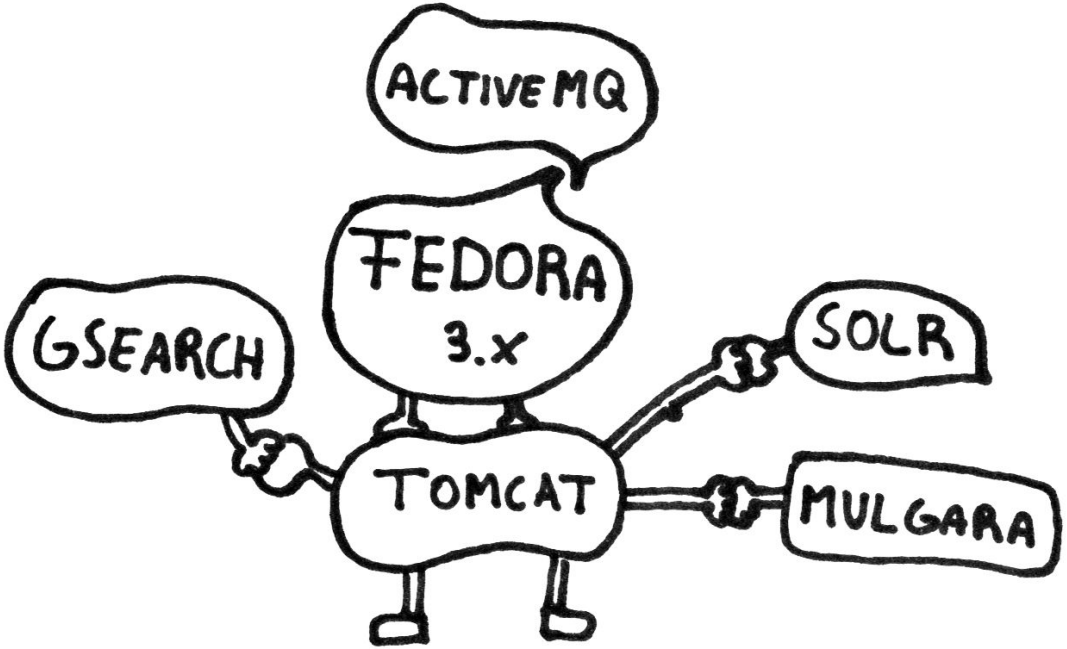
Lesson: Intro to Fedora 4.x

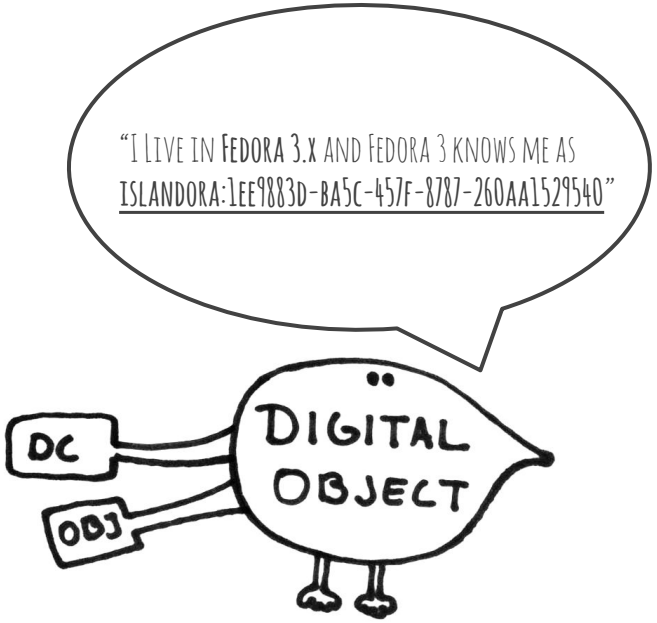
March 1, 2016

(My name is Diego Pino Navarro)

