Reactive Resources Stream Pipeline

Main Classes:

Resource Monad : Resource<ResourceOccurrence>;

ContentType (Representations Transforms)

* Transforms (XSLT / Custom Logic) for each ContentType type instance
* onOccurrence transform
* getOccurrences(S, P, O) transform
* getOccurringContexts(S, P, O) transform
* fromRepresentation(Representation)
* toRepresentation(ContentType)
* ContentType(s) (modelType/resourceOccurrenceType/encodingType) types instances:
  + [modelType]/ID/reference-tmrm
  + [modelType]/IDOccurrence/reference-tmrm
  + [modelType]/Subject/reference-tmrm
  + [modelType]/Predicate/reference-tmrm
  + [modelType]/Object/reference-tmrm
  + [modelType]/SubjectKind/reference-tmrm
  + [modelType]/PredicateKind/reference-tmrm
  + [modelType]/ObjectKind/reference-tmrm
  + [modelType]/Statement/reference-tmrm
  + [modelType]/Graph/reference-tmrm
  + [modelType]/Model/reference-tmrm
* Model Types:
  + FCA
  + DOM (OGM)
  + Activation (Actor / Role)
* Resource Occurrence Types:
  + ResourceOccurrence Classes
* Encoding Types:
  + Reference (Topic Maps TMRM)
  + RDF / RDFS
  + JSON-LD

Representation : ContentType instance

* ContentType
* Encoded State (XML / Custom Classes)

ResourceOccurrence

* Representation
* Methods (Dispatch to Representation ContentType Transforms):
  + onOccurrence(ResourceOccurrence occurrence) : ResourceOccurrence context (event)
  + getOccurrences(S, P, O)
  + getOccurringContexts(S, P, O)
  + getAttributes() : Attributes (by means of occurrences / schema)
    - getAttribute(Attribute)
    - setAttribute(Attribute, Value)
* Hierarchies (TODO) : ContentType hierarchies?

ResourceOccurrence(s) Classes:

* ID (URN : String, PrimeID : long)
* IDOccurrence : ID (occurringContext : IDOccurrence, embedding : long)
* 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, Attributes, Values) : SPO / Kinds (definitions)
  + Ex. (SK1, AttrX, ValY)
* Statement (SubjectKind, PredicateKind, ObjectKinds) : Kinds (schema)
  + Ex. (SK1, PK1, OK1)
* Graph (Statements Occurrences given their SPOs / Kinds contexts)
* Model (Graph Occurrences)
* ContentType (Models Occurrences)
* Representation?
* ResourceOccurrence?

Embed in URNs / ContentTypes ResourceOccurrence instance type and context instance ID. Example: urn:graph:subjectKind1. Dynamic Kinds ContentTypes.

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

ResourceOccurrence Events:

ResourceOccurrence::onOccurrence(ResourceOccurrence occurrence) : ResourceOccurrence context.

ID::onOccurrence(IDOccurrence) : URN

IDOccurrence::onOccurrence(SPO / Kinds) : ID

SPO / Kinds::onOccurrence(Statement) : IDOccurrence

Statement::onOccurrence(Graph) : SPO / Kinds

Graph::onOccurrence(Model) : Statement

Model::onOccurrence(ContentType) : Graph (merge)

ContentType::onOccurrence(Representation) : Model

Representation::onOccurrence(ResourceOccurrence) : ContentType

ResourceOccurrence Occurrences:

ResourceOccurrence::getOccurrences(S, P, O) : ResourceOccurrence. S, P, O filter /criteria / matching.

Leverages CPPE / RCV / FCA / Kinds / Alignment schema / instances inference / filter / query / traversal.

Representation::getOccurrences(S, P, O) : ResourceOccurrence

ContentType::getOccurrences(S, P, O) : Representation

Model::getOccurrences(S, P, O) : ContentType

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

ID::getOccurrences(S, P, O) : IDOccurrence

ResourceOccurrence Occurring Contexts:

ResourceOccurrence::getOccurringContext(S, P, O) : ResourceOccurrence. S, P, O filter /criteria / matching.

Leverages CPPE / RCV / FCA / Kinds / Alignment schema / instances inference / filter / query / traversal.

ResourceOccurrence::getOccurringContexts(S, P, O) : Representation

Representation::getOccurringContexts(S, P, O) : ContentType

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

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) : ID

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

Runtime:

Events: Model Messages.

Main Event Loop:

Aggregation, Alignment, Activation stream nodes Model Events Topic consumers / producers. Matches for Models ContentType(s).

Topic streaming:

Stream nodes consume Model (Representation?) Events and publish augmented Model (Representation?) Events Context back to the stream for further augmentation. Augmentation nodes update Model (Representation?) ContentType(s).

