Data Collection and MOdeling

Course 4 - Data Exchange Formats

Definition

“Data exchange is the process of taking data structured under a source schema and transforming it into a target schema, so that the target data is an accurate representation of the source data” (Principles of Data Integration)

“Data exchange is the mutual transfer of information between people and devices. This includes the sending of documents or photos. Data exchange increasingly takes place in electronic form. The term comes from the field of data processing. The exchange of data is a part of data processing and is standardized. This means that certain standards are defined and formats are used. These standards serve to make data exchange efficient and uniform worldwide.” (TeamDrive)

Data Exchange Formats

- API (Application Programming Interface): web services over HTTP; SOAP, REST, GraphQL (a later course deals with this topic)
- ETL (Extract, Transform, Load): data is transferred by direct database connection
- **File Transfer**: files are transferred (using various formats) and then loaded for processing
- RPC (Remote Procedure Call): a program call launches the execution of a procedure on another address space (even remote computer)
- Message Brokers/Events Server: data is transferred by a delivery service
- Data Streaming: several data sources send data continuously to a process that ingests it in sequence

Data Heterogeneity

- Technical: Data exchange format
- Syntactical: Character encoding

Open Standards for Exchange Data

- Formats: describe structure of data
- Data types: how values are expressed
- Data transfers: define rules on exchanging data or providing access
- Rules: describe what data to share, formats, schemas
- Maps: describes how models are expressed using data exchange formats

Data Formats

- Text Based
  - CSV
  - XML
  - JSON
  - YAML
  - RDF
  - Plain Text
  - Specialized formats: SVG, GML, WKT, KML
- Binary Based
  - CORBA (CDR)
  - BSON
  - Protocol Buffers (ProtoBuf)
  - Big Data: Avro, Parquet, Thrift
  - Specialized formats: WKB

Text Based

XML

Simple text-based format for structured data; derived from SGML; extensible

One of most often used format for sharing structured data; self-descriptive

Doesn’t use predefined tags; structure and tags must be defined

Hardware and software independent format

Separates data from presentation

W3C Recommendation

XML

Documents formatted as element trees

A document has a root element

Relationships between elements could be: parent, child, sibling

There is a (optional) prolog that defines the version and character encoding

Elements must have a closing tag; tags are case sensitive; should be properly nested

Elements can have: text, attributes, elements

XML

Well Formed Documents:

- must have a root
- elements must have a closing tag
- tags are case sensitive
- elements must be properly nested
- attribute values must be quoted

Valid Document:

- Well formed
- conform to Document Type Definition
  - DTD: structure, elements and attributes for XML documents
  - XML Schema (alternative to DTD)

XML

Browser usually access XML documents by loading them in an XML DOM object by means of an XML Parser

XML can be displayed using XSLT

XQuery is used to find and extract attributes and elements from XML

XSLT uses XPath to find information in an XML document

XQuery tutorial

XPath tutorial

XML

```
<?xml version="1.0"?>
<!DOCTYPE cars [
  <!ELEMENT car     (make,model,type,price)>
  <!ELEMENT make    (#PCDATA)>
  <!ELEMENT model   (#PCDATA)>
  <!ELEMENT type (sedan, coupe, hatchback)>
  <!ELEMENT price   (#PCDATA)>
]>
<car>
  <make>Dacia</make>
  <model>Spring</model>
  <type>hatchback</type>
  <price>10.000</price>
</car>
```

```
<?xml version="1.0"?>
  <!ELEMENT car     (make,model,type,price)>
  <!ELEMENT make    (#PCDATA)>
  <!ELEMENT model   (#PCDATA)>
  <!ELEMENT type (sedan, coupe, hatchback)>
  <!ELEMENT price   (#PCDATA)>
```

```
<?xml version="1.0"?>
<!DOCTYPE cars SYSTEM "cars.dtd">
<car>
  <make>Dacia</make>
  <model>Spring</model>
  <type>hatchback</type>
  <price>10.000</price>
</car>
```

JSON

JavaScript Object Notation is a self-describing text format for storing and transporting data

Easy conversion between JavaString object to JSON string and vice-versa

Syntax with few rules

Hierarchical structure

Has to be parsed

JSON

- Data represented in name-value pairs
- Data is separated by comma
- Objects are enclosed by curly braces
- Arrays are enclosed within square brackets

Keys must be strings
Values must fall under one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

JSON

JSON is just a string with a specified format; contains only properties, no methods

Double quotes are required for string values and keys

JSON can be validated against errors (if any, will break the application) JSONLint

JSON is mainly used for data exchange and applications minify it to save bandwidth. This doesn't change the data in the JSON file, but it makes the JSON code unreadable. In order to make it readable again a JSON Formatter can be used

JSON

```
{
  "car": {
          "make": "Dacia",
          "model": "Spring",
          "type": "hatchback",
          "price": 10000
  }
}
```

YAML

Is a human-friendly data serialization language for all programming languages, but also often used for writing configuration files.