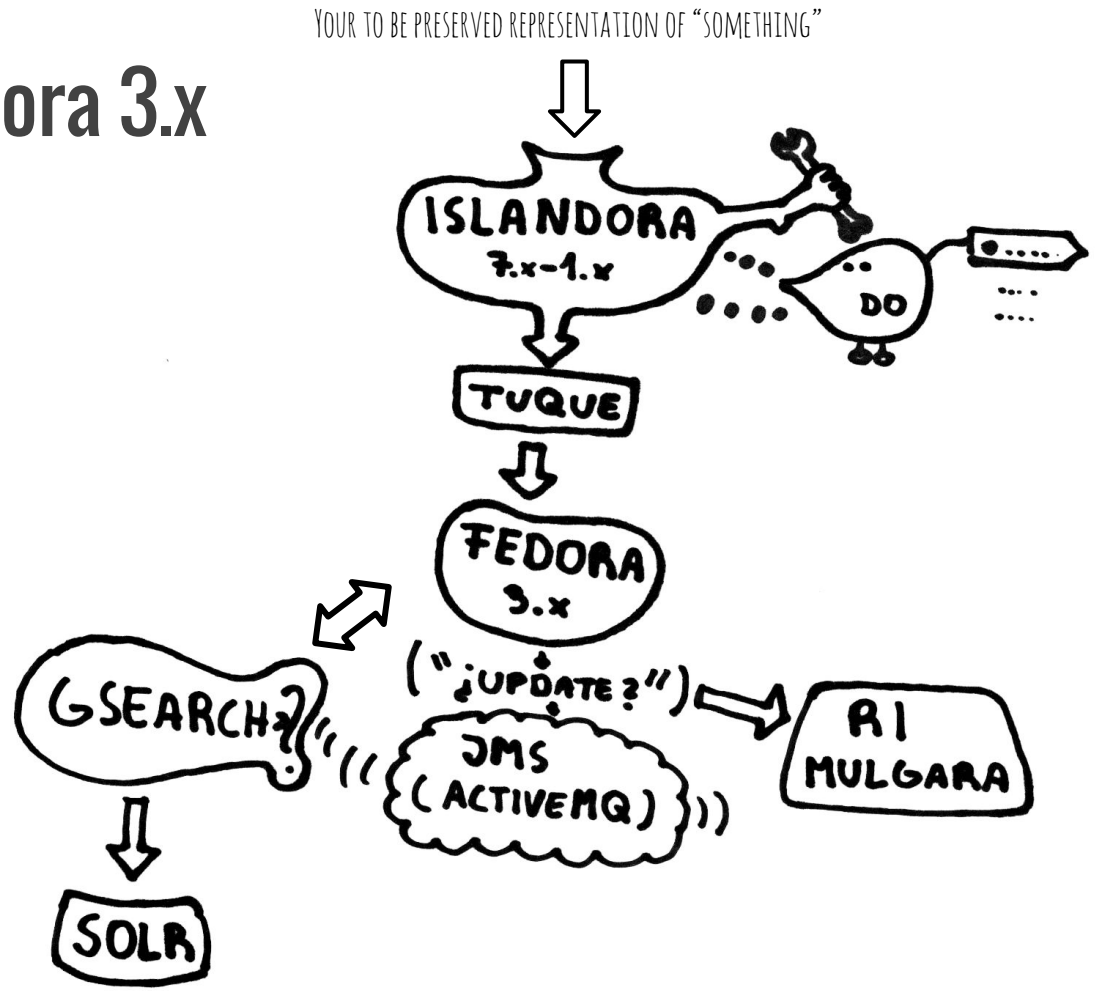
Let's start with what we know

The world we live on: Fedora 3.x

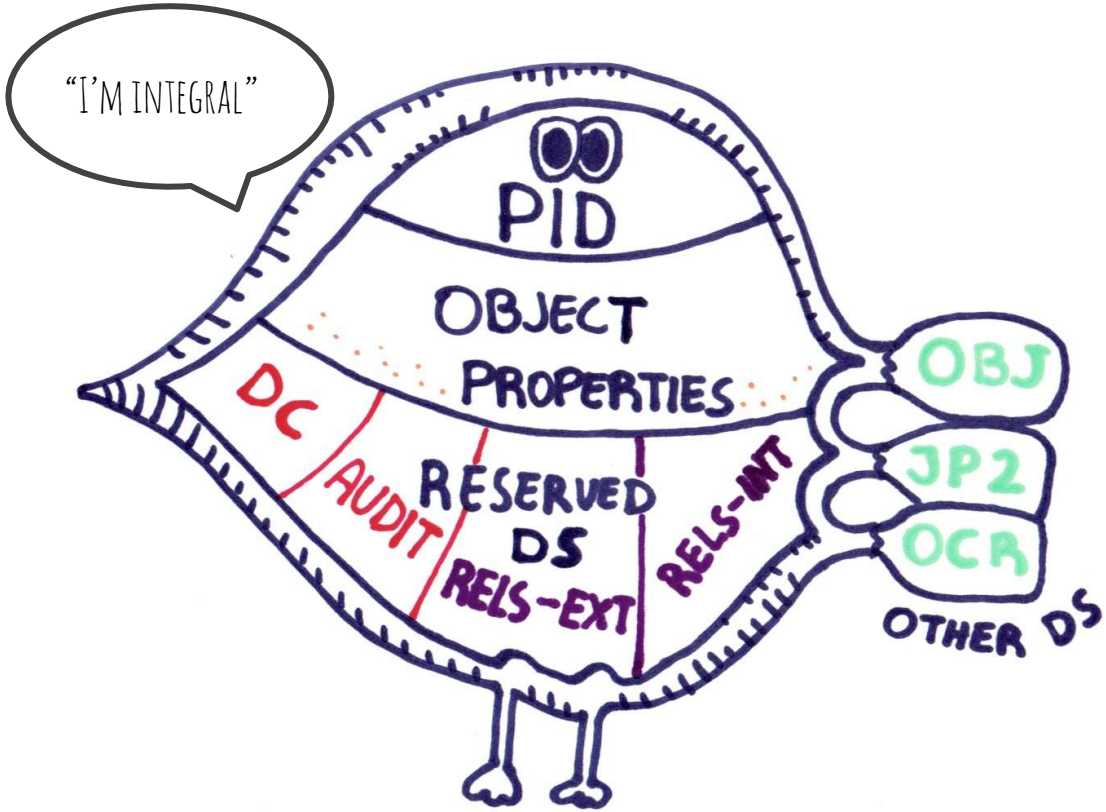




Ingesting in Fedora 3.x



Visiting an old friend: The Fedora DO

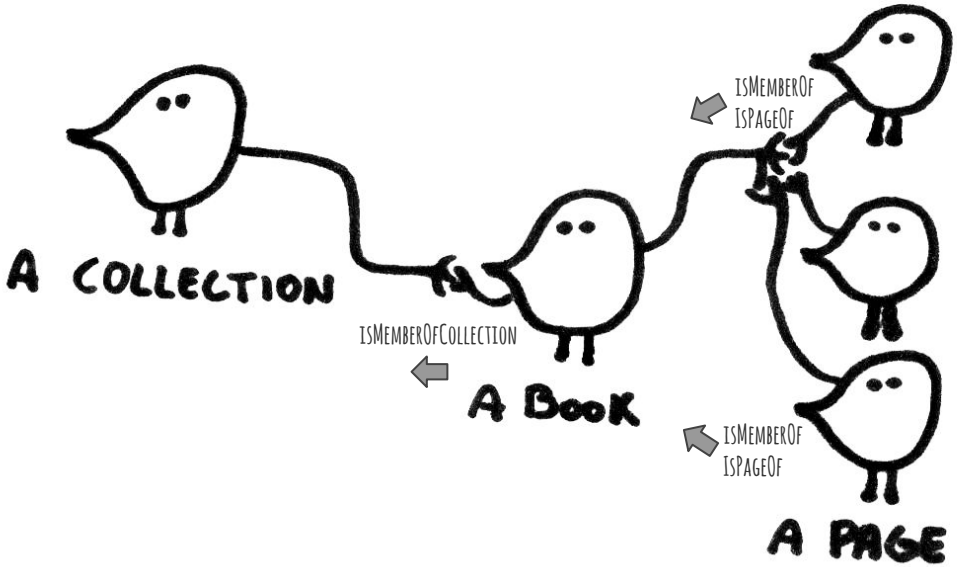


Some Facts:

