

Data Formats

UA.DETI.CBD

José Luis Oliveira / Carlos Costa

Outline

- ❖ We will look into several formats for **encoding data**
- ❖ **CSV** – Comma-Separated Values
- ❖ **XML** – Extensible Markup Language
- ❖ **JSON** – JavaScript Object Notation
- ❖ **BSON** – Binary JSON
- ❖ **RDF** – Resource Description Framework
- ❖ **Protocol Buffers**

web semantic

↓

Data encoding

- ❖ Software applications inevitably **change** over time.
 - In most cases, this also requires a change to data.
 - Old and new versions of the code, and old and new data formats, may potentially all coexist in the system at the same time.
- ❖ For the system to continue running smoothly, we need to maintain compatibility in both directions:
 - **Backward compatibility** - newer code can read data that was written by older code.
 - **Forward compatibility** - older code can read data that was written by newer code. It requires older code to ignore additions made by a newer version of the code.



Data encoding

- ❖ Programs usually work with data ...
 - In memory, data is kept in objects, structs, lists, arrays, hash tables, trees and so on.
 - Out of memory, to write data to a file, or send it over the network (i.e. a different sequence of bytes).
- ❖ The translation from the in-memory representation to a byte sequence is called **encoding** (also known as **serialization** or marshalling),
- ❖ The reverse is called **decoding** (parsing, **deserialization**, unmarshalling).

Language-specific formats

- ❖ Many programming languages come with built-in support for encoding in-memory objects into byte sequences.
 - Java (*Serializable*), Ruby (*Marshal*), Python (*pickle*), ...
- ❖ These encoding libraries are very convenient, but...
- ❖ ... reading the data in another language is very difficult.
 - using such kind of encoding commits to the current programming language.
- ❖ So, it is a bad idea to use these built-in encoding for anything other than **transient purposes**.

Textual Formats

- ❖ Main advantage: human-readable
 - Examples: CSV, JSON, XML and RDF
- ❖ But they bring some issues:
- ❖ Ambiguity between a number and a string
 - JSON handles this, but not integers # floating-point, i.e. lacks to specify precision.
- ❖ CSV does not have any schema
 - It is up to the application to define the meaning of each row and column.
- ❖ Despite some flaws, JSON, XML and CSV are good enough for many purposes.

Binary Encoding

- ❖ Binary encoding
- ❖ More compact, faster to parse.
 - For a small dataset, the gains are negligible, but once you get into the terabytes, the choice of data format can have a big impact.
- ❖ Some binary encodings for JSON
 - MessagePack, BSON, BSON, UBJSON, BJSON, and Smile, ...
- ❖ But none of them is as widely adopted as the textual versions of JSON and XML.

- ❖ **CSV – Comma-Separated Values**
- ❖ XML – Extensible Markup Language
- ❖ JSON – JavaScript Object Notation
- ❖ BSON – Binary JSON
- ❖ RDF – Resource Description Framework
- ❖ Protocol Buffers

CSV – Comma-Separated Values

- ❖ Unfortunately not fully standardized
 - Different field separators (commas, semicolons)
 - Different escaping sequences
 - No encoding information
- ❖ RFC 4180, RFC 7111
 - URI Fragment Identifiers for the text/csv Media Type
- ❖ File extension: *.csv
- ❖ Media type (MIME)
 - Content type: text/csv

Example

❖ Document

- A header line (optional) + records

firstname,lastname,year

Ana,Katrina,1974

Paul,Machado,1956

Luis,Morais,1974

Sofia,Silvasky,1986

Maria,Marinova,1976

XML

- ❖ CSV – Comma-Separated Values
- ❖ **XML – Extensible Markup Language**
- ❖ JSON – JavaScript Object Notation
- ❖ BSON – Binary JSON
- ❖ RDF – Resource Description Framework
- ❖ Protocol Buffers

XML – Extensible Markup Language

- ❖ Representation of semi-structured data
 - + a family of related technologies, languages, specifications, ...
- ❖ Derived from SGML, developed by W3C, since 1996
- ❖ Design goals
 - Simplicity, generality and usability across the Internet
- ❖ File extension: *.xml, content type: text/xml
- ❖ Versions: 1.0 and 1.1
- ❖ W3C recommendation
 - <http://www.w3.org/TR/xml11/>
- ❖ XML formats = particular languages
 - **XSD**, XSLT, XHTML, DocBook, ePUB, SVG, RSS, SOAP, ...

