Module 12

Tree-Structured data

CS 106 Winter 2018
Trees

Some data is **hierarchical**: we think of each part ("node") as "owning" or "enclosing" some sub-parts, down to some base level.
<table>
<thead>
<tr>
<th>Library of Congress Classification</th>
<th>QA</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA76</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sometimes, a node behaves like a Processing class: it has a specific slot set aside for each kind of child.
Sometimes, a node behaves like a Processing class: it has a specific slot set aside for each kind of child.
Sometimes, a node holds something more like a sequence of children.
Sometimes, a node holds something more like a sequence of children.
There are two standard ways that tree-structured data is passed around online:

- **XML**: eXtended Markup Language
- **JSON**: JavaScript Object Notation

Both are “simple” text-based formats for more or less arbitrary data. Both are available in Processing. We’ll use JSON because it’s nicer to read.
JSON is a small subset of the syntax of Javascript, which can be used to describe a few important data types.

**Primitive types:**
- Integers
- Floats
- Booleans
- Strings

**Compound types:**
- Arrays
- Trees
JSON primitive values

Integers: 0, 27, -4...
Floats: 0.003, 3.1415926, -18.77...
Booleans: true, false
Strings: “hello”, “pancakes!!”...
null
JSON arrays

A JSON Array is a comma-separated list of values, enclosed in square brackets

[]

[1, 2, 3]

[1, true, “hello”]

[-3.14, “kumquat”, [true, false]]

Unlike Processing, the elements do not need to be of one type!
JSON objects

A JSON Object is a comma-separated list of key: value pairs, enclosed in curly braces. It behaves like a dictionary! It maps string keys to arbitrary values.

```json
{}

{ "name": "Craig",
  "extension": 34589 }

{ "digits": [1,2,3], "good": true,
  "remark": "best digits ever" }
```
{  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {  
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type": "mobile",
      "number": "226 555-4567"
    }
  ],
  "children": ["Eunice", "Murgatroyd"],
  "spouse": null
}
Getting JSON Objects

```java
JSONObject stuff = loadJSONObject( "filename.json" );
```

Read the contents of the file into a JSONObject.

Also `loadJSONArray()`, `parseJSONObject()`, `parseJSONArray()`. 
Working with JSONArray

JSONArray arr = ...

int num_entries = arr.size();

... arr.getInt( 0 ) ...
... arr.getFloat( 12 ) ...
... arr.getBoolean( idx ) ...
... arr.getString( jdx ) ...

