# LD4PE Competency Index

2016-05-12

* 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 (may be out-of-date)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: Tweet-length assertion of knowledge, skill, or habit of mind
    - Benchmark: Action demonstrating accomplishment in related competencies

### Fundamentals of Resource Description Framework

* Identity in RDF
  + Knows that anything can be named with Uniform Resource Identifiers (URIs), such as agents, places, events, artifacts, and concepts.
  + Understands that a "real-world" thing may need to be named with a URI distinct from the URI for information about that thing.
  + Recognizes that URIs are "owned" by the owners of their respective Internet domains.
* RDF data model
  + Knows the subject-predicate-object component structure of a triple.
  + Understands that URIs and literals denote things in the world ("resources") real, imagined, or conceptual.
  + Understands the difference between literals and non-literal resources.
  + Understands that resources are declared members (instances) of classes using the property rdf:type.
  + Understands the use of datatypes and language tags with literals.
  + Understands blank nodes and their uses.
  + Understands that QNames define shorthand prefixes for long URIs.
  + Understands a named graph as one of the collection of graphs comprising an RDF dataset, with a graph name unique in the context of that dataset
  + Understands how an RDF class (named set of things) fundamentally differs from an object-oriented programming class, which defines a type of object bundling "state" (attributes with data values) and "behavior" (functions that operate on state).
  + Understands how a namespace, informally used in the RDF context for a namespace URI or RDF vocabulary, fundamentally differs from the namespace of data attributes and functions (methods) defined for an object-oriented class.
    - Uses prefixes for URIs in RDF specifications and data.
  + Understands the concept of the named graph.
  + Articulates 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
  + Grasps essential differences between schemas for syntactic validation (e.g., XML) and for inferencing (RDF Schema).
  + Differentiates hierarchical document models (eg, XML) and graph models (RDF).
  + Understands how an RDF class (naming a set of things in the world) fundamentally differs from an object-oriented programming "class", or "data type", which is used to create objects that bundle "state" (attributes with data values) and "behavior" (functions that operate on state).
* RDF serialization
  + Distinguishes the RDF abstract data model and concrete serializations of RDF data.
    - Expresses data in serializations such as RDF/XML, N-Triples, Turtle, N3, Trig, JSON-LD, and RDFa.
  + Understands RDF serializations as interchangeable encodings of a given set of triples (RDF graph).
    - Uses tools to convert RDF data between different serializations.