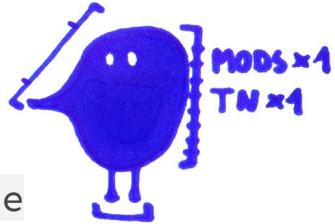
- PID is local
- Mostly XML (FOXML)
- Data streams act as "slots" for extra data
- Relates to others via RELS-EXT (RDF)
- How DS relate internally is defined via RELS-INT
- It's an instance of a class (well, mostly)

It's basically an encapsulated aggregation of DS

Among friends
(with a little help of Mulgara)



More Facts:



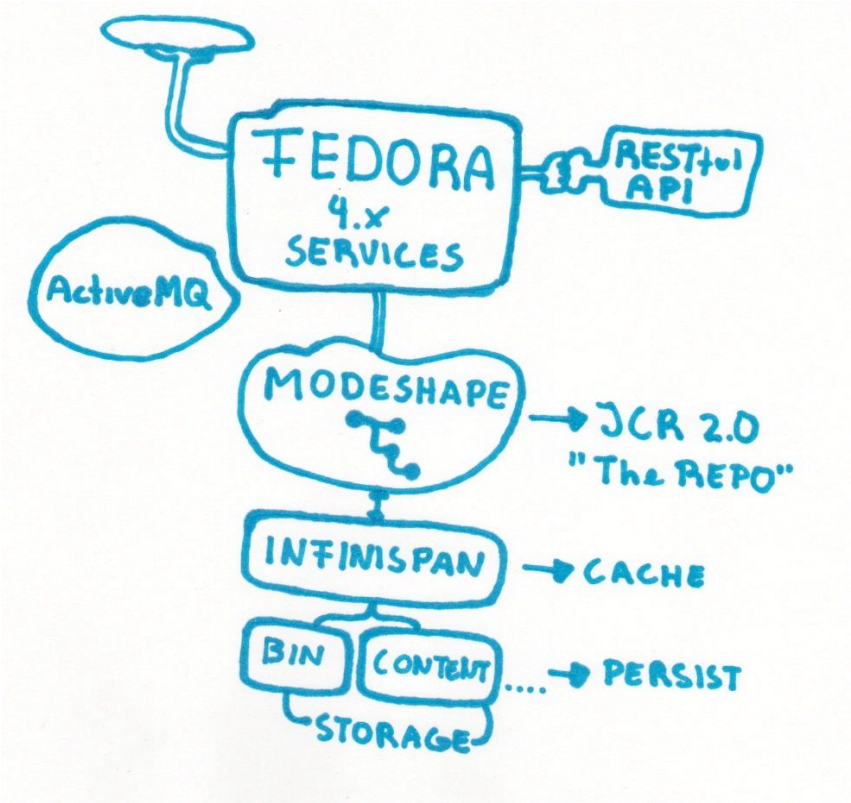
- Objects can be “instances” of one or more Classes
- We name this classes CMODELs, they act like a blueprint for DO
- CMODELs are also Objects

Do you see something strange?

- In OOP classes are not Objects
- Their “given meaning” is local to our Fedora Context

Now we can talk about Fedora 4.x

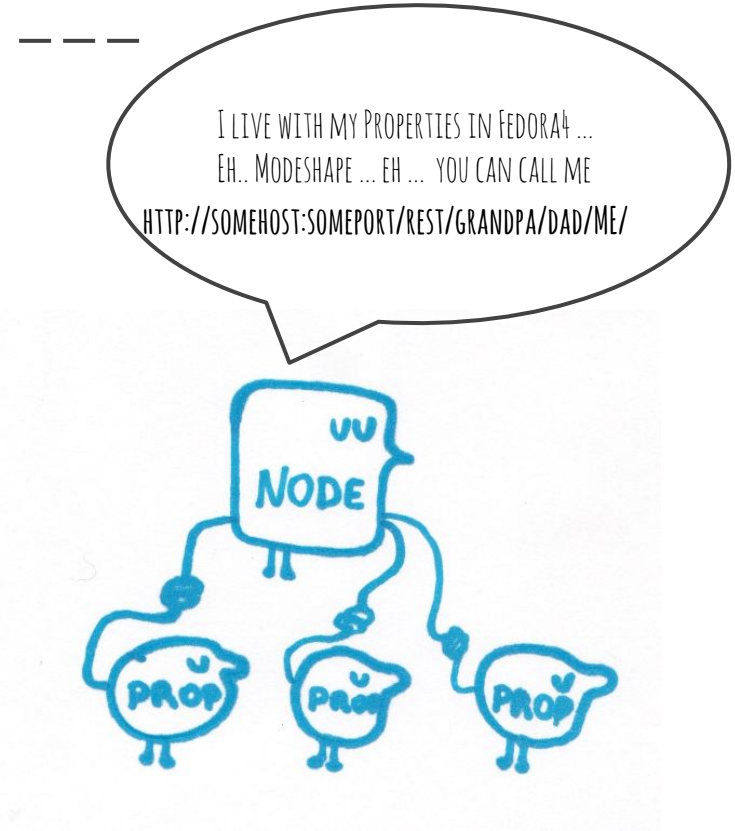
New World: Fedora 4.x



Architecture:

- Fedora 4 is not monolithic: it is really a pluggable API
- Each “layer” fulfills a different need and can be configured for different scenarios
- API is consistent and hides at, certain degree, lower layers complexity
- Much is Event Driven which allows for ASYNC workflows