May stand for "yet another markup language" or "YAML ain’t markup language" (a recursive acronym), emphasizing that YAML is for data, not documents.

Has features that come from Perl, C, XML, HTML, and other programming languages. Is also a superset of JSON, JSON files being valid in YAML.

YAML

Uses Python-style for indentation to indicate nested elements.

Tab characters are not allowed, whitespaces being used instead.

There are no formatting symbols, such as braces, square brackets, closing tags, or quotation marks (except for abbreviated forms of lists[] or dictionaries {}).

YAML files use a .yml or .yaml extension.

YAML

The structure of a YAML file is a map or a list.

A map is formed by key-value pairs. Each key must be unique.

A map needs to be resolved before it can be closed, and a new map is created.

A map can be created by increasing the indentation level or by resolving the previous map.

A list includes values listed in a specific order and may contain any number of items.

A list sequence starts with a dash (-) and a space, indentation separates it from the parent

A list can be embedded into a map.

YAML

YAML also can contain scalars (arbitrary data encoded in Unicode) that can be used as values such as strings, integers, dates, numbers, or booleans.

Null values can be specified by “null” or “~”

Booleans can be defined using “True”, “False”, “On”, “Off”

A file starts with three dashes (-). Dashes indicate the start of a new YAML document.

YAML supports multiple documents, compliant parsers recognizing each row of 3 dashes as the beginning of a new one.

YAML

```
car:
  make: Dacia
  model: Spring
  type: hatchback
  price: 10000
```

RDF

Resource Description Framework is a model to represent data about physical objects and abstract concepts.

Uses a graph format to express relations between entities.

Modeled concepts are considered resources. Information is represented using statements in the format:

```
<subject> <predicate> <object>
```

The expressed relation is between the subject and the object, both being considered resources. RDF statements are also known as triples.

RDF

The same resource could be used in several triples with different roles. In this way, connections between resources can be expressed.

RDFs defined by IRIs and literals

- IRI - URI expressed using UNICODE; can be used in all positions in the triple
- Literal - basic value: string, date, number, etc; can be used only in object part of the triple

RDF

```
/* triples as < subject, predicate, object > */

<root, car, make>

<make, type, Dacia>

<root, car, model>

<model, type, Spring>

<root, car, type>

<type, type, hatchback>

<root, car, price>

<price, type, 10000>
```

SVG

Stands for Scalable Vector Graphics

Is used to define vector-based graphics for the Web

Defines the graphics in XML format

Elements and attributes can be animated

Is a W3C recommendation

Integrates with W3C standards like DOM and XSL

SVG

Images can be created/edited with any text editor

Images can be searched, indexed, scripted, and compressed

Images are scalable, zoomable

Graphics don’t lose any quality if they are zoomed or resized

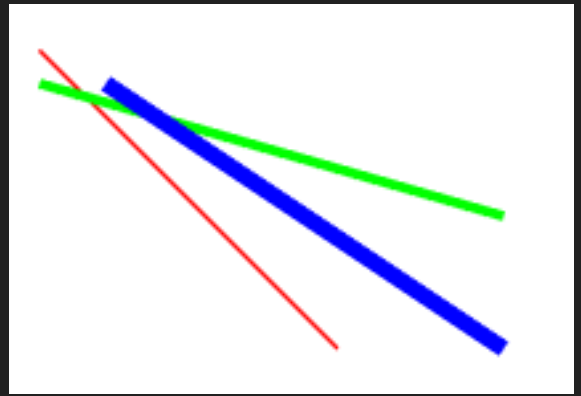
SVG is an open standard

SVG files are pure XML

Can be converted to binary image formats (PNG, JPG, PDF)

SVG

```
<svg width="160" height="160">
  <line x1="10" y1="10" x2="100" y2="100" style="stroke:#f00; stroke-width:1"/>
  <line x1="10" y1="20" x2="150" y2="60"  style="stroke:#0f0; stroke-width:3"/>
  <line x1="30" y1="20" x2="150" y2="100" style="stroke:#00f; stroke-width:5"/>
</svg>
```



Binary Based

BSON

Binary Javascript Object Notation is a binary encoded serialization of JSON documents.

