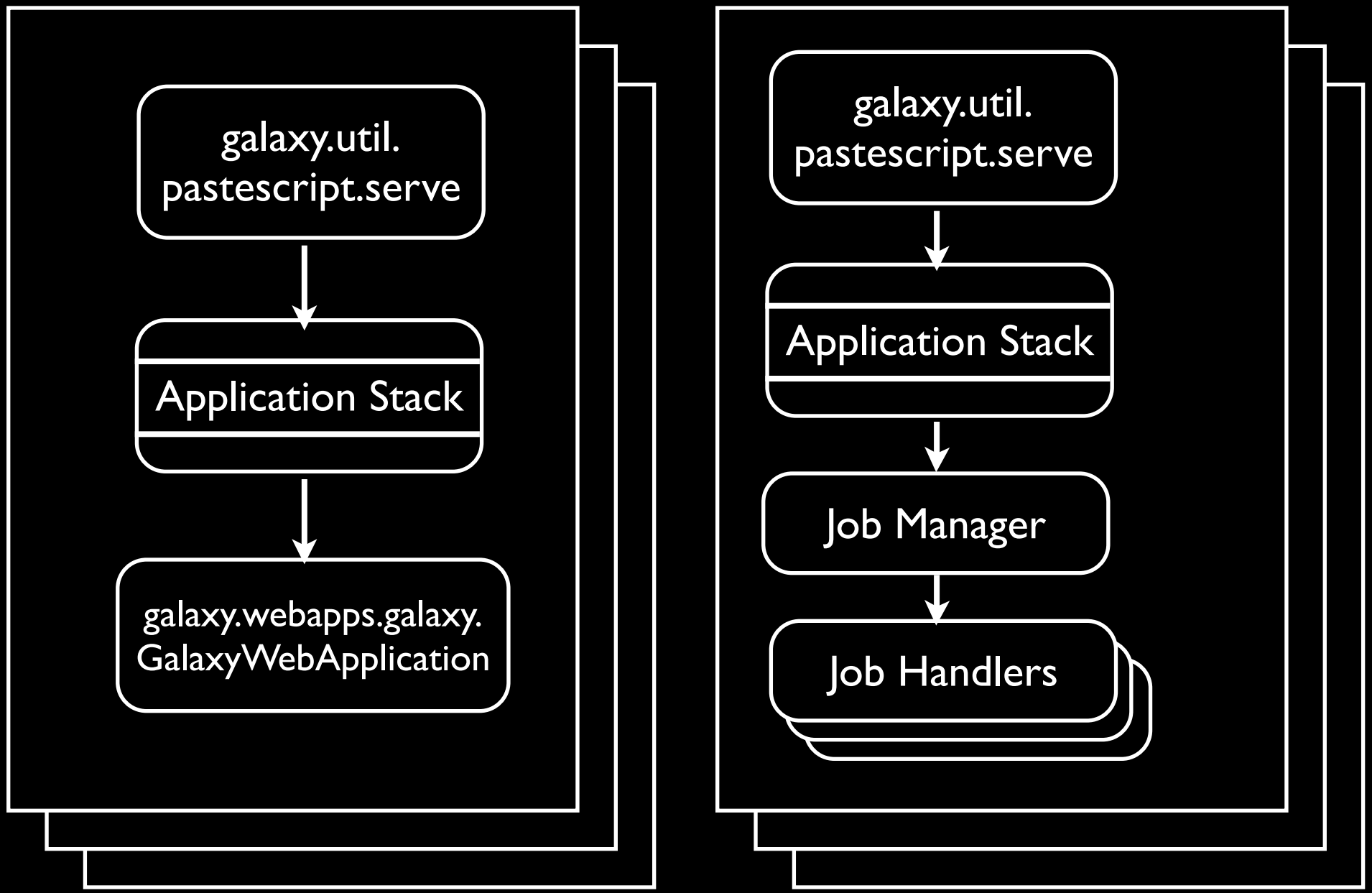
3: Control flow: running jobs


Browser
tool_runner
controller
tool_conf


Browser
tool_runner
controller
tool_conf
Job


Browser
tool_runner
controller
tool_conf
Job
Job Handler
Job Runner

Processes


galaxy.util.
pastescript.serve
Application Stack
galaxy.webapps.galaxy.
GalaxyWebApplication
galaxy.util.
pastescript.serve
Application Stack
Job Manager
Job Handlers

4. Tools in the age of the toolshed

The old way

1. Each tool specified by a tool.xml somewhere on the local filesystem (but typically under `tools`)

2. Tools to be loaded specified in tool_conf.xml, loaded by Galaxy at startup — *no representation in database beyond tool ids*

