inBloom

Learning Map and inBloom Index System

Data Model and Server

inBloom Index Data Model
Principles and Requirements

V0.9 22 Jan 2013

Applied Minds, LLC

1209 Grand Central Ave Glendale CA 91201

Contents

Background

The Learning Map Data Model

The Bill & Melinda Gates Foundation (the Foundation) supports the implementation of the Common Core State Standards for US K-12 education. The Foundation awarded a contract to Applied Minds, LLC (AMI) to develop a Learning Map Data Model (LMDM) with the goal of it becoming a standard for educational technology infrastructure. The Learning Map will provide the organizing framework that maps the relationship between learning objectives, including dependencies and higher level groupings. It will also allow educational media resources, such as courses, books and web content to be linked to learning objectives. Curricula aligned to the standards will exhibit great diversity in highlighting paths through the Learning Map, and a suite of tools will allow authoring and visualization of the Learning Map. We believe that this data model will eventually enable the creation of online learning tools that are more responsive to the individual needs of a student. The LMDM is inspired by and based on the philosophy of the more general Knowledge Web (See: http://edge.org/conversation/aristotle-the-knowledge-web).

inBloom Technology

The Foundation has, in collaboration with the Carnegie Corporation, initiated an ambitious effort, Shared Learning Infrastructure (SLI), now inBloom Technology, to provide a new technology infrastructure that supports the Common Core Standards and the Foundation's vision, to be implemented by inBloom. AMI has been contracted by inBloom to build an implementation of a Learning Map and Learning Registry System, suitable for third-party software developers to populate content and develop applications.

InBloom Index Data Model and API Specification

As part of this effort, AMI developed a data model and API specification for the inBloom Index. This document presents the principles and requirements for the use of the inBloom Index Data Model by software designers and developers. In keeping with the labeling of the API, this document is version 0.9, to indicate that it is a developer resource.

inBloom Index Data Model Principles and Requirements

Introduction

The InBloom Index is expected to hold and make searchable most, if not all, of the metadata (including paradata) and structured data associated with the inBloom's Shared Learning Initiative (SLI).

Data sources including the Learning Registry (LR) and other data systems within the inBloom Technology as well as external sources may be used by the SLC and its partners to populate the inBloom Index.

The Data Model for the inBloom Index has been developed with the goal of clean integration of a large number of data sources in a manner which supports search, access, and comparison of data. The intention is to provide a strong foundation that is easy for data developers to model their data sets for inclusion and modify as needed in the future.

Data Model Principles

All data that is intended to be stored in the inBloom Index must exist in a `strongly structured` form. Strongly structured data is defined to have the following qualities:

- All data should exist in structures having well defined boundaries. These structures are known as data entities.
- References to data entities must be unambiguous. (i.e., a reference to an entity only refer to a single instance.)
- All references must be locally explicit and "context independent". For example, the implicit context intended by of "Third item in an array" or "the current time and date" are not locally explicit, are thus their use as a disambiguator in a reference must be avoided.
- The data in an entity must be logically self-consistent.
- Data structures must be defined by schema which are declared, explicit, and exist in a machine interpretable form.

Developers may use any data model that results in strongly structured data. Furthermore, in order to assure that data are properly indexed by and findable in the inBloom Index, developers should create data schemata that are compatible with and follow the specific design of the inBloom Index Meta-Model.

The inBloom Index Meta-Model

The inBloom Index meta-model is a "model of models" for data structures. It is used as a set of rules and guidelines for creating schema for data sets. Philosophically and logically it is a mix of:

- Typed object/entity design (as found in object-oriented languages), and
- Graph model design (as found in tuple/graph stores).

The design of the inBloom Index Meta-Model is partially based on the meta-model from `schema.org`. A good starting place for schema designers is to describe their schemata in the "language" of `schema.org`, as this goes far in the process of making them ready for the inBloom Index.

Requirements from `schema.org`

Some of the features of the `schema.org` meta-model that are particularly useful include:

- All data structures are composed of distinct data `entities`, which contain a set of data `properties`.
- Each entity type contains a set of property types (its schema).
- An entity's schema must contain the one or more `types` (i.e., thus declaring the entity to be an `instance` of that type, as in an object oriented language).
- Instances of an entity type are likely to have properties that are included in the schema for that type.
- Entities do not have to be of a particular type in order to use properties from that type's schema.
- A property type defines a set of explicit `ranges` or "expected entity/data types" for instances of that property. Ranges are chosen from a set of literal data types or existing entity types.
- Entity types usually correspond to types of "real world" entities (e.g., person, document, organization, location) or are simply convenient data structure abstractions (e.g., postal address, nutritional information, geographic coordinate region).
- Entity types exist in a non-exclusive super/subtype hierarchy.

