Reactive Resources Stream Pipeline

Resource Monad : Resource<ResourceOccurrence>;

ContentType (Data / Transforms)

* onOccurrence transform
* getOccurrences(S, P, O) transform
* getOccurringContexts(S, P, O) transform
* fromRepresentation(Representation)
* toRepresentation(ContentType)
* ContentTypes? Encoding? Models types?

Representation : ContentType instance

* ContentType
* Encoded State

ResourceOccurrence

* Representation
* onOccurrence(ResourceOccurrence)
* getOccurrences(S, P, O)
* getOccurringContexts(S, P, O)
* getAttributes() : Attributes (by means of occurrences / schema)
  + getAttribute(Attribute)
  + setAttribute(Attribute, Value)

FCA contexts? Prime IDs? Sets contexts? Dimensional contexts? Activation contexts? Hierarchies?: ResourceOccurrence Models ContentType Schema?

ResourceOccurrence(s)

* IDOccurrence
* SPO : IDOccurrence (Occurrence)
  + Subject : IDOccurrence
  + Predicate : IDOccurrence
  + Object : IDOccurrence
* Kind<Player, Attribute, Value> (Role / Type)
  + SubjectKind : Subject implements Kind<Subject, Predicate, Object>
  + PredicateKind : Predicate implements Kind<Predicate, Subject, Object>
  + ObjectKind : Object implements Kind<Object, Predicate, Subject>
* Statement (Data, SPOs Occurrences) : SPO
* Statement (Kinds, Kinds Occurrences) : SPO
  + Ex. (SK1, AttrX, ValY)
* Graph (Statements Occurrences)
* Model (Graph Occurrences)

ResourceOccurrence(s) Activation (ContentType handled, Resource Monad bound):

IDOccurrence::onOccurrence(SPO / Kinds)

SPO / Kinds::onOccurrence(Statement)

Statement::onOccurrence(Graph)

Graph::onOccurrence(Model)

Model::onOccurrence

getOccurrences(S, P, O)? (CPPE / RCV / FCA / Kinds / Alignment schema / instances inference. Filter / query / traversal).

Model::getOccurrences(S, P, O)

Graph::getOccurrences(S, P, O) : Models

Statement::getOccurrences(S, P, O) : Graphs

SPO / Kinds::getOccurrences(S, P, O) : Statements

IDOccurrence::getOccurrences(S, P, O) : SPO / Kinds

getOccurringContexts(S, P, O)? (CPPE / RCV / FCA / Kinds / Alignment schema / instances inference. Filter / query / traversal).

Model::getOccurringContexts(S, P, O) : Graphs

Graph::getOccurringContexts(S, P, O) : Statements

Statement::getOccurringContexts(S, P, O) : SPO / Kinds

SPO / Kinds::getOccurringContexts(S, P, O) : IDOccurrence

IDOccurrence::getOccurringContexts(S, P, O) : URN

Events: Model Messages.

Main Event Loop:

Naming, Registry, Index stream nodes Model Events Topic consumers / producers.

Main Event Loop Topic:

Stream nodes consume Model Events and publish augmented Model Event Context back to the stream for further augmentation.

Resource Activation: each stream node unfolds consumed Model Event and invokes occurrences events, traversing occurrences / occurring contexts getters. Node augmentation logic in Resources Representations ContentType(s) transforms.

Datasource node: Produces Models Events published to the topic and listens for Model Events for syncing backends state.

Producer node: consumes Model Events, publishes Activation API from Models and produces API interactions Model Events.

* Models? Augmented Model in Events Context?
* Naming: Resource Factory. URN Crafting / Matching. Aggregation (type / state / order inference).
* In step Model Context:
* IDOccurrence::onOccurrence
* SPO / Kinds::onOccurrence
* Statement::onOccurrence
* Graph::onOccurrence
* Model::onOccurrence
* Registry: Resource Repository. URNs Resolution / CRUD. Alignment (equivalence / upper matching, link prediction).
* In step Model Context:
* IDOccurrence::onOccurrence
* SPO / Kinds::onOccurrence
* Statement::onOccurrence
* Graph::onOccurrence
* Model::onOccurrence
* Index: Resource Contents URNs Resolution (inferences, transforms). Activation (possible verbs / state changes / transforms).
* In step Model Context:
* IDOccurrence::onOccurrence
* SPO / Kinds::onOccurrence
* Statement::onOccurrence
* Graph::onOccurrence
* Model::onOccurrence