Example

```
<?xml version="1.1" encoding="UTF-8"?>
<movie year="2007">
  <title>The Great Marnoto</title>
  <actors>
    <actor>
      <firstname>Jakim</firstname>
      <lastname>Dalmeida</lastname>
    </actor>
    <actor>
      <firstname>Sofia</firstname>
      <lastname>Ravara</lastname>
    </actor>
  </actors>
  <director>
    <firstname>Paulo</firstname>
    <lastname>Castanho</lastname>
  </director>
</movie>
```

Constructs – Element

- ❖ Marked using opening and closing tags
 - ... or an abbreviated tag in case of empty elements
- ❖ Each element can be associated with a set of attributes
- ❖ Well-formedness is required
- ❖ Types of content
 - Empty content
 - Text content
 - Element content
 - Sequence of nested elements
 - Mixed content
 - Elements arbitrarily interleaved with text

Constructs – Attribute

- ❖ Name-value pair
- ❖ Escaping sequences (predefined entities)
 - Used within values of attributes or text content of elements
 - E.g.: `<` for `<`, `>` for `>`, `"` for `"`, ...
- ❖ All available XML constructs
 - Basic: element, attribute, text
 - Other: comment, processing instruction, ...

JSON

- ❖ CSV – Comma-Separated Values
- ❖ XML – Extensible Markup Language
- ❖ **JSON – JavaScript Object Notation**
- ❖ BSON – Binary JSON
- ❖ RDF – Resource Description Framework
- ❖ Protocol Buffers

JSON – JavaScript Object Notation

Assim como o XML é hierárquico

- ❖ Open standard for data interchange
- ❖ Design goals
 - Simplicity: text-based, easy to read and write
 - Universality: object and array data structures
 - Supported by majority of modern programming languages
- ❖ Derived from JavaScript (but language independent)
- ❖ Started in 2002
- ❖ File extension: *.json
- ❖ Content type: application/json
- ❖ <http://www.json.org/>

JSON structure

❖ JSON is built on two structures:

❖ A **collection of name/value pairs**.

- In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.

❖ An **ordered list** of values.

- In most languages, this is realized as an *array*, vector, list, or sequence.

Example

```
{  
  "title": "The Great Marnoto",  
  "year": 2007,  
  "actors": [  
    {  
      "firstname": "Jakim",  
      "lastname": "Dalmeida"  
    },  
    {  
      "firstname": "Sofia",  
      "lastname": "Ravara"  
    }  
  ],  
  "director": {  
    "firstname": "Paulo",  
    "lastname": "Castanho"  
  }  
}
```

Data Structure – Object

- ❖ Unordered collection of name-value pairs (properties)
 - Correspond to structures such as objects, records, structs, dictionaries, hash tables, keyed lists, associative arrays, ...

❖ Example

```
{ "name" : "Manuel Sliav", "year" : 2000 }  
{ }
```

Data Structure – Array

- ❖ Ordered collection of values
 - Correspond to structures such as arrays, vectors, lists, sequences, ...
- ❖ Values can be of different types, duplicate values are allowed
- ❖ Example

```
[ 2, 7, 7, 5 ]
```

```
[ "Some person", 1979, 77 ]
```

```
[ ]
```

Data Structure – Value

- ❖ Unicode string
 - Enclosed with double quotes
 - Backslash escaping sequences
 - Example: "a \n b \" c \\ d"
- ❖ Number
 - Decimal integers or floats
 - Examples: 1, -0.5, 1.5e3
- ❖ Nested object
- ❖ Nested array
- ❖ Boolean value: true, false
- ❖ Missing information: null

```
{
  "stuff": {
    "onetype": [
      {"id":1,"name":"John"},
      {"id":2,"name":"Don"}
    ],
    "othertype":
      {"id":2,"company":"ACME"}
  },
  "otherstuff": {
    "thing": [[1,42],[2,2]]
  }
}
```

BSON

- ❖ CSV – Comma-Separated Values
- ❖ XML – Extensible Markup Language
- ❖ JSON – JavaScript Object Notation
- ❖ **BSON – Binary JSON**
- ❖ RDF – Resource Description Framework
- ❖ Protocol Buffers

BSON – Binary JSON

- ❖ Binary-encoded serialization of JSON documents
 - Design characteristics: lightweight, **traversable**, efficient
 - **convenient storage of binary information:**
 - better suitable for exchanging **images** and **attachments**
 - designed for **fast in-memory manipulation**
 - **extra data types (then JSON):**
 - double, date, byte array, JavaScript code, ...
- ❖ **Used by MongoDB**
 - Document NoSQL database for JSON documents
 - Data storage and network transfer format
- ❖ File extension: *.bson
- ❖ <http://bsonspec.org/>

Example

❖ JSON

```
{  
  "title" : "Marnoto",  
  "year" : 2007  
}
```

❖ BSON

```
      t   i   t   l   e               M  
2200 0000 0274 6974 6c65 0008 0000 004d  
a r   n o   t o           y e   a r   2007    // = 0x07d7  
6172 6e6f 746f 0010 7965 6172 00d7 0700  
0000
```

Document Structure

❖ Document

- serialization of one JSON object or array

❖ JSON object is serialized directly

❖ JSON array is first transformed to a JSON object

- Property names correspond to position numbers , e.g.

