LD4PE Competency Index

Version: 2017-06-25 15:20:21

Raw file: https://github.com...

View at: https://ld4pe.github.com...

[A] __Topic Cluster__

- [B] __Topic__
  - [C] __Competency__: Tweet-length assertion of knowledge, skill, or habit of mind.
    - [D] __Benchmark__: Action demonstrating accomplishment in related competencies.

[A] Fundamentals of Resource Description Framework

- [B] Identity in RDF
  - [C] Knows that anything can be named with Uniform Resource Identifiers (URIs), such as agents, places, events, artifacts, and concepts.
  - [C] Understands that a "real-world" thing may need to be named with a URI distinct from the URI for information about that thing.
  - [C] Recognizes that URIs are "owned" by the owners of their respective Internet domains.
  - [C] 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.
- [B] RDF data model
  - [C] Knows the subject-predicate-object component structure of a triple.
  - [C] Understands the difference between literals and non-literal resources.
  - [C] Understands that URIs and literals denote things in the world ("resources") real, imagined, or conceptual.
  - [C] Understands that resources are declared to be members (instances) of classes using the property rdf:type.
  - [C] Understands the use of datatypes and language tags with literals.
  - [C] Understands blank nodes and their uses.
  - [C] Understands that QNames define shorthand prefixes for long URIs.
    - [D] Uses prefixes for URIs in RDF specifications and data.
  - [C] Articulates differences between the RDF abstract data model and the XML and relational models.
  - [C] Understands the RDF abstract data model as a directed labeled graph.
  - [C] Knows graphic conventions for depicting RDF-based models.

        - [D] Can use graphing or modeling software to share those models with others.
    - [C] 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.
    - [C] 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.

- [B] Related data models

    - [C] Grasps essential differences between schemas for syntactic validation (e.g., XML) and for inferencing (RDF Schema).
    - [C] Differentiates hierarchical document models (eg, XML) and graph models (RDF).
    - [C] 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).

- [B] RDF serialization

    - [C] Understands RDF serializations as interchangeable encodings of a given set of triples (RDF graph).

        - [D] Uses tools to convert RDF data between different serializations.

    - [C] Distinguishes the RDF abstract data model and concrete serializations of RDF data.

        - [D] Expresses data in serializations such as RDF/XML, N-Triples, Turtle, N3, Trig, JSON-LD, and RDFa.

[A] Fundamentals of Linked Data

- [B] Web technology

    - [C] Knows the origins of the World Wide Web (1989) as a non-linear interactive system, or hypermedia, built on the Internet.
    - [C] 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).
    - [C] Knows HyperText Markup Language, or HTML (1991+), as a language for "marking up" the content and multimedia components of Web pages.
    - [C] Knows HTML5 (2014) as a version of HTML extended with support for complex web and mobile applications.
    - [C] Knows Hypertext Transfer Protocol, or HTTP (1991+), as the basic technology for resolving hyperlinks and transferring data on the World Wide Web.
    - [C] 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.

- [B] Linked Data principles

    - [C] 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.
    - [C] 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.

- [B] Linked Data policies and best practices
    - [C] Knows the primary organizations related to Linked Data standardization.
        - [D] Participates in developing standards and best practice with relevant organizations such as W3C.
- [B] Non-RDF linked data

[A] RDF vocabularies and application profiles

- [B] Finding RDF-based vocabularies
  * [D] [MOVE]Knows portals and registries for finding RDF-based vocabularies.
  * [D] Finds properties and classes in the Linked Open Vocabularies (LOV) observatory and explores their versions and dependencies.
- [B] Designing RDF-based vocabularies
    - [C] Uses RDF Schema to express semantic relationships within a vocabulary.
        - [D] Correctly uses sub-class relationships in support of inference.
        - [D] Correctly uses sub-property relationships in support of inference.
    - [C] Reuses published properties and classes where available.
    - [C] Coins namespace URIs, as needed, for any new properties and classes required.
        - [D] Drafts a policy for coining URIs for properties and classes.
        - [D] Chooses "hash"- or "slash"-based URI patterns based on requirements.
    - [C] 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).
    - [C] 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.
    - [C] 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.
    - [C] 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.
    - [C] 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").
    - [C] 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.
    - [C] 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.
    - [C] Knows the naming conventions for RDF properties and classes.
- [B] Maintaining RDF vocabularies
    - [C] Understands policy options for persistence guarantees.
        - [D] Can draft a persistence policy.