Understanding data in Fedora 4.x: Modeshape



Modeshape:in-memory hierarchical database

It’s fast

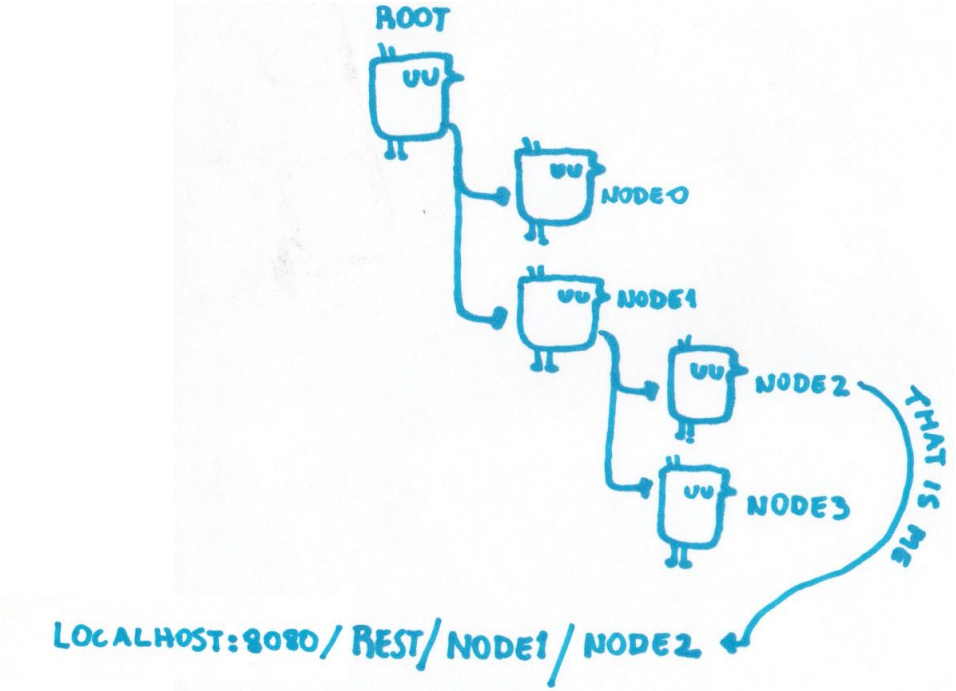
It’s safe

Implements JCR (Repo Services)

Data is handled as nodes in trees

Nodes can have 0-N properties

Understanding data in Fedora 4.x: Modeshape



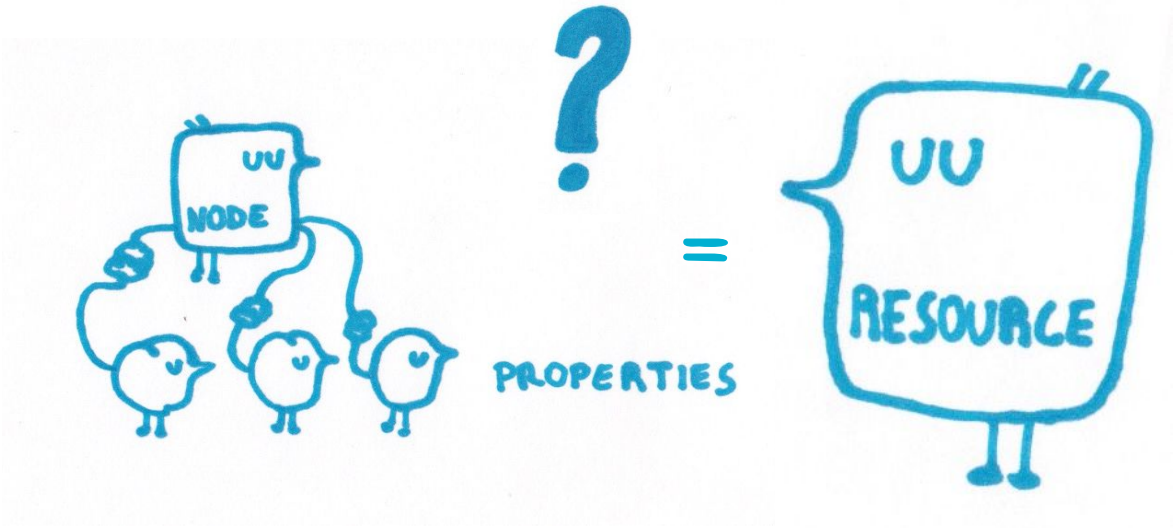
Some Facts:

- Each **Node** is identified by a **PATH**
- Nodes have **properties**
- Relations between nodes are directed
- The tree is acyclic: these relations don't form loops
- A tree is a graph

= Acyclic Directed Graph

Remember: this is "storage", but also more!

Understanding data in Fedora 4.x: Resources



A Resource is our "Data":

a **Node +**

```
---RDF----
 +  REPO Properties
 +  User Properties
---BINARY---
 +  A file?
```

identified by a **PATH**

yes same as the NODE

¿Why are you trying to confuse me?

(I don’t care about how data is managed inside)

We can’t escape this notion
It pops-up! (for good reasons)

Trees everywhere: the RESOURCE PATH

http://somehost:someport/rest/grandpa/dad/me/

Fact: our data is kept in a **tree**

PATHS describe hierarchies.

- Resources use **PATHS** as **identifiers**
- **PATHS** denote also an access location (URI) when using our Fedora 4 REST endpoint

So: "*how we **store** our data pops-up to how we **reference** and **access** our data.*"

RDF and why a PATH is a good idea

RDF and the semantic Web: quick and dirty

RDF = Resource Description Framework

Perfect for:

- Describing a Resource (a digital representation of a thing)
  - Typing (My resource is a “book”)
  - <subject_URI> <predicate> “some value” = triple
- Relations among resources
  - <subject_URI> <predicate> <object_URI> = triple
- Moving around/sharing Resources (Standard, can be serialized to multiple representations: JSON-LD, N-Triples, Turtle, RDF/XML)

HEY, MMM.. HTTP://SOMEHOST:SOMEPORT/REST/GRANDPA/DAD/ME/ IS AN URI!