### Fundamentals of Linked Data

* Web technology
  + Knows the origins of the World Wide Web (1989) as a non-linear interactive system, or hypermedia, built on the Internet.
  + Understands that Linked Data (2006) extended the notion of a web of documents (the Web) to a notion of a web of finer-grained data (the Linked Data cloud).
  + Knows HyperText Markup Language, or HTML (1991+), as a language for "marking up" the content and multimedia components of Web pages.
  + Knows HTML5 (2014) as a version of HTML extended with support for complex web and mobile applications.
  + Knows Hypertext Transfer Protocol, or HTTP (1991+), as the basic technology for resolving hyperlinks and transferring data on the World Wide Web.
  + Knows Representational State Transfer, or REST (2000) as a software architectural style whereby browsers can exchange data with web servers, typically on the basis of well-known HTTP actions.
  + Knows that Uniform Resource Identifiers, or URIs (1994), include Uniform Resource Locators (URLs, which locate web pages) as well as location-independent identifiers for physical, conceptual, or web resources.
* Linked data principles
  + Knows Tim Berners-Lee's principles of Linked Data: use URIs to name things, use HTTP URIs that can be resolved to useful information, and create links to URIs of other things.
  + Knows the "five stars" of Open Data: put data on the Web, preferably in a structured and preferably non-proprietary format, using URIs to name things, and link to other data.
* 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
* 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 the naming conventions for RDF 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.
  + Knows Web Ontology Language, or OWL (2004), as a RDF vocabulary of properties and classes that extend support for expressive data modeling and automated inferencing (reasoning).
  + Knows that the word "ontology" is ambiguous, referring to any RDF vocabulary, but more typically a set of OWL classes and properties designed to support inferencing in a specific domain.
  + Knows Simple Knowledge Organization System, or SKOS (2009), an RDF vocabulary for expressing concepts that are labeled in natural languages, organized into informal hierarchies, and aggregated into concept schemes.
  + Knows SKOS eXtension for Labels, or SKOS-XL (2009), a small set of additional properties for describing and linking lexical labels as instances of the class Label.
  + Understands that in a formal sense, a SKOS concept is not an RDF class but an instance and, as such, is not formally associated with a set of instances ("class extension").
  + Understands that SKOS can express a flexibly associative structure of concepts without enabling the more rigid and automatic inferences typically specified in a class-based OWL ontology.
  + Understands that in contrast to OWL sub-class chains, hierarchies of SKOS concepts are designed not to form transitive chains automatically because this is not how humans think or organize information.
* 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.
  + Understands the trade-offs between publishing RDF vocabularies in periodic, numbered releases versus more continual or incremental approaches.
    - Can express and justify a versioning policy.
* Publishing RDF vocabularies
  + Understands that to be "dereferencable", a URI should be usable to retrieve a representation of the resource it identifies.
    - Ensures that when dereferenced by a Web browser, a URI returns a representation of the resource in human-readable HTML.
    - Ensures that when dereferenced by an RDF application, a URI returns representation of the resource in the requested RDF serialization syntax.
  + Understands the typical publication formats for RDF vocabularies and their relative advantages.
  + Understands the purpose of publishing RDF vocabularies in multiple alternative formats using content negotiation.
* Mapping RDF vocabularies
  + Understands that the properties of hierarchical subsumption within an RDF vocabulary -- rdfs:subPropertyOf and rdfs:subClassOf -- can also be used to express mappings between vocabularies.
  + Understands that owl:equivalentProperty and owl:equivalentClass may be used when equivalencies between properties or between classes are exact.
* 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)
  + Understands that to be "persistent", a URI must have a stable, well-documented meaning and be plausibly intended to identify a given resource in perpetuity.
  + Understands trade-offs between "opaque" URIs and URIs using version numbers, server names, dates, application-specific file extensions, query strings or other obsoletable context.
  + Recognizes the desirability of a published namespace policy describing an institution's commitment to the persistence and semantic stability of important URIs.
* Creating RDF data
  + Generates RDF data from non-RDF sources.
  + Knows methods for generating RDF data from tabular data in formats such as Comma-Separated Values (CSV).
  + Knows methods such as Direct Mapping of Relational Data to RDF (2012) for transforming data from the relational model (keys, values, rows, columns, tables) into RDF graphs.
* 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
  + Uses available resources for named entity recognition, extraction, and reconciliation.

### Interacting with RDF Data

* Finding RDF Data
  + Knows relevant resources for discovering existing Linked Data datasets.
    - Retrieves and accesses RDF data from the "open Web".
  + 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.
* Processing RDF data using programming languages
  + Understands how components of the RDF data model (datasets, graphs, statements, and various types of node) are expressed in the RDF library of a given programming language by constructs such as object-oriented classes.
    - Uses an RDF programming library to serialize RDF data in available syntaxes.
    - Uses RDF-specific programming methods to iterate over components of RDF data.
    - Uses RDF-library-specific convenience representations for common RDF vocabularies such as RDF, Dublin Core, and SKOS.
    - Programatically associates namespaces to prefixes for use in serializing RDF or when parsing SPARQL queries.
    - Uses RDF programming libraries to extract RDF data from CSV files, databases, or web pages.
    - Uses RDF programming libraries to persistently stores triples in memory, on disk, or to interact with triple stores.
    - Programatically infers triples using custom functions or methods.
  + Understands how the pattern matching of SPARQL queries can be expressed using functionally equivalent constructs in RDF programming libraries.
    - Uses RDF-specific programming methods to query RDF data and save the results for further processing.
    - Uses utilities and convenience functions the provide shortcuts for frequently used patterns, such as matching the multiple label properties used in real data.
    - Uses RDF libraries to process various types of SPARQL query result.
* 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 True/False test for a match to a query pattern.
    - 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.
  + 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 data
  + 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.
  + Knows that Web Ontology Language (OWL) is available in multiple "flavors" that are variously optimized for expressivity, performant reasoning, or for applications involving databases or business rules.
  + Understands that OWL Full supports all available constructs and is most appropriately used when reasoning performance is not a concern.
* 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/DELETE 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 "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