No way to access old tool configurations after updates

In the ToolShed

ToolShed Repository

stored as mercurial repo on disk in ToolShed
several types: unrestricted, suite, tool dependency
unrestricted can have multiple installable revisions

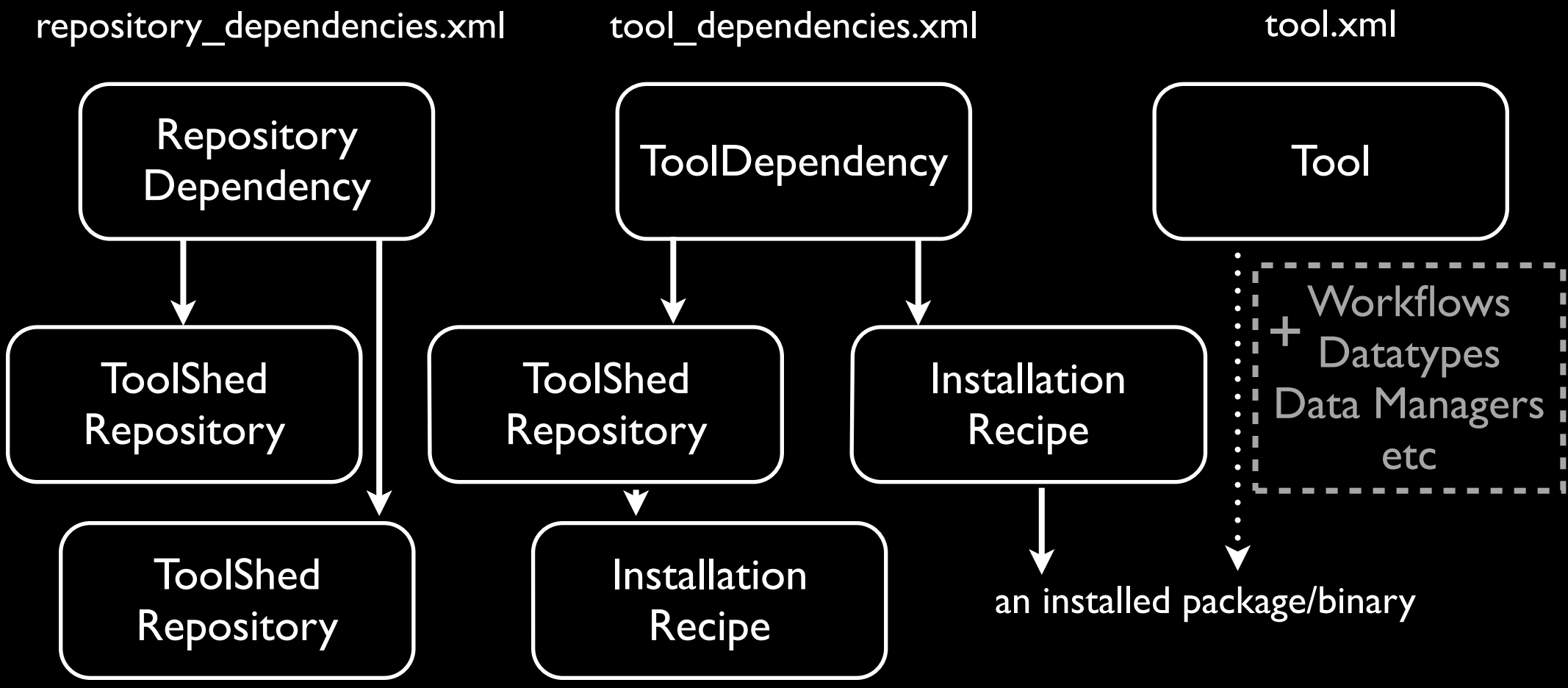
lib.galaxy.webapps.tool_shed / lib.tool_shed