- [B] Versioning RDF vocabularies
    - [C] Knows technical options for the form, content, and granularity of versions.
    - [C] Understands the trade-offs between publishing RDF vocabularies in periodic, numbered releases versus more continual or incremental approaches.
        - [D] Can express and justify a versioning policy.
- [B] Publishing RDF vocabularies
    - [C] Understands the typical publication formats for RDF vocabularies and their relative advantages
    - [C] Understands the purpose of publishing RDF vocabularies in multiple formats using content negotiation.
    - [C] Understands that to be "dereferencable", a URI should be usable to retrieve a representation of the resource it identifies.
        - [D] Ensures that when dereferenced by a Web browser, a URI returns a representation of the resource in human-readable HTML.
        - [D] Ensures that when dereferenced by an RDF application, a URI returns representation of the resource in the requested RDF serialization syntax.
- [B] Mapping RDF vocabularies
    - [C] 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.
    - [C] Understands that owl:equivalentProperty and owl:equivalentClass may be used when equivalencies between properties or between classes are exact.
    - [C] Recognizes that owl:sameAs, while popular as a mapping property, has strong formal semantics that can entail unintended inferences.
- [B] RDF application profiles
    - [C] Identifies real-world entities in an application domain as candidates for RDF classes.
    - [C] Identifies resource attributes and relationships between domain entities as candidates for RDF properties.
    - [C] Investigates how others have modeled the same or similar application domains.
        - [D] Communicates a domain model with words and diagrams.
        - [D] Participates in the social process of developing application profiles.

[A] Creating and transforming Linked Data

- [B] Managing identifiers (URI)
    - [C] 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.
    - [C] Understands trade-offs between "opaque" URIs and URIs using version numbers, server names, dates, application-specific file extensions, query strings or other

    obsoletable context.
  - [C] Recognizes the desirability of a published namespace policy describing an institution's commitment to the persistence and semantic stability of important URIs.

- [B] Creating RDF data

  - [C] Generates RDF data from non-RDF sources.
  - [C] Knows methods for generating RDF data from tabular data in formats such as Comma-Separated Values (CSV).
  - [C] 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.
- [B] Versioning RDF data
- [B] RDF data provenance

- [B] Cleaning and reconciling RDF data

  - [C] Cleans a dataset by finding and correcting errors, removing duplicates and unwanted data.

- [B] Mapping and enriching RDF data

  - [C] Uses available resources for named entity recognition, extraction, and reconciliation.

[A] Interacting with RDF data

- [B] Finding RDF data

  - [C] Knows relevant resources for discovering existing Linked Data datasets.
  - [C] Retrieves and accesses RDF data from the "open Web".
  - [C] Monitors and updates lists which report the status of SPARQL endpoints.
  - [C] Uses available vocabularies for dataset description to support their discovery.
  - [C] Registers datasets with relevant services for discovery.

- [B] Processing RDF data using programming languages.

  - [C] 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.

    - [D] Uses an RDF programming library to serialize RDF data in available syntaxes.
    - [D] Uses RDF-specific programming methods to iterate over components of RDF data.
    - [D] Uses RDF-library-specific convenience representations for common RDF vocabularies such as RDF, Dublin Core, and SKOS.

  - [C] Programatically associates namespaces to prefixes for use in serializing RDF or when parsing SPARQL queries.

    - [D] Uses RDF programming libraries to extract RDF data from CSV files, databases, or web pages.
    - [D] Uses RDF programming libraries to persistently stores triples in memory, on disk, or to interact with triple stores.
    - [D] Programatically infers triples using custom functions or methods.

    - [C] Understands how the pattern matching of SPARQL queries can be expressed using functionally equivalent constructs in RDF programming libraries.
        - [D] Uses RDF-specific programming methods to query RDF data and save the results for further processing.
        - [D] Uses utilities and convenience functions the provide shortcuts for frequently used patterns, such as matching the multiple label properties used in real data.
        - [D] Uses RDF libraries to process various types of SPARQL query result.

- [B] Querying RDF data
    - [C] Understands that a SPARQL query matches an RDF graph against a pattern of triples with fixed and variable values.
    - [C] Understands the basic syntax of a SPARQL query.
        - [D] Uses angle brackets for delimiting URIs.
        - [D] Uses question marks for indicating variables.
        - [D] Uses PREFIX for base URIs.
    - [C] Demonstrates a working knowledge of the forms and uses of SPARQL result sets (SELECT, CONSTRUCT, DESCRIBE, and ASK).
        - [D] Uses the SELECT clause to identify the variables to appear in a table of query results.
        - [D] Uses the WHERE clause to provide the graph pattern to match against the graph data.
        - [D] Uses variables in SELECT and WHERE clauses to yield a table of results.
        - [D] Uses ASK for a True/False result test for a match to a query pattern.
        - [D] Uses DESCRIBE to extract a single graph containing RDF data about resources.
        - [D] Uses CONSTRUCT to extract and transform results into a single RDF graph specified by a graph template.
        - [D] Uses FROM to formulate queries with URLs and local files.
    - [C] Understands how to combine and filter graph patterns using operators such as UNION, OPTIONAL, FILTER, and MINUS.
        - [D] Uses UNION to formulate queries with multiple possible graph patterns.
        - [D] Uses OPTIONAL to formulate queries to return the values of optional variables when available.
        - [D] Uses FILTER to formulates queries that eliminate solutions from a result set.
        - [D] Uses NOT EXISTS to limit whether a given graph pattern exists in the data.
        - [D] Uses MINUS to remove matches from a result based on the evaluation of two patterns.
        - [D] Uses NOT IN to restrict a variable to not being in a given set of values.
    - [C] Understands the major SPARQL result set modifiers, e.g., to limit or sort results, or to return distinct results only once.
        - [D] Uses ORDER BY to define ordering conditions by variable, function call, or expression.
        - [D] Uses DISTINCT to ensure solutions in the sequence are unique.
        - [D] Uses OFFSET to control where the solutions processed start in the overall sequence of solutions.