`["Some", "Another"] → { "0" : "Some", "1" : "Another" }`

❖ Structure

- Document size (total number of bytes)
- Sequence of elements
- Terminating hexadecimal 00 byte

```

           t i t l e                               M
2200 0000 0274 6974 6c65 0008 0000 004d
a r n o t o           y e a r      2007 // = 0x07d7
6172 6e6f 746f 0010 7965 6172 00d7 0700
0000
```

Document Structure

❖ Element

- serialization of one JSON property

❖ Structure

- Type selector
 - 02 (string), 03 (object), 04 (array)
 - 01 (double), 10 (32-bit integer), 12 (64-bit integer)
 - 08 (boolean), 09 (datetime), 11 (timestamp)
 - 0A (null)
 - ...
- Property name
 - Unicode string terminated by 00
- Property value

```
                t i t l e                M
2200 0000 0274 6974 6c65 0008 0000 004d
a r n o t o y e a r 2007 // = 0x07d7
6172 6e6f 746f 0010 7965 6172 00d7 0700
0000
```

RDF

- ❖ CSV – Comma-Separated Values
- ❖ XML – Extensible Markup Language
- ❖ JSON – JavaScript Object Notation
- ❖ BSON – Binary JSON
- ❖ **RDF – Resource Description Framework**
- ❖ Protocol Buffers

RDF – Resource Description Framework

- ❖ Language for representing information about resources in the World Wide Web
 - + a family of technologies, languages, specifications, ...
 - Used in graph databases and in the context of the Semantic Web, Linked Data, ...
- ❖ Developed by W3C
 - Started in 1997
- ❖ Versions: 1.0 and 1.1
- ❖ W3C recommendations
 - <https://www.w3.org/TR/rdf11-concepts/>
 - Concepts and Abstract Syntax
 - <https://www.w3.org/TR/rdf11-mt/>
 - Semantics

Statements

- ❖ RDF is based on the concept that every **resources** can have different **properties** which have **values**.
- ❖ Resource - Any real-world entity
 - **Referents** = resources identified by IRI (Internationalized Resource Identifier)
 - E.g. physical things, documents, abstract concepts, ...
<http://db.pt/movies/Marnoto>
<http://db.pt/terms#actor>
<mailto:somegirl@nowhere.com>
<urn:issn:0167-6423>
 - **Values** = resources for literals
 - E.g. numbers, strings, ...

Statements

- ❖ Example of a **statement** about a web page:

<http://www.example.org/index.html> has an **author** whose name is Pete Maravich.

- ❖ A RDF statement is a **triple** that contains a:

- **Resource**, the **subject** of a statement
- **Property**, the **predicate** of a statement
- **Value**, the **object** of a statement

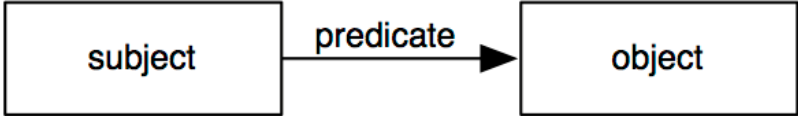
- ❖ Several properties for this web page could be:

<http://www.example.org/index.html> has an **author** whose name is Pete Maravich.

<http://www.example.org/index.html> has a **language** which is English.

<http://www.example.org/index.html> has a **title** which is Example_Title.

RDF Triple

Graphical form	
Triple	subject predicate object
Relational form	predicate(subject,object)
RDF/XML	<pre><rdf:Description rdf:about="subject"> <ex:predicate> <rdf:Description rdf:about="object"/> </ex:predicate> </rdf:Description></pre> <p><i>Handwritten notes in red:</i> - Next to <code><ex:predicate></code>: <i>Propriedade</i> - Next to <code>rdf:about="subject"</code>: <i>Recurso</i> - Next to <code>rdf:about="object"/></code>: <i>Valor</i></p>
Turtle	subject ex:predicate object.