In the ToolShed

ToolShed Repository

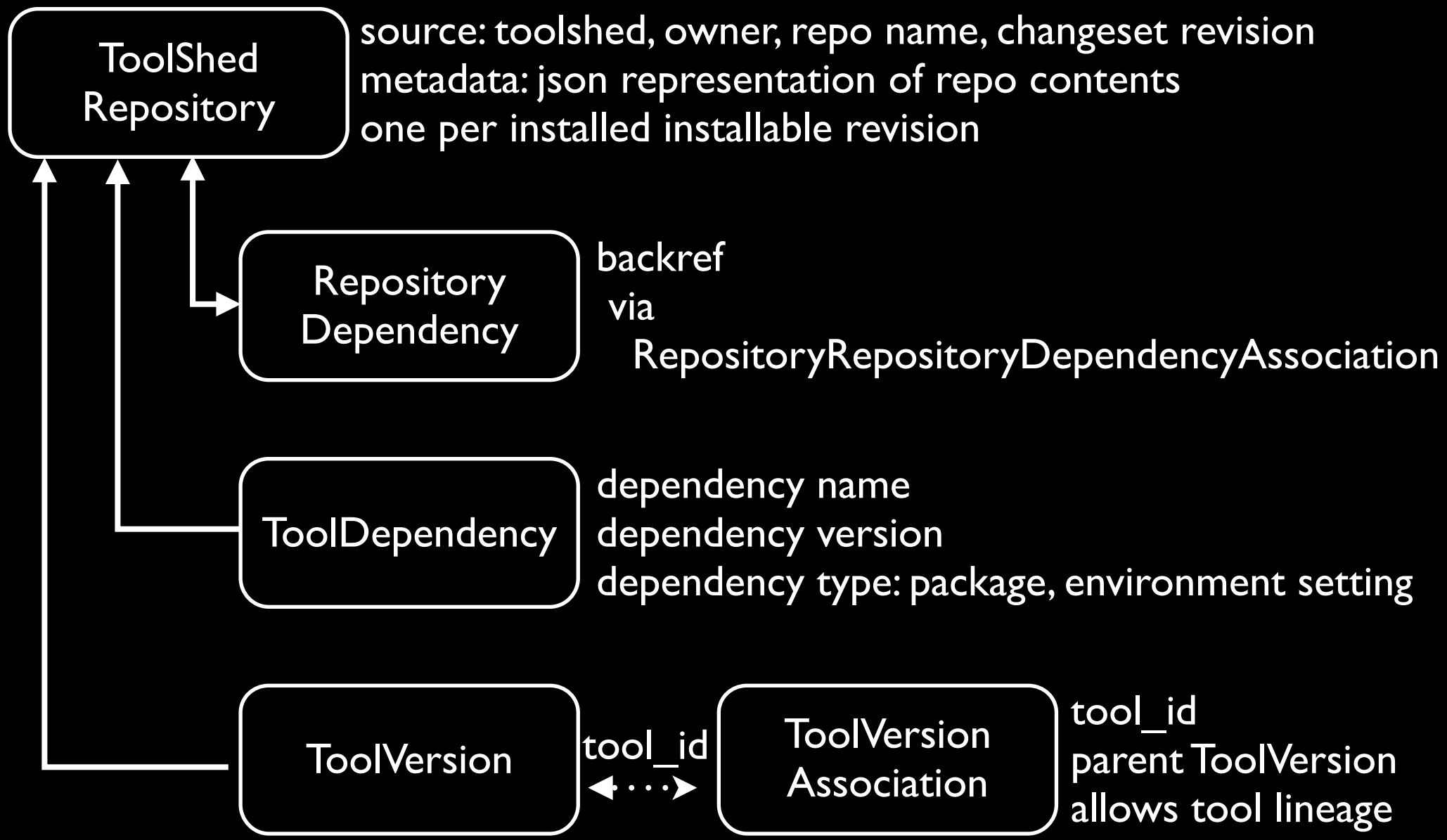
stored as mercurial repo on disk in ToolShed
several types: unrestricted, suite, tool dependency
unrestricted can have multiple installable revisions