        - [D] Uses LIMIT to restrict the number of solutions processed for query results.
        - [D] Uses projection to transform a solution sequence into one involving only a subset of the variables.

    - [C] Understands the use of SPARQL functions and operators.

        - [D] Uses the regular expression (regex()) function for string matching.
        - [D] Uses aggregates to apply expressions over groups of solutions (GROUP BY, COUNT, SUM, AVG, MIN) for partitioning results, evaluating projections, and filtering.
        - [D] Uses the lang() function to return the language tag of an RDF literal.
        - [D] Uses the langMatches() function to match a language tag against a language range.
        - [D] Uses the xsd:decimal(expn) function to convert an expression to an integer.
        - [D] Uses the GROUP BY clause to transforms a result set so that only one row will appear for each unique set of grouping variables.
        - [D] Uses the HAVING clause to apply a filter to the result set after grouping.

    - [C] Differentiates between a Default Graph and a Named Graph, and formulates queries using the GRAPH clause.

        - [D] Formulates advanced queries using FROM NAMED and GRAPH on local data.
        - [D] Formulates advanced queries using FROM NAMED on remote data.
        - [D] Formulates advanced queries on data containing blank nodes.
        - [D] Formulates advanced queries using subqueries.
    - [C] Uses a temporary variable to extend a query.
    - [C] Understands the role of Property Paths and how they are formed by combining predicates with regular expression-like operators.

    - [C] Understands the concept of Federated Searches.

        - [D] Formulates advanced queries on a remote SPARQL endpoint using the SERVICE directive.
        - [D] Uses federated query to query over a local graph store and one or more other SPARQL endpoints.
        - [D] Pulls data from a different SPARQL endpoints in one single query using the SERVICE directive.
    - [C] Converts/manipulates SPARQL query outputs (RDF-XML, JSON) to the exact format required by a third party tools and APIs.
    - [C] Reads and understands high-level descriptions of the classes and properties of a dataset in order to write queries.

    - [C] Uses available tools, servers, and endpoints to issue queries against a dataset.

        - [D] Execute SPARQL queries using the Jena ARQ command-line utility.
        - [D] Queries multiple local data files using ARQ.
        - [D] Uses ARQ to evaluate queries on local data.
        - [D] Uses Fuseki server to evaluate queries on a dataset.
        - [D] Queries multiple data files using Fuseki.
        - [D] Accesses DBPedia's SNORQL/SPARQL endpoint and issues simple queries.

- [B] Visualizing RDF data

    - [C] Uses publicly available tools to visualize data.

        - [D] Uses Google FusionTables to create maps and charts.
    - [C] Distills results taken from large datasets so that visualizations are human-friendly.
    - [C] Converts/manipulates SPARQL query outputs (RDF-XML, JSON) to the exact format required by third party tools and APIs.

- [B] Reasoning over RDF data

    - [C] Understands the principles and practice of inferencing.
    - [C] Uses common entailment regimes and understands their uses.
    - [C] Understands the role of formally declared domains and ranges for inferencing.
    - [C] Understands how reasoning can be used for integrating diverse datasets.
    - [C] 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.
    - [C] Understands that OWL Full supports all available constructs and is most appropriately used when reasoning performance is not a concern.
- [B] Assessing RDF data quality

- [B] RDF data analytics

    - [C] Uses available ontology browsing tools to explore the ontologies used in a particular dataset.

- [B] Manipulating RDF data

    - [C] Knows the SPARQL 1.1 Update language for updating, creating, and removing RDF graphs in a Graph Store

        - [D] Uses INSERT/DELETE to update triples.
        - [D] Uses a CONSTRUCT query to preview changes before executing an INSERT/DELETE operation.

    - [C] Knows the SPARQL 1.1 Graph Store HTTP protocol for updating graphs on a web server (in "restful" style).

        - [D] Uses GET to retrieve triples from a default graph or a named graph.
        - [D] Uses PUT to insert set of triples into a new graph (or replace an existing graph).
        - [D] Uses DELETE to remove a graph.
        - [D] Uses POST to add triples to an existing graph.
        - [D] Uses proper syntax to request specific media types, such as Turtle.

    - [C] Understands the difference between SQL query language (which operates on database tables) and SPARQL (which operates on RDF graphs).

[A] Creating Linked Data applications

- [B] Storing RDF data