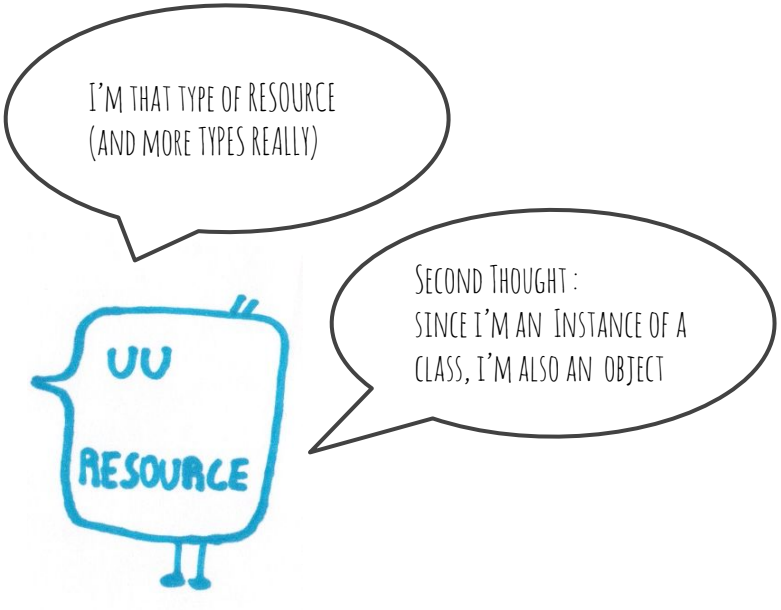
RDF and the semantic Web: MORE AWESOMENESS

The meanings we give to our Resources are not longer local

Ontologies:

A formal way to describe types (classes), properties and their relations in a certain knowledge domain.

So the concept of a “what a book is” can be formally defined, shared and understood by others (humans and machines).



<http://somehost:someport/rest/grandpa/dad/me> RDF:TYPE <http://fedora.info/definitions/v4/repository#RESOURCE>

Or simply

</rest/grandpa/dad/me> A FEDORA:RESOURCE

Some real RDF (text/turtle format serialization)

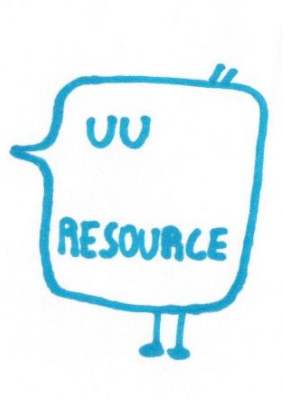
```
...
@prefix schema: <http://schema.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix fedora: <http://fedora.info/definitions/v4/repository#> .
@prefix nfo: <http://www.semanticdesktop.org/ontologies/2007/03/22/nfo/v1.1/> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix xs: <http://www.w3.org/2001/XMLSchema> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://localhost:8080/rest/grandpa/dad/me> a ldp:RDFSource , ldp:Container , schema:book , fedora:Container
, fedora:Resource ;
       dc:title "Surviving a F4 Migration"^^<http://www.w3.org/2001/XMLSchema#string> ;
       fedora:lastModifiedBy "bypassAdmin"^^<http://www.w3.org/2001/XMLSchema#string> ;
       fedora:createdBy "bypassAdmin"^^<http://www.w3.org/2001/XMLSchema#string> ;
       fedora:created "2016-02-29T23:51:53.75Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
       schema:author "Pino, Diego"^^<http://www.w3.org/2001/XMLSchema#string> ;
       fedora:lastModified "2016-03-01T00:13:59.157Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
       schema:name "Surviving a F4 Migration"^^<http://www.w3.org/2001/XMLSchema#string> ;
       schema:about "Fedora 4 and Repositories"^^<http://www.w3.org/2001/XMLSchema#string> ;
       fedora:writable "true"^^<http://www.w3.org/2001/XMLSchema#boolean> ;
       fedora:hasParent <http://localhost:8080/rest/grandpa/dad> ;
       fedora:numberOfChildren "0"^^<http://www.w3.org/2001/XMLSchema#int> ;
       fedora:exportsAs <http://localhost:8080/rest/grandpa/dad/me/fcr:export?format=jcr/xml> .

<http://localhost:8080/rest/grandpa/dad/me/fcr:export?format=jcr/xml> dc:format <http://fedora.
info/definitions/v4/repository#jcr/xml> .
```

Types of Resources in Fedora 4

Resource types (rdf:type) a.k.a “classes”

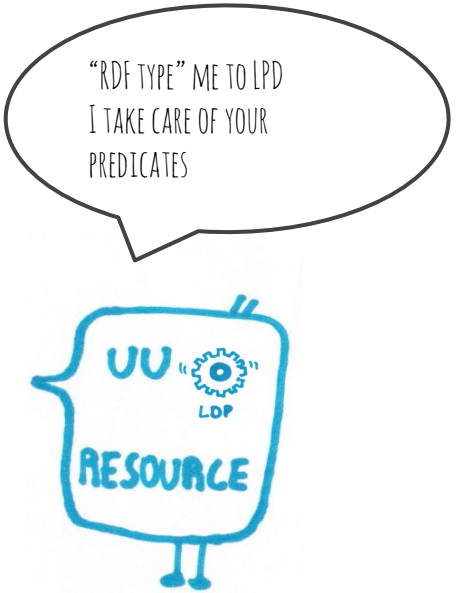
	Base	by Content	LDP	User Semantics
	a Fedora:Resource	a Fedora:Binary a Fedora:Container	a ldp:NonRDFSource a ldp:RDFSource a ldp:Container (a dp:IndirectContainer OR a ldp:DirectContainer)	a schema:Book (or whatever)

LDP: Linked Data Platform

RDF is complex: let’s make it easier

LPD provides a WEB based architecture for reading and writing Linked Data.

