* Project:
* Environment:
* Frameworks / Libraries. Docs.
* Runtime. Tools. Services:
* Jersey / CDI. Spring.
* RDF4J (OpenRDF Sesame).
* R2RQ. Teiid. Metamodel. Any23.
* Predictions: ML / BigData (Spark).
* Index: Lucene / Solr. Quads VSM.
* Registry (Zippers): hierarchical key / value. Map Reduce.
* Naming: DIDs Services. [schema.org](http://schema.org), [sameAs.org](http://sameas.org). Matching (mappings). WordNet, OpenNLP, Wikipedia.
* Reactive (Bus): Vert.x
* Dataflow: SCDF Pipelines.
* Designer: OpenRefine. Protege Web. Editors / browsers (Forms / Flows).
* Protocols (SAILs): SPARQL, JSON-LD, JDBC (OGM / OData: Driver DatabaseMetadata). Messages (Functional / Augmentations) Declarative Services (templates).
* Clients:
* Debug Console.
* Declarative UI: ZK / XUL Templates (Content types / Components activation).
* Services APIs: Declaratively stated services: templates / queries.
* IDE. Runtime Deployment (Application).
* Clients: OData / JDBC / WS-\* (CXF) / HATEOAS / etc.
* Lectures. Topics.
* Schedule. Topics.
* Triple store backend. RDF4J:
* Quad Layers SAIL. Core layers schema. IO: RDFS / OWL (classes, types, sameAs, etc.). SPARQL. Upper Ontology.
* Objects Layers SAIL. IO: Objects (Layers / DOM / OGM) HATEOAS.
* Functional Layers SAIL. IO: Augmentations: Browse / Navigate Model (Objects / Monads). Message Driven.
* Augmentations / Domain Services / Dataflow SAIL. IO: Messages built in Functional SAIL / Augmentation Contexts results streams. Contexts: Model Contexts, Domains Services REST Contexts URIs (APIs), connector / clients plugins (signatures). DIDs Domain Services: internal URIs. DIDs / URI mappings / APIs. Signatures. Bus.
* Augmentations: Perform Aggregation, Activation, Alignment according input Message. Update Model Contexts (upwards), Occurrences (downwards) according Augmentation stream results (Contexts). Enqueue further dataflow messages.
* Dataflow (Predicates, Mappings, Functions) domain / range (signatures). SAIL registry (types / kinds bus). Bus: topics, queues. Reactive interfaces. Dispatcher.
* Services SAIL I/O: Connectors / Clients. Distributed nodes Connector sources.
* URI: Jersey / CDI APIs. Persistence interface template methods.
* Layers APIs.
* Statement URI: Layer URI + Layer URI instance ID.
* Layers CRUD tests.
* Functional APIs.
* Functional tests (Inside Flat Map: dataflow over Object APIs):
* Predicate tests. Activation.
* Mapping tests. Aggregation.
* Function tests. Alignment.
* (MessageDomain, Predicate, Mapping, Function);
* Flat Map: Dataflow chaining over Message transforms and dataflow results.
* Augmentations performed over individual Message roles in dataflow and in dataflow with other Message roles (APIs / contexts).
* Message: Functional wrapper of Layers Contexts hierarchies statements. Hierarchy: template methods, predicate, mapping, function behaviors (inside flat map and on message statement role positions: internal dataflows).
* Predicate, Mapping, Function, URI, Statement, Value. Root hierarchy types.
* (Statement, OldEmployees : Predicate URI / Activation, SalaryUpdate : Mapping URI / Aggregation, Percentage : Function URI / Alignment
* (Domain / Context matching signatures (async streams / topics), Employee : Aggregation / Predicate, Developer : Activation / Mapping / Attributes, ProgrammingLanguages / Alignment / Function / Values);
* Monad::of(URI instance : hier);
* Monad::flatMap(Statement) : Monad (of Statement hier).
* Dispatch: each layer instance consumes matching or forwards upwards (layers signatures / zippers) incoming messages. Resulting Message enqueued for further processing.
* URI::onMessage (template methods: context, occurrence, attribute, value, role, contexts, occurrences, etc.)