# LD4PE Competency Index

2016-01-18

* generated [Compindex-generated.md](https://github.com/ld4pe/cieb/blob/master/CompIndex/CompIndex-generated.md) from [Compindex.txt](https://github.com/ld4pe/cieb/blob/master/CompIndex/CompIndex.txt)
* converted [Compindex-generated.md](https://github.com/ld4pe/cieb/blob/master/CompIndex/CompIndex-generated.md) into GDOC (via DOCX) for collecting comments
* see latest published version at <http://explore.dublincore.net/comp-index/> (may be out-of-date)

### Topic cluster

* Topic
  + Competency: a tweet-length phrase expressing a concept or skill to be learned
    - Benchmark: specific task that demonstrates competency

### Fundamentals of Resource Description Framework

* Identity in RDF
  + Knows that anything can be named with URIs, such as agents, places, events, artifacts, and concepts
* RDF data model
  + Knows the subject-predicate-object component structure of a triple
  + Understands the difference between literals and non-literals
  + Understands that any URI or literal denotes some "thing" in the world whether real, imagined, or conceptual; and, all things are called "resources"
  + Understands that "resources" may be divided into groups called classes and the members of a class are known as instances of the class (rdf:type)
  + Understands the use of datatypes and language tags with literals
  + Understands blank nodes and their appropriate use
  + Formulates QNames as a shorthand mechanism in writing prefixes for long URIs
    - Uses prefixes for URIs in RDF specifications and data
  + Understands the concept of the named graph
  + Can articulate differences between the RDF abstract data model and the XML and relational models
  + Understands the use of RDF Schema to create and interpret RDF vocabularies
  + Understands the RDF abstract data model as a directed labeled graph
  + Knows graphic conventions for depicting RDF-based models
    - Can use graphing or modeling software to share those models with others
* Related data models
  + Distinguishes validation schemas (eg, XML) and inferencing schemas (eg, RDF Schema) and their respective benefits
  + Differentiates hierarchical document models (eg, XML) and graph models (RDF)
* RDF serialization
  + Distinguishes the RDF abstract data model and its concrete serializations
    - Expresses RDF data in Turtle syntax
    - Expresses RDF data in RDFa syntax
  + Understands RDF serializations and graphical representations as just alternative ways to encode a given set of triples (RDF graph)
    - Uses tools to convert RDF data between different serializations

### Fundamentals of Linked Data

* Web technology
* Linked data principles
* Linked Data architectures and services
* Linked Data policies and best practices
  + Knows the primary organizations related to Linked Data standardization
    - Participates in developing standards and best practice with relevant organizations such as W3C
* Non-RDF Linked Data

### RDF vocabularies and application profiles

* Finding RDF-based vocabularies
  + Uses available resources for named entity recognition, extraction, and reconciliation
  + Retrieves and accesses RDF data from the "open Web"
* Designing RDF-based vocabularies
  + Uses RDF Schema to express semantic relationships within a vocabulary
    - Correctly uses sub-class relationships in support of inference
    - Correctly uses sub-property relationships in support of inference
  + Knows naming conventions for properties and classes
  + Reuses published properties and classes where available
    - Uses portals and registries to find existing RDF-based vocabularies
  + Coins namespace URIs, as needed, for any new properties and classes required
    - Drafts a policy for coining URIs for properties and classes
    - Chooses "hash"- or "slash"-based URI patterns based on requirements
* Maintaining RDF vocabularies
  + Understands policy options for persistence guarantees
    - Can draft a persistence policy
* Versioning RDF vocabularies
  + Knows technical options for the form, content, and granularity of versions
    - Can express and justify a versioning policy
* Publishing RDF vocabularies
  + Understands the typical publication formats for RDF vocabularies, their relative advantages, and options for content negotiation
* Mapping RDF vocabularies
* RDF application profiles
  + Identifies real-world entities in an application domain as requirements for RDF classes
  + Identifies resource attributes and relationships between domain entities as requirements for RDF properties
  + Investigates how others have modeled the same or similar application domains
    - Communicates a domain model with words and diagrams
    - Participates in the social process of developing application profiles

### Creating and transforming RDF Data

* Managing identifiers (URIs)
* Creating RDF data
  + Structures data using blank nodes where appropriate
  + Generates RDF data from non-RDF sources
* Versioning RDF data
* RDF data provenance
* Cleaning and reconciling RDF data
  + Cleans a dataset by finding and correcting errors, removing duplicates and unwanted data
* Mapping and enriching RDF data