- F4 implements **LDP** (that is the reason we can access our resources via our **PATHS**/URIS and do Stuff on them)
- Aids in resource discovery (follow your nose approach)
- Some Resources are understood as **Containers**
- Those can “Magically” manage triples between resources: they provide a **service**



LDP Container (Or Basic Container)

<> a yummy:Banana ;
dc11:title “Extra Creamy” .

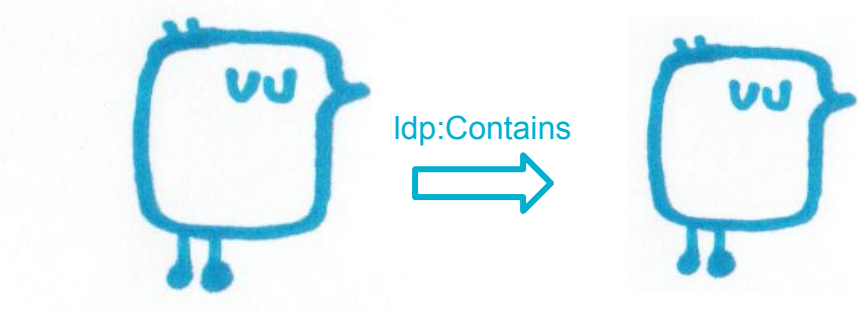
⟨/REST/FRUITS/BANANAS/⟩

```
HTTP POST “/rest/fruits/bananas/” with Slug = “greenbanana”
```

a ldp:Container ;

LDP Container (Or Basic Container)

⟨/REST/FRUITS/BANANAS/⟩ ⟨/REST/FRUITS/BANANAS/GREENBANANA⟩



LDP added a relation from
</rest/fruits/bananas/> to new
</rest/fruits/bananas/greenbanana/>

Consequences:

- default Tree builder
- Property becomes Server Managed:

if we remove “greenbanana”, triple
⟨/REST/FRUITS/BANANAS/⟩ ldp:Contains ⟨/REST/FRUITS/BANANAS/GREENBANANA⟩
gets also removed

LDP Direct Container

<> a yummy:Apple ;
dc11:title “Extra Juicy” .

⟨/REST/FRUITS/APPLES/⟩

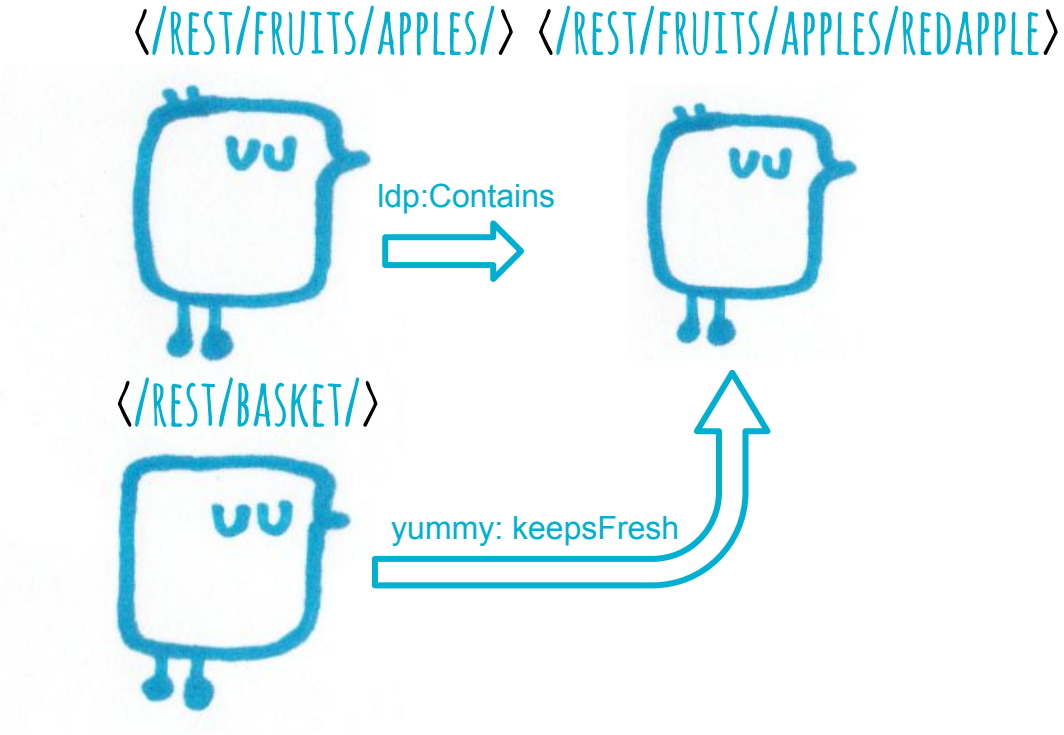
HTTP POST “/rest/fruits/apples/” with Slug = “redapple”

a ldp:DirectContainer ;
ldp:membershipResource ⟨/REST/BASKET/⟩ ;
ldp:hasMemberRelation yummy:keepsFresh .

⟨/REST/BASKET/⟩

a ldp:Container

LDP Direct Container



LDP added a relation from
</rest/basket/> to new
</rest/fruits/apples/redapple/>

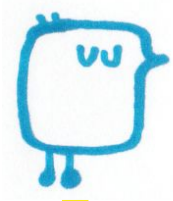
Consequences:

- Breaks the default Tree concept
- Property becomes Server Managed:

if we remove “redapple”, triple
⟨/REST/BASKET/⟩ yummy: keepsFresh ⟨/REST/FRUITS/APPLES/REDAPPLE⟩
gets also removed

Cool!

LDP Indirect Container



<> yummy:theBerry </REST/BLUEBERRYBUSH/BLUEBERRY01/> .