Ads optional non-JSON-native data types, like dates and binary data.

Compared to other binary formats, like Protocol Buffers, BSON is more "Schema-less" than Protocol Buffers, providing the advantage of flexibility at a slight disadvantage of space efficiency.

BSON

Type	Description
byte	1 byte (8 bits)
int32	32 bits signed integer
int64	64 bits signed integer
decimal128	16 bytes decimal floating point
date	64 bits integer
objectId	12 bytes composed field
array	Data type dependent storage

BSON offers additional data types compared to JSON; decimal128 is used for high precision decimal representation (in financial and trading applications)

BSON is lightweight, large documents being stored and transferred easily

It is efficient as the stored data is prefixed with length fields and array indices where is the case

Protocol Buffers

Used (Google) for storing and interchanging structured information.

Serves as a basis for a custom remote procedure call (RPC) system that is used for inter-machine communication at Google.

- language-neutral
- platform-neutral
- extensible

It is smaller, faster, and simpler than XML.

Data structure is defined once, then special generated source code can be used to easily write and read structured data using a variety of languages.

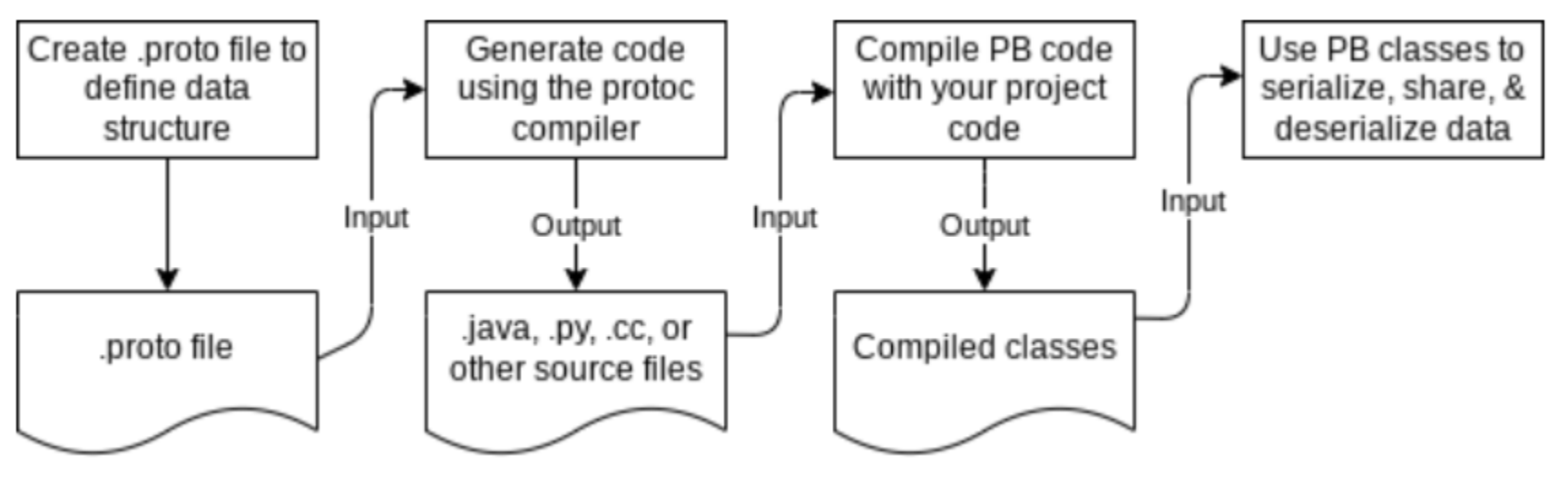
Protocol Buffers

Some advantages of protocol buffers:

- Compact data storage
- Fast parsing
- Availability in many programming languages
- Optimized functionality through automatically-generated classes

A messages can be read by code written in any supported programming language. A Java program on one platform can produce data from one software system, serialize it based on a .proto definition, and then another application in Python, running on a different platform, can extract specific values from that serialized data.

Protocol Buffers



Protobuf workflow

Further reading

- JSON vs XML
- XML