RDF example

```
<?xml version="1.0"?><br>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:same="http://msome.lycos.com/elements/1.0/myschema#">
  <rdf:Description rdf:about="http://www.cio.com/archive/km.html">
    <same:title>Less for Success</same:title>
    <same:author>Alice Dragoon</same:author>
    <same:subject>
      <rdf:Bag>
        <rdf:li>knowledge management</rdf:li>
        <rdf:li>technology investments</rdf:li>
      </rdf:Bag>
    </same:subject>
    <same:format>text/html</same:format>
    <same:status>active</same:status>
    <same:created>2004-10-19</same:created>
    <same:funFactor>3</same:funFactor>
  </rdf:Description><br>
</rdf:RDF>
```

Serialization approaches

- ❖ RDF/XML notation
 - XML syntax for RDF (.rdf, .rdfs, .owl, .xml)
 - <https://www.w3.org/TR/rdf-syntax-grammar/>
- ❖ Turtle notation (Terse RDF Triple Language)
 - .ttl extension
 - <https://www.w3.org/TR/turtle/>
- ❖ N-Triples notation
 - .nt extension
 - <https://www.w3.org/TR/n-triples/>
- ❖ JSON-LD notation
 - JSON-based serialization for Linked Data
 - <https://www.w3.org/TR/json-ld/>

Turtle Notation

- ❖ Compact text format, various abbreviations for common usage patterns

- File extension: *.ttl
- Content type: text/turtle
- <https://www.w3.org/TR/turtle/>

- ❖ Example

```
@prefix i: <http://db.com/terms#>
```

```
@prefix m: <http://db.com/movies/>
```

```
@prefix a: <http://db.ccom/actors/>
```

```
m:Marnoto
```

```
  i:actor a:Dalmeida , a:Jakim ;
```

```
  i:year "2007" ;
```

```
  i:director [ i:firstname "Paulo" ; i:lastname "Castanho" ]
```

Protocol Buffers

- ❖ CSV – Comma-Separated Values
- ❖ XML – Extensible Markup Language
- ❖ JSON – JavaScript Object Notation
- ❖ BSON – Binary JSON
- ❖ RDF – Resource Description Framework
- ❖ **Protocol Buffers**

Protocol Buffers

- ❖ Protocol Buffers is a binary encoding library that require a **schema** for any data that is encoded.
- ❖ Extensible mechanism for serializing structured data
 - Used in communication protocols, data storage, ...
- ❖ Developed (and widely used) by Google
 - Mostly for server-side communication
- ❖ Design goals
 - Language-neutral, platform-neutral
 - **Small, fast, simple**
- ❖ File extension: *.proto
- ❖ <https://developers.google.com/protocol-buffers/>

Protocol Buffers

❖ Intended usage

- Schema creation
 - automatic source code generation
 - sending messages between applications

❖ Components

- Interface description language
- Source code generator (**protoc** compiler)
- Supported languages
 - Official: C++, C#, Java, Python, Ruby ...
 - 3rd party: Perl, PHP, Scala, ...
- Binary serialization format
- Compact, not self-describing

Schema: encoding examples

```
message Person {  
    required string user_name = 1;  
    optional int64 favorite_number = 2;  
    repeated string interests = 3;  
}
```

```
syntax = "proto3";  
message Actor {  
    string firstname = 1;  
    string lastname = 2;  
}  
message Movie {  
    string title = 1;  
    int32 year = 16;  
    repeated Actor actors = 17;  
    enum Genre {  
        UNKNOWN = 0;  
        COMEDY = 1;  
    }  
    repeated Genre genres = 2048;  
}
```

Example: ProtoBuf to Java

```
message Person {  
  required string name = 1;  
  required int32 id = 2;  
  optional string email = 3;  
}
```

protoc

Person builder

java

```
Person john = Person.newBuilder()  
    .setId(1234)  
    .setName("John Doe")  
    .setEmail("jdoe@example.com")  
    .build();  
output = new  
    FileOutputStream(args[0]);  
john.writeTo(output);
```

java

Summary

Data encoding formats:

- ❖ **Programming-language-specific** ...

- restricted to a single programming language.

- ❖ **Textual** formats

- widespread, and its compatibility depends on the use.
- Somewhat vague about datatypes, namely numbers and binary strings.

- ❖ **Binary** schema-driven formats

- More compact and efficient encoding, with clearly defined forward and backward compatibility semantics.
- The schema can be useful for documentation and code generation in statically typed languages.

Summary

Data formats

- ❖ **Relational:** CSV
- ❖ **Tree:** XML, JSON
- ❖ **Graph:** RDF
- ❖ **Binary:** BSON, Protocol Buffers

Other binary formats

❖ Avro

- Apache

❖ Thrift

- Facebook + Apache

❖ MessagePack

... and many others

❖ A good comparison is available at:

- https://en.wikipedia.org/wiki/Comparison_of_data_serialization_formats

Resources

- ❖ Martin Kleppmann, ***Designing Data-Intensive Applications***, O'Reilly Media, Inc., 2017.
- ❖ Pramod J Sadalage and Martin Fowler, ***NoSQL Distilled*** Addison-Wesley, 2012.
- ❖ Eric Redmond, Jim R. Wilson. ***Seven databases in seven weeks***, Pragmatic Bookshelf, 2012.
- ❖ Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, ***Database systems: the complete book*** (2nd Ed.), Pearson Education, 2009.