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

Datasource node: Produces Models (Representation?) Events published to the topic and listens for Model (Representation?) Events for syncing back backends state.

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

* Augmented Model in Events Context
* Aggregation Node: type / state / order inference. FCA Model.
* Augmentation Node unfolded Model Context ResourceOccurrence events stream (ContentType transforms):
* Functional Input Model Traversal (Representation unfolding):
  + Representation::getOccurrences(S, P, O) : ResourceOccurrence
  + Representation::onOccurrence(ResourceOccurrence)
  + ContentType::onOccurrence(Representation) : Model
  + Model::onOccurrence(ContentType) : Graph
  + Graph::onOccurrence(Model) : Statement
  + Statement::onOccurrence(Graph) : SPO / Kinds
  + SPO / Kinds::onOccurrence(Statement) : IDOccurrence
  + IDOccurrence::onOccurrence(SPO / Kinds) : ID
  + ID::onOccurrence(IDOccurrence) : URN
* Functional Output Model Building (Representation folding):
  + ID::getOccurrences(S, P, O) : IDOccurrence
  + IDOccurrence::getOccurrences(S, P, O) : SPO / Kinds
  + SPO / Kinds::getOccurrences(S, P, O) : Statements
  + Statement::getOccurrences(S, P, O) : Graphs
  + Graph::getOccurrences(S, P, O) : Models
  + Model::getOccurrences(S, P, O) : ContentType
  + ContentType::getOccurrences(S, P, O) : Representation
  + Representation::getOccurrences(S, P, O) : ResourceOccurrence
* Publish Augmented Model (Representation?)
* Alignment Node: equivalence / upper ontology alignment, link prediction. DOM (OGM) Model.
* Augmentation Node unfolded Model Context ResourceOccurrence events stream (ContentType transforms):
* Functional Input Model Traversal (Representation unfolding):
  + Representation::getOccurrences(S, P, O) : ResourceOccurrence
  + Representation::onOccurrence(ResourceOccurrence)
  + ContentType::onOccurrence(Representation) : Model
  + Model::onOccurrence(ContentType) : Graph
  + Graph::onOccurrence(Model) : Statement
  + Statement::onOccurrence(Graph) : SPO / Kinds
  + SPO / Kinds::onOccurrence(Statement) : IDOccurrence
  + IDOccurrence::onOccurrence(SPO / Kinds) : ID
  + ID::onOccurrence(IDOccurrence) : URN
* Functional Output Model Building (Representation unfolding):
  + ID::getOccurrences(S, P, O) : IDOccurrence
  + IDOccurrence::getOccurrences(S, P, O) : SPO / Kinds
  + SPO / Kinds::getOccurrences(S, P, O) : Statements
  + Statement::getOccurrences(S, P, O) : Graphs
  + Graph::getOccurrences(S, P, O) : Models
  + Model::getOccurrences(S, P, O) : ContentType
  + ContentType::getOccurrences(S, P, O) : Representation
  + Representation::getOccurrences(S, P, O) : ResourceOccurrence
* Publish Augmented Model (Representation?)
* Activation Node: possible verbs / state changes / transforms. DCI (Actor / Role) Model.
* Augmentation Node unfolded Model Context ResourceOccurrence events stream (ContentType transforms):
* Functional Input Model Traversal (Representation unfolding):
  + Representation::getOccurrences(S, P, O) : ResourceOccurrence
  + Representation::onOccurrence(ResourceOccurrence)
  + ContentType::onOccurrence(Representation) : Model
  + Model::onOccurrence(ContentType) : Graph
  + Graph::onOccurrence(Model) : Statement
  + Statement::onOccurrence(Graph) : SPO / Kinds
  + SPO / Kinds::onOccurrence(Statement) : IDOccurrence
  + IDOccurrence::onOccurrence(SPO / Kinds) : ID
  + ID::onOccurrence(IDOccurrence) : URN
* Functional Output Model Building (Representation folding):
  + ID::getOccurrences(S, P, O) : IDOccurrence
  + IDOccurrence::getOccurrences(S, P, O) : SPO / Kinds
  + SPO / Kinds::getOccurrences(S, P, O) : Statements
  + Statement::getOccurrences(S, P, O) : Graphs
  + Graph::getOccurrences(S, P, O) : Models
  + Model::getOccurrences(S, P, O) : ContentType
  + ContentType::getOccurrences(S, P, O) : Representation
  + Representation::getOccurrences(S, P, O) : ResourceOccurrence
* Publish Augmented Model (Representation?)
* Naming Helper
* Registry Helper
* Index Helper