</REST/FRUITS/BERRIES/>

HTTP POST “/rest/fruits/berries/” with Slug = “firstBerry”

a ldp:IndirectContainer ;
ldp:membershipResource </REST/BASKET/> ;
ldp:hasMemberRelation yummy:keepsFresh ;
ldp:insertedContentRelation yummy:theBerry .

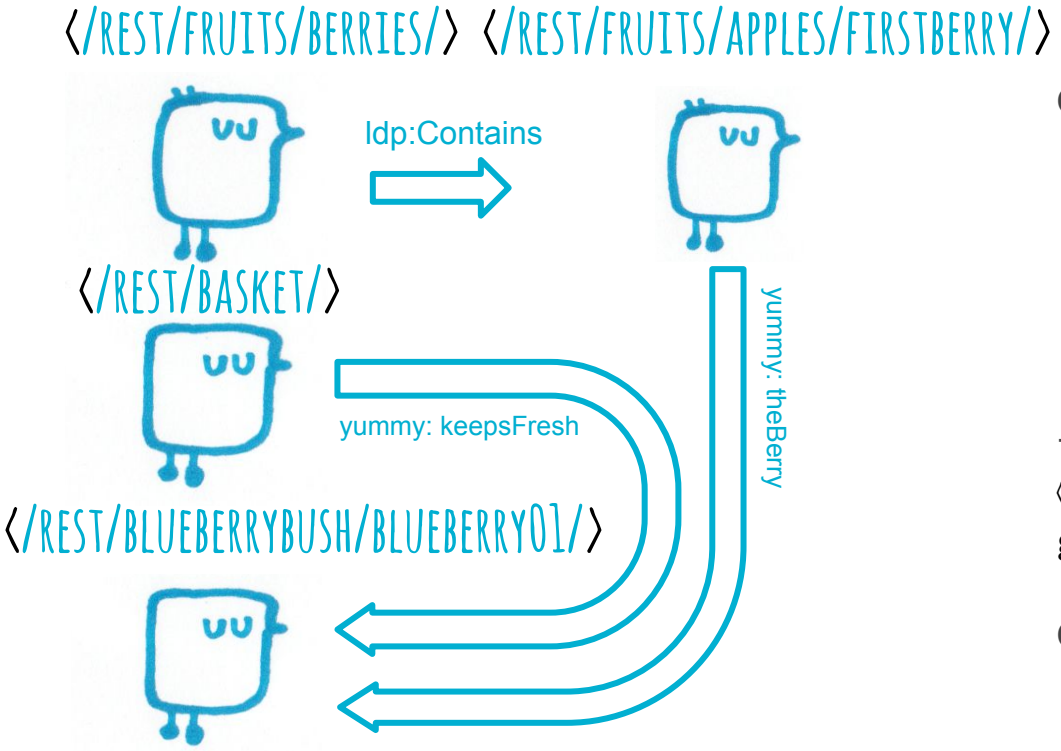
</REST/BASKET/>

a ldp:Container

</REST/BLUEBERRYBUSH/BLUEBERRY01/>

a ldp:Container,
yummy:Blueberry ;

LDP Indirect Container



LDP added a relation from
</rest/basket/> to new
</rest/blueberrybush/blueberry01/>

Consequences:

- Breaks the default Tree concept
- You have control over the Subject,Predicate and Object
- Property becomes Server Managed:

if we remove “firstberry”, triple
</REST/BASKET/> yummy: keepsFresh </REST/BLUEBERRYBUSH/BLUEBERRY01/>
gets also removed

Cool!

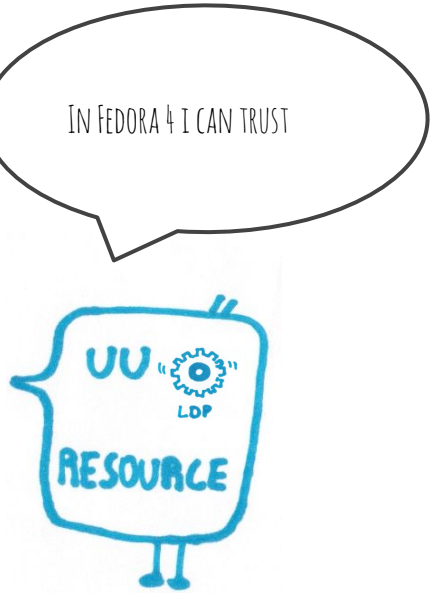
LDP helps building relations (think of self-deposit aid)

What other services does F4 provide for my Resources?

F4 Services

Fedora 4 is build to last

- Restful API on Resource URIs (Paths) (Create/Read/Update/Delete) = LDP
- Tombstones (Deleted resources keep their PATHS)
- Versioning (/fcr:versions) and Memento to come
- Authorization - WebACL
- Atomic Batch Operations = TX (start/do stuff -> commit/rollback?)
- Fixity (/fcr:fixity) on NonRDFSource

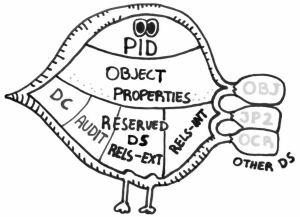


Let’s tie old and new concepts

Mapping F3 Object to F4 Resources (plural)

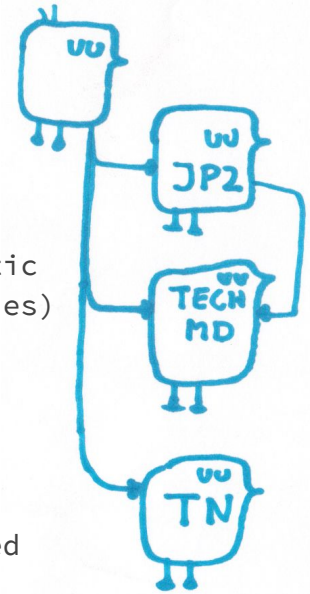
Fedora 3 Object

- ID:
  - PID (namespace:number)
- Base Properties
  - (FOXML, limited set)
- Class:
  - CMODEL (deals mostly with structure)
- Relations to other Objects
  - RELS-EXT: RDF
- Relations between DS
  - RELS-INT: RDF
- DATASTREAMS:
  - XML/BINARY/ETC



