CptS 415 Big Data

JSON

Srini Badri

What is a JSON?

- JavaScript Object Notation (JSON) is a language-independent data format
- Originally developed for stateless communication between browser (client) and web server
- Widely used in different programming languages include JavaScript, Python, Java, C/C++

Example: JSON

- Book.json:

```
{
    "title": "Database Systems",
    "author": "C. J. Date",
    "year": 1995,
    "publisher": "Addison-Wesley"
}
```

name (key) value

JSON Data Format

- Data objects defined within { } (curly brackets)
- Data objects consists of a collection of name (key) - value pairs
- name (key) - value pairs:
  - the relationship is specific by a colon (:)
  - are separated by other name - value pairs by a comma (,)
  - name is of String data type and should be unique
  - value can be of Number, String, Boolean or Object data type

Data Types in JSON

- Name (Key):
  - String data type
  - Specified by double quotes ("")

- Value:
  - Number - integers and floating-point numbers
  - String - sequence of Unicode characters enclosed within double quotes ("")
  - Boolean - true / false values
  - Object - collection of name-value pairs enclosed within curly brackets ({ })
  - Array - list of elements enclosed within square brackets ([ ])

Flat vs Nested JSON

- Flat:
  - data objects are flat, and can be referenced directly

```
{
    "id": "001",
    "name": "Joe",
    "gpa": 3.0,
    "DB_course_no": "331",
    "DB_title": "DB",
    "DB_credit": 3.0
}
```

Flat vs Nested JSON

- Nested:
    - data objects are nested, and are referenced by their qualifying name

```
{
    "id": "001",
    "name": "Joe",
    "gpa": 3.0,
    "course_enrolled":
    {
        "course_no": "331",
        "title": "DB",
        "credit": 3.0
    }
}
```

Representing Relational Databases

- A Relational Database for School

Student

ID	Name	GPA
001	Joe	3.0
002	Mary	4.0
…	…	…

Course

CNO	TITLE	CREDIT
331	DB	3.0
350	Web	4.0
…	…	…

Enrollment

ID	CNO
001	331
002	350
002	331
…	…

Student Centric Approach

Student

ID	Name	GPA
001	Joe	3.0
002	Mary	4.0
…	…	…

Course

CNO	TITLE	CREDIT
331	DB	3.0
350	Web	4.0
…	…	…

Enrollment

ID	CNO
001	331
002	350
002	331
…	…

JSON Data Model - Student Centric

Student_001.json

```
{
    "id": "001",
    "name": "Joe",
    "gpa": 3.0,
    "enrolled":[
      {
        "course_no": "331",
        "title": "DB",
        "credit": 3.0
      }
    ]
}
```

JSON Data Model - Student Centric

Student_002.json

```
{
    "id": "002",
    "name": "Mary",
    "gpa": 4.0,
    "enrolled":[
      {
        "course_no": "331",
        "title": "DB",
        "credit": 3.0
      },
      {
        "course_no": "350",
        "title": "Web",
        "credit": 4.0
      }
    ]
}
```

JSON Data Model - Student Centric (Single File)

Students.json

```
[
  {
    "id": "001",
    "name": "Joe",
    "gpa": 3.0,
    "enrolled":[
      {
        "course_no": "331",
        "title": "DB",
        "credit": 3.0
      }
    ]
  },
  {
    "id": "002",
    "name": "Mary",
    "gpa": 4.0,
    "enrolled":[
      {
        "course_no": "331",
        "title": "DB",
        "credit": 3.0
      },
      {
        "course_no": "350",
        "title": "Web",
        "credit": 4.0
      }
    ]
  }
]
```

JSON Schema

- JSON Schema is used to specify constraints on JSON Data
- JSON Schema is not required. It is, however, recommended for validating JSON data
- JSON Schema is itself is a JSON document

JSON Schema - Format

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema"
    "$id": <unique_id>
    "type": "object",
    "properties": {
     <property_name>: {"type": <property_type>},
     <property_name>: {"type": <property_type>},
     ....
     }
}
```

JSON Schema - $schema

- $schema:
  - specifies the draft of the JSON schema standard used
  - can be used for version control

- Example:
  - $schema: "https://json-schema.org/draft/2020-12/schema"

JSON Schema - $id

- $id:
  - specifies the URI (Uniform Resource Identifier) of the schema
  - used to determine base URI of the schema
- Example:
  - $id: "https://wsu.edu/cpts415/schemas/student"

JSON Schema - type

- type:
  - specifies the constraint on the JSON data
  - "object" specifies JSON object of the format {<name>:<value>}

- Example:
  - "type": "object"
  - "type": "string"
  - "type": "number"
  - "type": ["string", "number"]

JSON Schema - properties

- properties:
  - specifies constraints on the name - value pairs in JSON data
  - each property specifies constraints on the data type, range, etc.

- Example:

```
"properties": {
    "id": {"type": "string"},
     "name": {"type": "string"}
   }
```

Example - JSON Schema

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://wsu.edu/cpts415/schemas/student",
    "type": "object",
    "properties":
    {
        "id": {"type":"string"},
        "name": {"type": "string"},
        "gpa": {"type": "number"},
        "enrolled":
        {
            "type": "array",
            "items":
            {
                "type": "object",
                "properties":
                {
                    "course_no": {"type": "string"},
                    "title": {"type": "string"},
                    "credit": {"type": "number"}
                }
            }
        }
    }
}
```

JSON Schema - Additional Constraints

- required:
    - constraint use to specify properties that are required
    - specified using an array of properties (eg. "required": ["id", "name"])
    - default: properties are not required

- minimum / maximum / exclusiveMinimum / exclusiveMaximum:
    - Specifies the range for data values of number type
    - minimum/maximum include the constant (i.e. minimum <= value, value <= maximum)
    - exclusiveMinimum / exclusiveMaximum exclude the constant (i.e. exclusiveMinimum < value, value < exclusiveMaximum)

JSON Schema - References

```
{
   "$schema": "https://json-schema.org/draft/2020-12/schema",
   **"$id": "https://wsu.edu/cpts415/schemas/address",**
   "type": "object",
   "properties":
   {
      "street": {"type":"string"},
      "city": {"type": "string"},
      "state": {"type": "string"},
      "zip": {"type": "number"}
   }
}
```

JSON Schema - References (cont.)

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://wsu.edu/cpts415/schemas/student",
    "type": "object",
    "properties":
    {
        "id": {"type":"string"},
        "name": {"type": "string"},
        "gpa": {"type": "number"},
        "address": {"$ref": "https://wsu.edu/cpts415/schemas/address"},
        "enrolled":
        {
            "type": "array",
            "items":
            {
                "type": "object",
                "properties":
                {
                    "course_no": {"type": "string"},
                    "title": {"type": "string"},
                    "credit": {"type": "number"}
                }
            }
        }
    }
}
```

JSON Data Model with References

Student_002.json

```
{
    "id": "002",
    "name": "Mary",
    "gpa": 4.0,
    "address": {"street": "915 N Broadway Ave",
"city":"Everett", "state":"WA", "zip":98201},
    "enrolled":[
      {
        "course_no": "331",
        "title": "DB",
        "credit": 3.0
      },
      {
        "course_no": "350",
        "title": "Web",
        "credit": 4.0
      }
    ]
}
```

Additional Resources:

https://json-schema.org/