Each installable revision can have



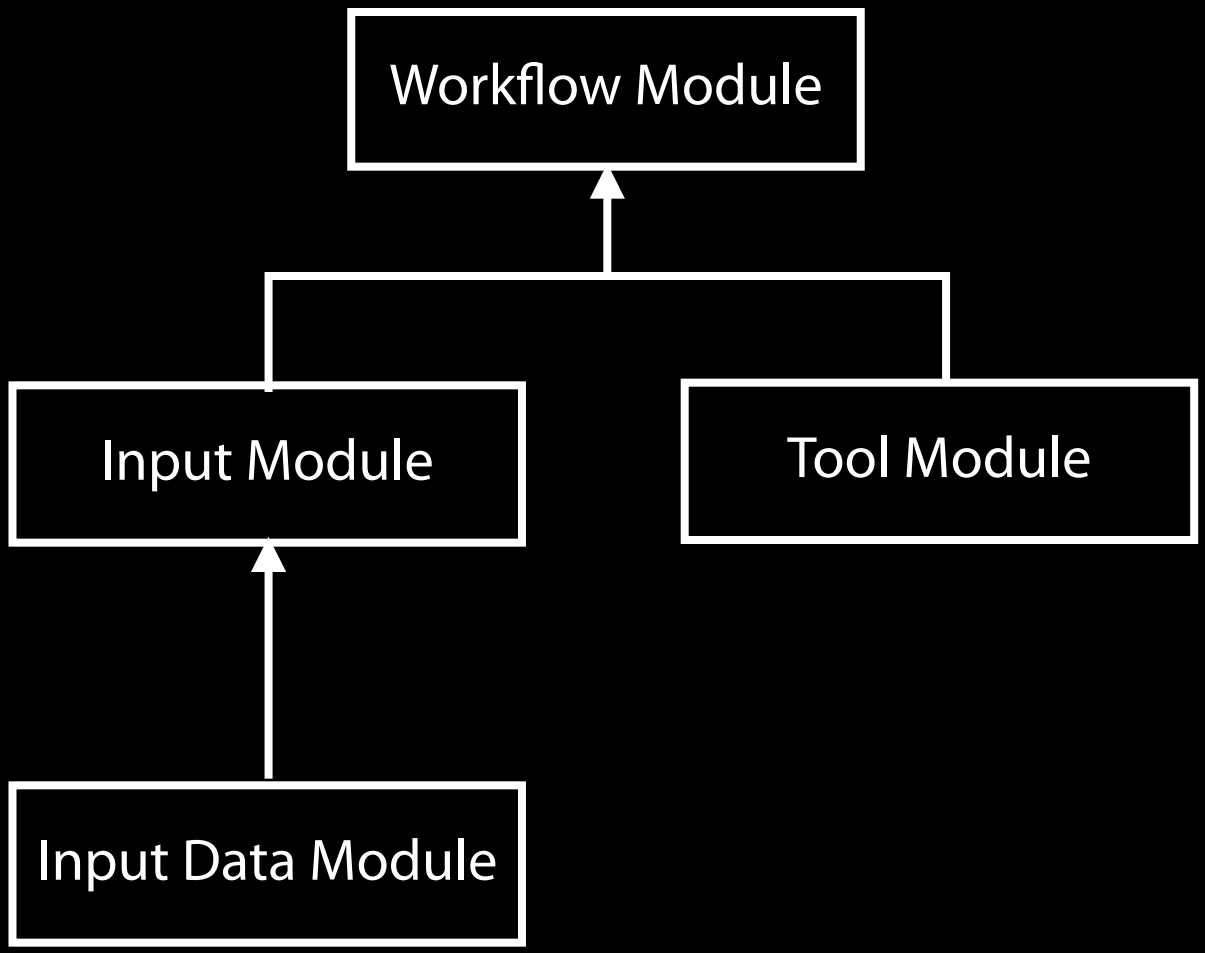
Installed in Galaxy

app.install_model



5. Galaxy workflows

lib/galaxy/workflow/modules.py



Workflow modules have:

Config time state — in the workflow editor used to generate the form associated with a given step and update it

Runtime state — similar but used for parameters set at workflow runtime

As well as conversion from JSON <-> Workflow Module instance <-> workflow_step encoded in database

Workflow scheduling:

*Currently w*orkflows are scheduled like any other job

All intermediate datasets and connections are created and each step is sent as a job to the JobManager

Pausing: when intermediate steps fail the workflow is paused. *Although, this actually applies to any dependent jobs*

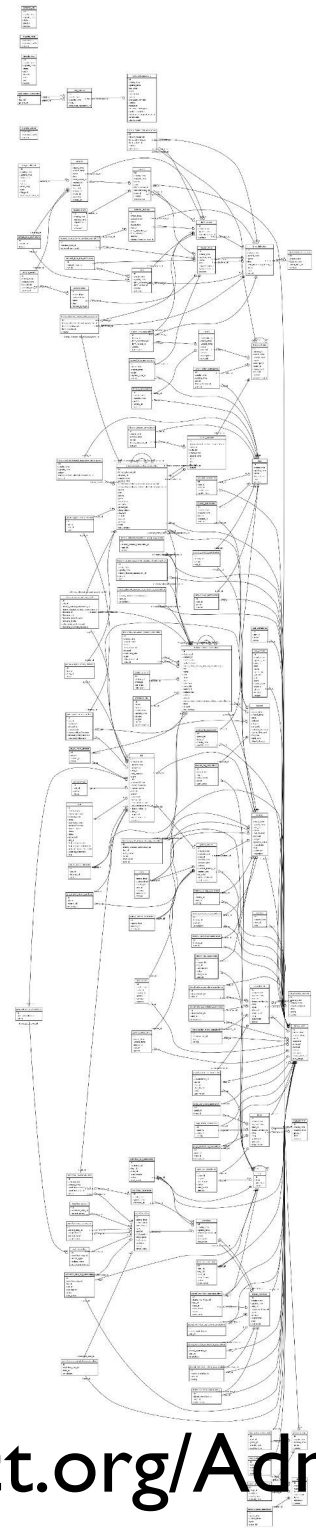
6. Galaxy data organization

Where does data in Galaxy go?