Fedora 4 Resource

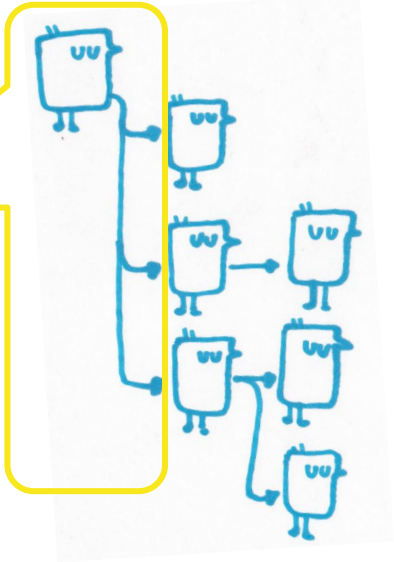
- ID
  - Resource Path (URI)
  - You can keep you old ones as RDF properties
- Class and Properties are RDF
  - class = rdf:type (Fully Semantic and defined in formal Ontologies)
  - Resource Description
  - Relations to other Resources
  - Metadata RDF (dc, MODS)
- DATASTREAMS (forget about them):
  - if can't be described as own properties, then another linked Resource
  - Binaries = <> rdf:type Fedora: Binary;



You could need a graph of F4 resources to match a Fedora 3 Object!

If i ask Fedora 4 (REST API) for a Resource, do i get the whole Graph tree up? No =(.

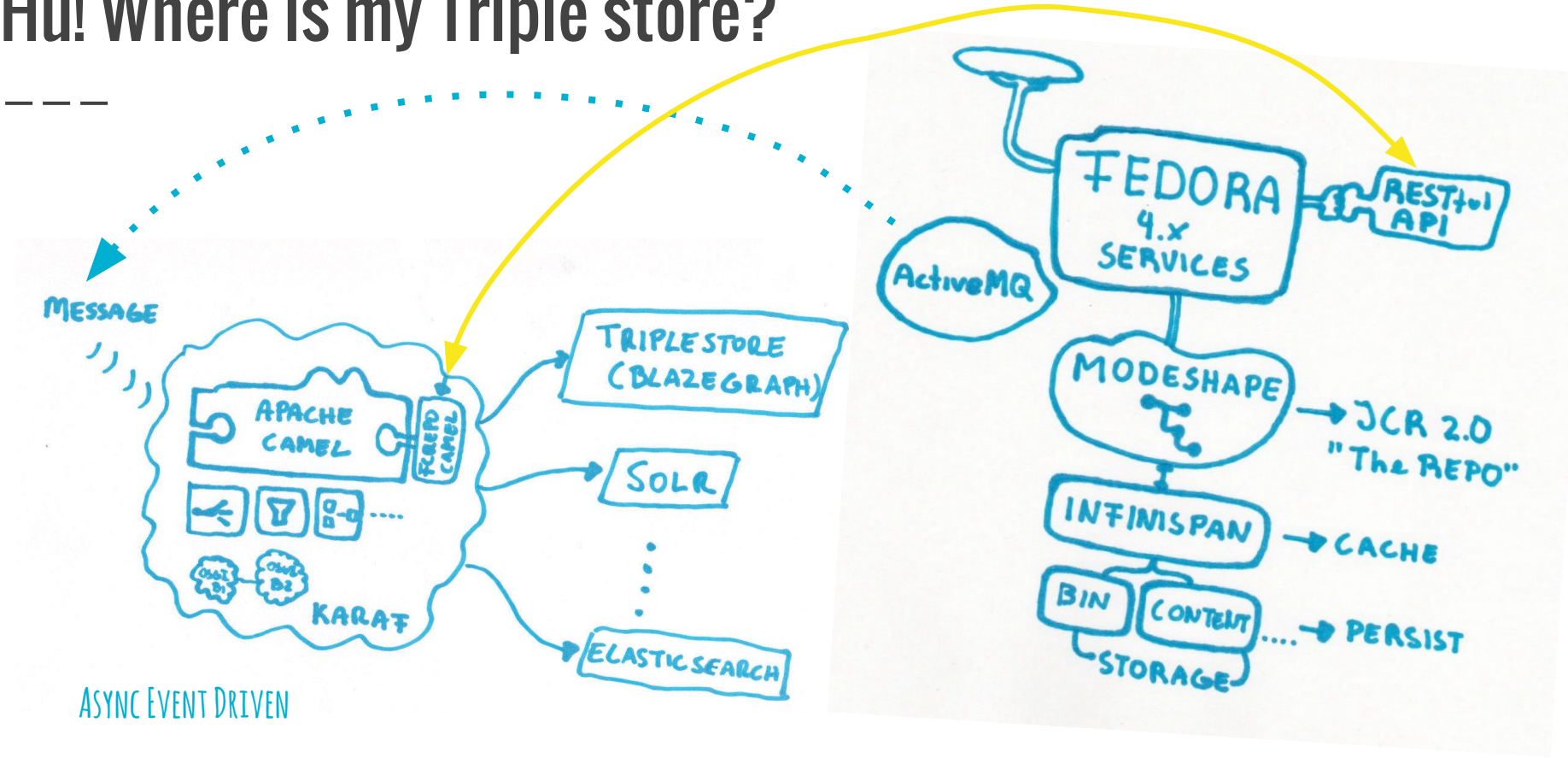
We get only the referenced Resource



So we need a triple store right?

Yeah, it's RDF, so a triple store is what we need

Hu! Where is my Triple store?



Async Event Driven

Things we have learned

- Fedora 4 is an API
- Fedora 4 is RDF based
- Resources are identified by PATHS and live in a tree
- LDP aids in tree creation/ triple creation
- We have cool services
- Async Workflow for indexing

There is nothing to fear (really)

Next Session:
Fedora 4 hands-on: PCDM and beyond