### Interacting with RDF Data

* Finding RDF Data
  + Retrieves and accesses RDF data from the "open Web"
  + Uses relevant resources to discover existing Linked Data datasets
  + Monitors and updates lists which report the status of SPARQL endpoints
  + Uses available vocabularies for dataset description to support their discovery
  + Registers datasets with relevant services for discovery
* Programming RDF Data
* Querying RDF Data
  + Understands that a SPARQL query matches an RDF graph against a pattern of triples with fixed and variable values
  + Understands the basic syntax of a SPARQL query
    - Uses angle brackets for delimiting URIs
    - Uses question marks for indicating variables
    - Uses PREFIX for base URIs
  + Formulates advanced queries on data containing blank nodes
  + Demonstrates a working knowledge of the forms and uses of SPARQL result sets (SELECT, CONSTRUCT, DESCRIBE, and ASK)
    - Uses the SELECT clause to identify the variables to appear in a table of query results
    - Uses the WHERE clause provide the graph pattern to match against the graph data
    - Uses variables in SELECT and WHERE clauses to yield a table of results
    - Uses ASK for a simple True/False result in testing whether a query pattern has a solution
    - Uses DESCRIBE to extract a single graph containing RDF data about resources
    - Uses CONSTRUCT to extract and transform results into a single RDF graph specified by a graph template
  + Understands how to combine and filter graph patterns using operators such as UNION, OPTIONAL, FILTER, and MINUS
    - Uses UNION to formulate queries with multiple possible graph patterns
    - Uses OPTIONAL to formulate queries to return the values of optional variables when available
    - Uses FILTER to formulates queries that eliminate solutions from a result set
    - Uses NOT EXISTS to limit whether a given graph pattern exists in the data
    - Uses MINUS to remove matches from a result based on the evaluation of two patterns
    - Uses NOT IN to restrict a variable to not being in a given set of values
  + Understands the major SPARQL result set modifiers, e.g., to limit or sort results, or to return distinct results only once
    - Uses ORDER BY to define ordering conditions by variable, function call, or expression
    - Uses DISTINCT to ensure solutions in the sequence are unique
    - Uses OFFSET to control where the solutions processed start in the overall sequence of solutions
    - Uses LIMIT to restrict the number of solutions processed for query results
    - Uses projection to transforms a solution sequence into one involving only a subset of the variables
  + Understands the use of SPARQL functions and operators
    - Uses the regular expression (regex()) function for string matching
    - Uses aggregates to apply expressions over groups of solutions (GROUP BY, COUNT, SUM, AVG, MIN) for partitioning results, evaluating projections, and filtering
    - Uses the lang() function to return the language tag of an RDF literal
    - Uses the langMatches() function to match a language tag against a language range
    - Uses the xsd:decimal(expn) function to convert an expression to an integer
    - Uses the GROUP BY clause to transforms a result set so that only one row will appear for each unique set of grouping variables
    - Uses the HAVING clause to apply a filter to the result set after grouping
  + Differentiates between a Default Graph and a Named Graph, and formulates queries using the GRAPH clause
    - Formulates advanced queries using FROM NAMED and GRAPH on local data
    - Formulate advanced queries using FROM NAMED on remote data
  + Formulate advanced queries using subqueries
  + Uses a temporary variable to extend a query
  + Understands the role of Property Paths and how they are formed by combining predicates with regular expression-like operators
  + Understands the concept of Federated Search
    - Formulates advanced queries on a remote SPARQL endpoint using the SERVICE directive
    - Uses federated query to query over a local graph store and one or more other SPARQL endpoints
    - Pulls data from a different SPARQL endpoints in one single query using the SERVICE directive
  + Understands the principles and practice of inferencing
    - Uses common entailment regimes and understands their uses
  + Understands the role of formally declared domains and ranges for inferencing
  + Understands how reasoning can be used for integrating diverse datasets
  + Converts/manipulates SPARQL query outputs (RDF-XML, JSON) to the exact format required by a third party tools and APIs
  + Formulates queries using FROM with URLs and local files
  + Reads and understands high-level descriptions of the classes and properties of a dataset in order to write queries
  + Uses available tools, servers, and endpoints to issue queries against a dataset
    - Execute SPARQL queries using the Jena ARQ command-line utility
    - Queries multiple local data files using ARQ
    - Uses ARQ to evaluate queries on local data
    - Uses Fuseki server to evaluate queries on a dataset
    - Queries multiple data files using Fuseki
    - Accesses DBPedia's SNORQL/SPARQL endpoint and issues simple queries
* Visualizing RDF Data
  + Uses publicly available tools to visualize data
    - Uses Google FusionTables to create maps and charts
  + Distills results taken from large datasets so that visualizations are human-friendly
  + Converts/manipulates SPARQL query outputs (RDF-XML, JSON) to the exact format required by a third party tools and APIs
* Reasoning over RDF
* Assessing RDF data quality
* RDF Data analytics
  + Uses available ontology browsing tools to explore the ontologies used in a particular dataset
* Manipulating RDF Data
  + Knows the SPARQL 1.1 Update language for updating, creating, and removing RDF graphs in a Graph Store
    - Uses INSERT to add triples
    - Uses DELETE to remove triples
    - Uses the INSERT/DELETE operation to update triples
    - Uses a CONSTRUCT query to preview changes before executing an INSERT/DELETE operation
  + Knows the SPARQL 1.1 Graph Store HTTP protocol for updating graphs on a Web server (in arguably "RESTful" style)
    - Uses GET to retrieve triples from a default graph or a named graph
    - Uses PUT to insert set of triples into a new graph (or replace an existing graph)
    - Uses DELETE to remove a graph
    - Uses POST to add triples to an existing graph
    - Uses proper syntax to request specific media types, such as Turtle
  + Understands the difference between SQL query language (which operates on database tables) and SPARQL (which operates on RDF graphs)

### Creating Linked Data applications

* Storing RDF data
* Linked Data application architecture
* Linked Data mashups