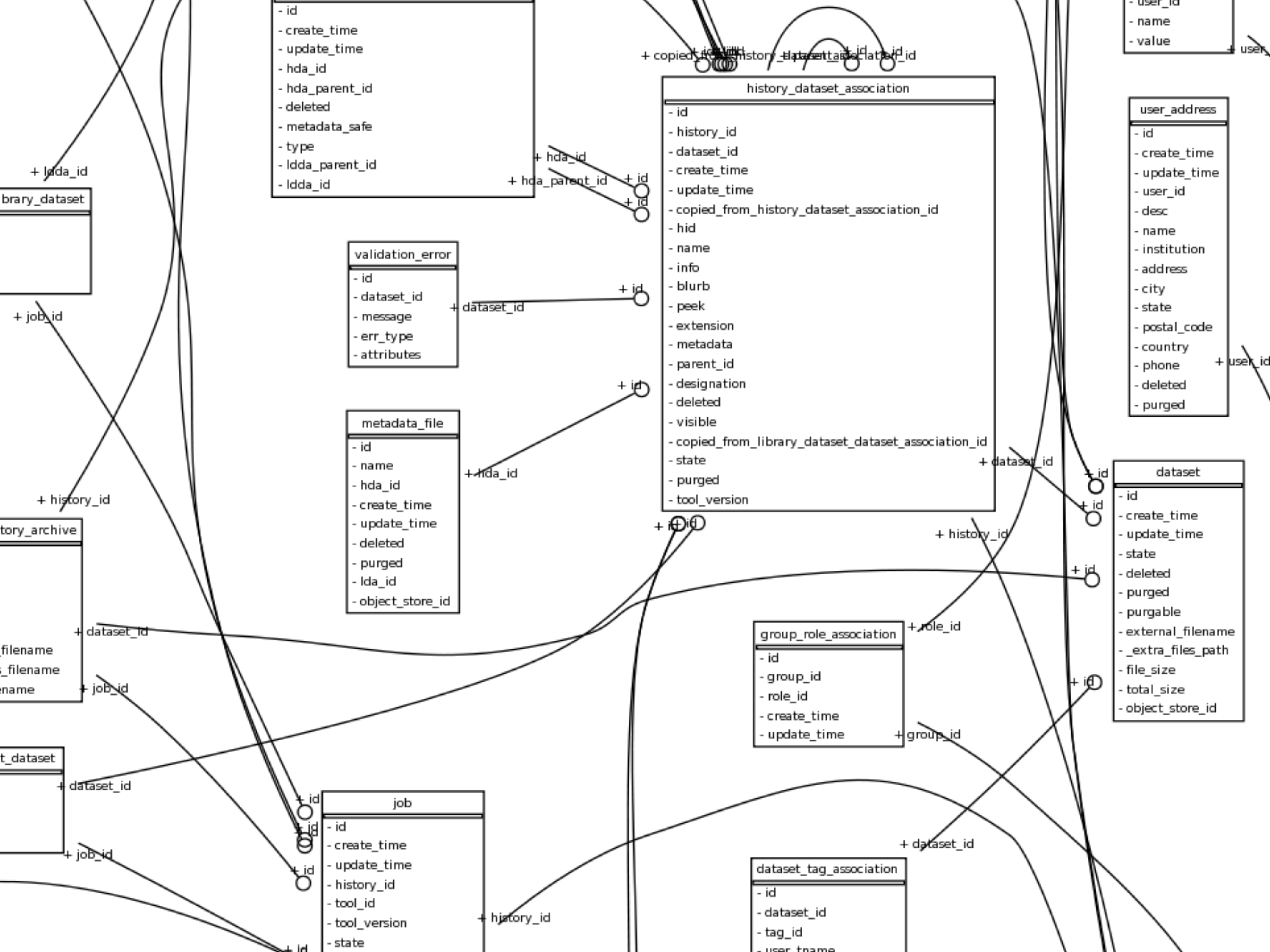
1. “Metadata” is stored in a SQL database (preferable Postgres): Users, workflows, histories, dataset metadata… everything a user creates interacting with Galaxy except the raw contents of datasets