Additional Requirements

While `schema.org` is an excellent basis for a meta-model, the inBloom Index requires some additional rigor to create a strongly structured repository of indexable data. These requirements include:

- The elements of a schema are treated as entities, just like any other. i.e., `Entity types` and `property types` are instances of the "type of types".

- `Example:` The entity type `person` is an instance of the type "`entity types`". The property type `gender` that is part of the person schema is an instance of the type "`property types`".

- In general, choices of property enumeration members should be modeled as instances of a type that are members of that enumeration. Thus, a property type that is an enumeration should have a range that is an entity type rather than a literal data type.
  - `Example:` The `gender` property of a person should be chosen from instances of the `gender` property type, such as "male", "female", "other".
- Unless restrictions are explicitly declared in the schema, entities may have multiple property instances of any given property type. Unlike `schema.org`, the plurality of the name of the property type is not used to indicate cardinality restrictions.
  - `Example:` A `person` type may more than one `sibling` property instance.
- Unless declared otherwise in the schema, a property is considered optional.
  - `Example:` A `person` type may have `child` property instances (optional), and a `person` must have a `first name` property instance (mandatory).
- The choice of range for a property type should follow a preference order from most to least constrained. In practical terms, the following kinds of ranges are ordered from most to least desirable:
  - An enumeration entity type (e.g., The `gender` type has "male", "female", and "other" entity instances.)
  - Any other entity type (e.g., `person`)
  - A specialized literal type (e.g., timestamp, URL)
  - A numeric literal (e.g., integer, float)
  - A UTF-8 string
- Every entity must have one or more fully qualified globally unique identifier (GUID) properties.
  - `Example:` The language "Urdu" has the GUID `urd` as defined by the ISO 639-3 standard.
- Known broken schemata in the core of `schema.org` are not followed.
  - `Example:` The `e-mail address` property for an entity of type `person` in `schema.org` exists in multiple parts of the schema, with no usage preference order described.

Globally Unique Identifiers (GUIDs)

Possibly the most important element of the inBloom Index Meta-Model is the requirement and proper use of GUIDs, which are required to have the following qualities:

- An entity must have at least one GUID before the inBloom Index can index it. It is strongly recommended that the data source (creator) assign this first GUID. This provides multiple benefits:
  - A data entity can always be unambiguously found without manual assistance.
  - Two entities with very incomplete data can still be reconciled.
  - Entity reconciliation becomes much cheaper.
  - Different versions of the same entity can easily be distinguished.
- A GUID must be fully qualified. i.e., its containing `namespace` must be explicit.
  - `Example:` The 1998 paperback edition ("manifestation") of the book *The Society of Mind* (by Marvin Minsky) has the GUID `978-0671657130` in the ISBN-13 namespace.
- A GUID must be expressible in Universal Resource Name (URN) format as per Internet Engineering Task Force (IETF) Request for Comment (RFC) `RFC 2141`.
  - `Example: urn:isbn-13:978-0671657130`
- Two (apparent) entities with the same GUID are the same entity. Thus, a GUID must **NEVER** refer to more than one entity.
  - `Note:` The use of URN syntax for GUIDs means that namespaces are explicit. A helpful way to insure GUID uniqueness is to claim and use a unique (informal) namespace based on the name of the assigning entity; e.g., inBloom could choose the informal namespace `inbloom.org`
  - `Example: urn:isbn-10:0465026850` refers only to the 1979 hardcover edition of *Gödel, Escher, Bach: An Eternal Golden Braid* (by Douglas Hofstadter), but not to the 1989 paperback edition, which is `urn:isbn-10:0394756827` . These are considered to be distinct entities. It is not allowed for any GUID to refer to both editions. Thus, typical URLs are probably a poor choice as a GUID since they refer to many entities over time (i.e., different versions of a Web page).
- Two (apparent) entities with different GUIDs may actually be the same entity or may be different entities.
  - `Example: urn:isbn-13:978-0671657130` and `urn:isbn-10:0671657135` both refer to the same edition of *The Society Of Mind*.