... arr.getJSONObject( 5 ) ...
... arr.getJSONArray( num_entries - 1 ) ...
Working with JSONObject

```java
JSONObject obj = ...

... obj.getInt( "key" ) ...
... obj.getFloat( "fieldname" ) ...
... obj.getBoolean( "phone" ) ...
... obj.getString( "address" ) ...
... obj.getJSONObject( "whatever" ) ...
... obj.getJSONArray( "etc." ) ...
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 35,
    "address": {
        "streetAddress": "51 Strange Street",
        "city": "Kitchener",
        "province": "ON",
        "postalCode": "N3K 1E7"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "519 555-1234"
        },
        {
            "type": "mobile",
            "number": "226 555-4567"
        }
    ],
    "children": ["Eunice", "Murgatroyd"],
    "spouse": null
}
{  "firstName": "John",  "lastName": "Smith",  "age": 35,  "address": {    "streetAddress": "51 Strange Street",    "city": "Kitchener",    "province": "ON",    "postalCode": "N3K 1E7"  },  "phoneNumbers": [{    "type": "home",    "number": "519 555-1234"  }, {    "type": "mobile",    "number": "226 555-4567"  }],  "children": ["Eunice", "Murgatroyd"],  "spouse": null}

obj
null

obj.getString( "firstName" );
{  
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type": "mobile",
      "number": "226 555-4567"
    }
  ],
  "children": ["Eunice", "Murgatroyd"],
  "spouse": null
}

obj.getInt( "age" );
null
{ "firstName": "John",
"lastName": "Smith",
"age": 35,
"address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
},
"phoneNumbers": [
    {
        "type": "home",
        "number": "519 555-1234"
    },
    {
        "type": "mobile",
        "number": "226 555-4567"
    }
],
"children": ["Eunice", "Murgatroyd"],
"spouse": null
}

obj.getJSONArray("phoneNumbers").getJSONObject(1)
{  "firstName": "John",
   "lastName": "Smith",
   "age": 35,
   "address": {
      "streetAddress": "51 Strange Street",
      "city": "Kitchener",
      "province": "ON",
      "postalCode": "N3K 1E7"
   },
   "phoneNumbers": [
      {
         "type": "home",
         "number": "519 555-1234"
      },
      {
         "type": "mobile",
         "number": "226 555-4567"
      }
   ],
   "children": ["Eunice", "Murgatroyd"],
   "spouse": null
}

obj.getJSONArray("phoneNumbers").getJSONObject(1).getString("number");
Example: RSS Feeds

Radiolab
A podcast powered by FeedBurner

A podcast is rich media, such as audio or video, distributed via RSS. Feeds like this one provide updates whenever there is new content. FeedBurner makes it easy to receive content updates in popular podcatchers.

Learn more about syndication and FeedBurner...

Subscribe Now!

...with web-based podcatchers. Click your choice below:
- Subscribe with NewsGator + netvibes
- My Yahoo!
- Google

...with iTunes:
Add to iTunes

...with something else (copy this address):
http://feeds.wnyc.org/radiolab

Get more info on other podcatchers:
- Get Juice
- Get FireANT
- Nimiq
- Get ZIPOD

View Feed XML

feeds.wnyc.org/radiolab
Radiolab is a show about curiosity. Where sound illuminates ideas, and the boundaries blur between science, philosophy, and human experience. Radiolab is heard around the country on more than 500 member stations. Check your local station for airtimes. Embed the Radiolab widget on your blog or website.

Radiolab is supported, in part, by the Alfred P. Sloan Foundation, enhancing public understanding of science and technology in the modern world. More information about Sloan at www.sloan.org. All press inquiries may be directed to Jennifer Houlihan Roussel at (646) 829-4497.
An RSS feed is an XML document. We *could* parse it directly in Processing, but we’ll make life simpler by converting it to JSON first.

See [rss2json.com](http://rss2json.com).
JSONObject obj = ...;

String first_title =
  obj.getJSONArray( "items" )
  .getJSONObject( 0 )
  .getString( "title" );
Example: counting files
+1!
LEVEL UP
Going live

All `load` functions accept URLs as parameters in addition to file names!

```plaintext
loadStrings()
loadImage()
loadShape()
loadTable()
loadJSONObject()
loadJSONArray()
```
Functions like `loadStrings()` and `loadImage()` allow you to access fixed content over the internet. `loadJSONObject()` is more like calling a function over the web.
Welcome to Open Data API

Hello and Heads up! (September 18th, 2017)

Hello!

I wanted to let you know that effective immediately the Open Data API project at Waterloo is under a new team. We’re looking forward to understanding what exists now, getting feedback from current users, and having a clear plan to communicate before moving forward. We’d like to continue to build on the great work done by those before us that made this project possible.

For the immediate future we will maintain the current production system and fix only critical issues without introducing new features. Part of the motivation for moving forward has been to ensure that the API is fit for use in production.

api.uwaterloo.ca
Example: classrooms

The UW API supports requests like “what courses are scheduled in a given classroom?”

GET /buildings/{building}/{room}/courses.{format}
Example: classrooms

The UW API supports requests like “what courses are scheduled in a given classroom?”

GET /buildings/{building}/{room}/courses.{format}

https://api.uwaterloo.ca/v2/buildings/STC/0040/courses.json
Example: classrooms

The UW API supports requests like “what courses are scheduled in a given classroom?”

GET /buildings/{building}/{room}/courses.{format}
Most online APIs require you to register for a key.
The Google APIs

Google offers dozens of APIs for web designers and developers.

Some are specifically related to popular Google products, like Gmail and Analytics, while others are more specialized and aren’t part of public programs.

All are free to use, of course. You can view all of Google’s APIs and code tools on their site directory.

- Feed API – The Google Feed API lets you download any public feed (including RSS, Media RSS, and Atom) and then combine them into mashups. It simplifies the mashup process by using JavaScript rather than more complex server-side coding.

- Places API – Google Places is a large directory of local businesses and attractions all around the world. The Places API lets you access that information and display it on your website, as well as display check-ins by users.


- Geocoding API – The Geocoding API lets you convert any address into geographic locations.