2. Dataset contents is stored in file_path, typically database/files

3. Data used by tools that is not user specific is stored in



https://wiki.galaxyproject.org/Admin/Internals/DataModel


- id
- create_time
- update_time
- hda_id
- hda_parent_id
- deleted
- metadata_safe
- type
- ldda_parent_id
- ldda_id
+ ldda_id
brary_dataset
+ job_id
+ hda_id
+ hda_parent_id
+ id
+ id
validation_error
- id
- dataset_id
- message
- err_type
- attributes
+ dataset_id
+ id
metadata_file
- id
- name
- hda_id
- create_time
- update_time
- deleted
- purged
- lda_id
- object_store_id
+ hda_id
+ id
+ history_id
tory_archive
+ dataset_id
filename
_filename
ename
+ job_id
t_dataset
+ dataset_id
+ job_id
+ id
+ id
+ id
job
- id
- create_time
- update_time
- history_id
- tool_id
- tool_version
- state
+ history_id
+ copied_from_history_dataset_association_id
history_dataset_association
- id
- history_id
- dataset_id
- create_time
- update_time
- copied_from_history_dataset_association_id
- hid
- name
- info
- blurb
- peek
- extension
- metadata
- parent_id
- designation
- deleted
- visible
- copied_from_library_dataset_dataset_association_id
- state
- purged
- tool_version
+ id
+ history_id
+ dataset_id
group_role_association
- id
- group_id
- role_id
- create_time
- update_time
+ role_id
+ group_id
+ dataset_id
dataset_tag_association
- id
- dataset_id
- tag_id
- user_tname
- user_id
- name
- value
+ user_
user_address
- id
- create_time
- update_time
- user_id
- desc
- name
- institution
- address
- city
- state
- postal_code
- country
- phone
- deleted
- purged
+ user_id
+ id
+ id
dataset
- id
- create_time
- update_time
- state
- deleted
- purged
- purgable
- external_filename
- _extra_files_path
- file_size
- total_size
- object_store_id
+ id
+ id

Galaxy data model is not database entity driven

Entities are defined in galaxy.model as objects

SQLAlchemy is used for object relation mapping

Mappings are defined in galaxy.model.mapping in two parts — a table definition and a mapping between objects and tables including relationships

Migrations allow the schema to be migrated forward automatically

It *rarely* makes sense to access the Galaxy database directly

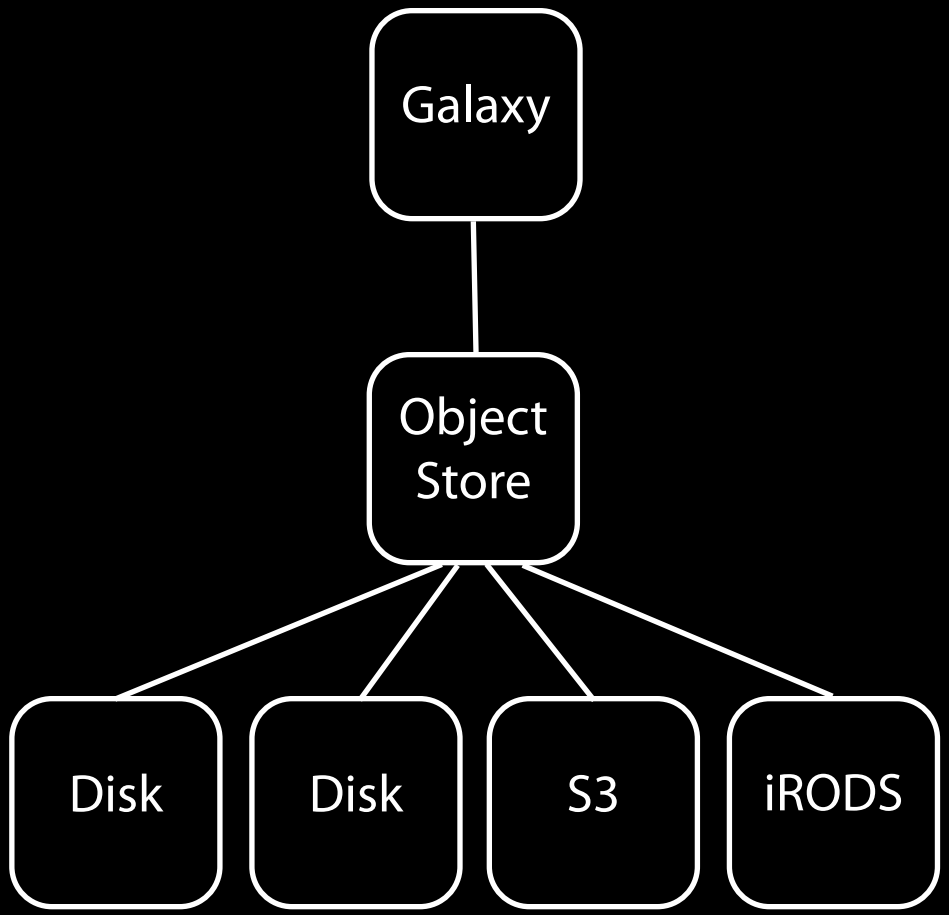
Where does data in Galaxy go?

1. “Metadata” is stored in a SQL database (preferable Postgres): Users, workflows, histories, dataset metadata… everything a user creates interacting with Galaxy except the raw contents of datasets

2. Dataset contents is stored in ~~file_path, typically database/files~~ objectstore

3. Data used by tools that is not user specific is stored in

Data Abstraction



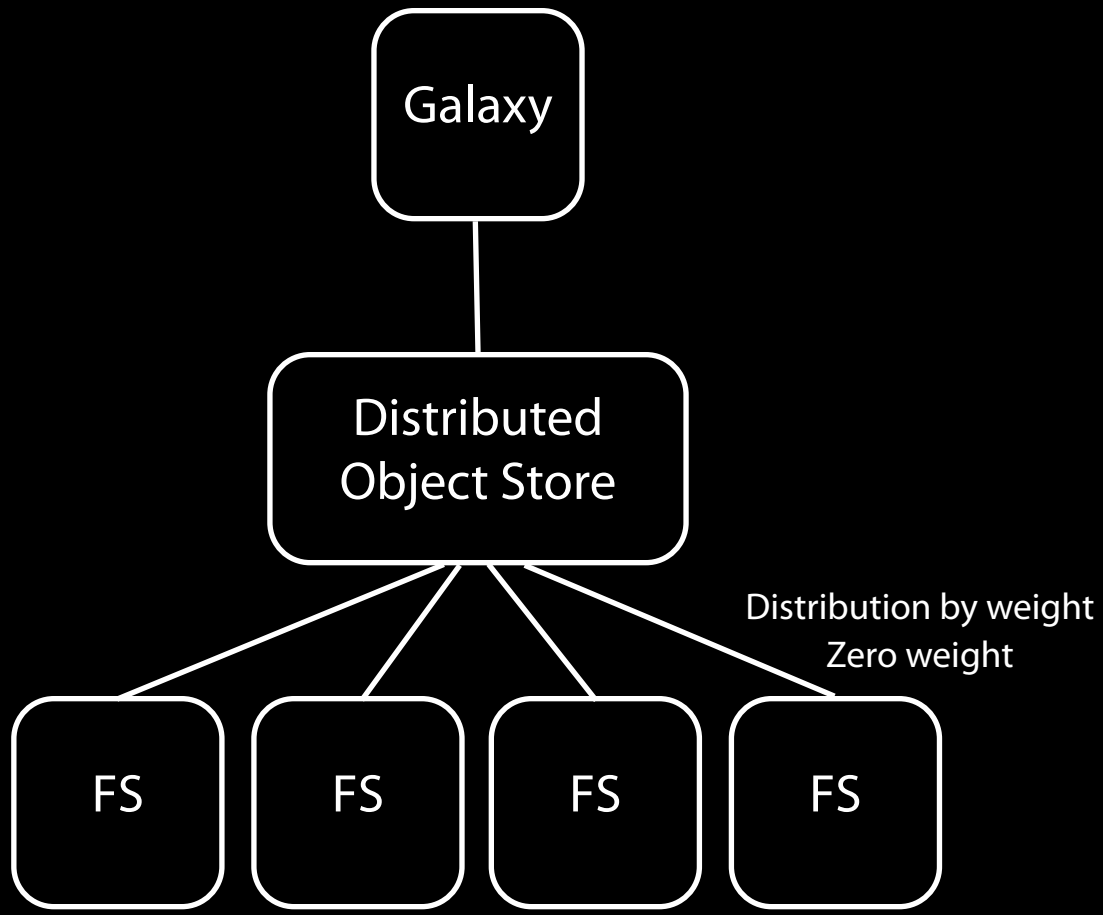
Data Abstraction

```
>>> fh = open( dataset.file_path, 'w' )
>>> fh.write( 'foo' )
>>> fh.close()
>>> fh = open( dataset.file_path, 'r' )
>>> fh.read()
```

```
>>> update_from_file( dataset, file_name='foo.txt' )
>>> get_data( dataset )
>>> get_data( dataset, start=42, count=4096 )
```

Data Abstraction

Distributed Object Store



Data Abstraction

Benefits

- Grow beyond original capacity
- Avoid migrating data offline
- Tier storage
- Let your users bring their own storage
- Use resources w/o a shared filesystem (with iRODS)
- Remove IO bottlenecks

Data tables and location files

Data tables provide an abstraction which tools use to access indexes of data which can be accessed on the local filesystem

Tool config reference data table by name with abstract columns

```
<param name="mafType" type="select" label="Choose alignments">
  <options from_data_table='indexed_maf_files'>
    <filter type="data_meta" ref="input1" key="dbkey" column="dbkey"
    <validator type="no_options" message="No alignments are available
  </options>
</param>
```

Data tables configuration maps abstract data table to a concrete file

```
<!-- Locations of MAF files that have been indexed with bx-python -->
<table name="indexed_maf_files">
    <columns>name, value, dbkey, species</columns>
    <file path="tool-data/maf_index.loc" />
</table>
```

Which can provide any information, but typically locations of data at a given site

```
#This is a sample file distributed with Galaxy that is used by some
#alignment tools.  The maf_index.loc file has this format (white space
#characters are TAB characters):
#
#<Display_name UID> <indexed_for:build1,build2,build3>  <exists_in_maf:build1,build2,build3>    <Comma_Se
#
ENCODE TBA (hg17)   ENCODE_TBA_hg17 armadillo,baboon,galGal2,panTro1,colobus_monkey,cow,canFam1,dusky_tit
ENCODE MAVID (hg17) ENCODE_MAVID_hg17   armadillo,baboon,galGal2,panTro1,colobus_monkey,cow,canFam1,dusky
ENCODE TBA (hg16)   ENCODE_TBA_hg16 armadillo,baboon,galGal2,panTro1,colobus_monkey,cow,canFam1,dusky_tit
8-way multiZ (hg17) 8_WAY_MULTIZ_hg17   canFam1,danRer1,fr1,galGal2,hg17,mm5,panTro1,rn3    canFam1,danRe
17-way multiZ (hg18)    17_WAY_MULTIZ_hg18  hg18,panTro1,bosTau2,rheMac2,mm8,rn4,canFam2,echTel1,loxAfr1,
3-way multiZ (hg18,panTro2,rheMac2) 3_WAY_MULTIZ_hg18   hg18,panTro2,rheMac2    hg18,panTro2,rheMac2    A
5-way multiZ (hg18,panTro2,rheMac2,mm8,canFam2) 5_WAY_MULTIZ_hg18   hg18,panTro2,rheMac2,mm8,canFam2    h
28-way multiZ (hg18)    28_WAY_MULTIZ_hg18  hg18    hg18,dasNov1,otoGar1,felCat3,galGal3,panTro2,bosTau3,
15-way multiZ (dm2) 15_WAY_MULTIZ_dm2   dm2,droSim1,droSec1,droYak2,droEre2,droAna3,dp4,droPer1,droWil1,d
17-way multiZ (mm8) 17_WAY_MULTIZ_mm8   mm8 hg18,panTro1,bosTau2,rheMac2,mm8,rn4,canFam2,echTel1,loxAfr1,
8-way multiZ (ponAbe2)  8_WAY_MULTIZ_ponAbe2    ponAbe2,hg18,panTro2,rheMac2,calJac1,mm9,monDom4,ornAna1
```

Data Managers

Special class of Galaxy tool which allows for the download and/or creation of data that is stored within Data Tables and their location files.

These tools handle e.g. the creation of indexes and the addition of entries/lines to the data table / .loc file via the Galaxy admin interface.

Data Managers can be defined locally or installed through the Tool Shed.

Available in: Admin GUI, Workflows, API

Special class of Galaxy tool

```
<tool id="data_manager_fetch_genome_all_fasta" name="Reference Genome" version="0.0.1" tool_type="manage_data">
```

```
<outputs>
    <data name="out_file" format="data_manager_json"/>
</outputs>
```

Writes a JSON description of new data table entries as content of tool output file

```
{
   "data_tables":{
      "all_fasta":[
         {
            "path":"sacCer2.fa",
            "dbkey":"sacCer2",
            "name":"S. cerevisiae June 2008 (SGD/sacCer2) (sacCer2)",
            "value":"sacCer2"
         }
      ]
   }
}
```

This creates a new entry in the Tool Data Table:

```
#<unique_build_id>      <dbkey>         <display_name>  <file_path>
sacCer2 sacCer2 S. cerevisiae June 2008 (SGD/sacCer2) (sacCer2) /Users/dan/galaxy-central/tool-data/sacCer2/seq/sacCer2.fa
```

Where the sacCer2.fa file was placed by the tool in the output file's extra_files_path

data_manager entry inside <data_managers> tag in data_mananger_conf.xml

```
<data_manager tool_file="data_manager/bwa_index_builder.xml" id="bwa_index_builder" version="0.0.1">
    <data_table name="bwa_indexes">
        <output>
            <column name="value" />
            <column name="dbkey" />
            <column name="name" />
            <column name="path" output_ref="out_file" >
                <move type="directory" relativize_symlinks="True">

                    <target base="${GALAXY_DATA_MANAGER_DATA_PATH}">${dbkey}/bwa_index/${value}</target>
                </move>
                <value_translation>${GALAXY_DATA_MANAGER_DATA_PATH}/${dbkey}/bwa_index/${value}/${path}</value_translation>
                <value_translation type="function">abspath</value_translation>
            </column>
        </output>
    </data_table>
</data_manager>
```

informs Galaxy about

- which data tables to expect for new entries
- special handling of provided JSON values and files

Q&A