Working with Web APIs

Lecture 13

Passing arguments: GET

- Arguments are key-value pairs

```
Mascot: Brutus Buckeye
Dept: CS&E
```

- Can be encoded as part of URL

```
scheme://FQDN:port/path?query#fragment
```

- application/x-www-form-urlencoded
  - Each key-value pair separated by & (or ;)
  - Each key separated from value by =
  - Replace spaces with + (arcane!)
  - Then normal URL encoding

```
Mascot=Brutus+Buckeye&Dept=CS%26E
```

Examples

- Wikipedia search

```
http://en.wikipedia.org/
    w/index.php?
    search=ada+lovelace
```

- OSU news articles

```
https://news.osu.edu/
    ?
    q=Goldwater&search.x=1&search.y=0
```

- Random passwords from random.org

```
https://random.org/
    passwords/?
    num=5&len=8&format=plain
```

  - Demo: use Chrome dev tools to "Copy as cURL"
  - See guidelines and API for http clients

Passing Arguments: POST

- Encoded as part of the request *body*
- Advantages:
  - Arbitrary length (URL length is limited)
  - Arguments are not saved in browser history
  - Result is not cached by browser
  - Slightly more secure (not really)
    - But args are less likely to be accidentally shared, because they aren't obvious in the location bar
- Content-Type indicates encoding
  - application/x-www-form-urlencoded
    - Same encoding as used with GET
  - multipart/form-data
    - Better for binary data (else 1 byte→3 bytes)
  - More options too:
    - application/xml, application/json, …

Passing Args: GET vs POST

- GET

```
GET /passwords/?num=5&len=8&format=plain
HTTP/1.1
Host: www.random.org
```

- POST

```
POST /passwords/ HTTP/1.1
Host: www.random.org
Content-Type: application/x-www-form-
urlencoded
Content-Length: 24

num=5&len=8&format=plain
```

Passing Args: Summary

- Arguments in GET requests
  - Request query string
  - Limited length, highly visible
  - application/x-www-form-urlencoded
- Arguments in POST requests
  - Request body
  - No size limit, not bookmarked
  - Choices for how to encode, most common:
    - application/x-www-form-urlencoded
    - multipart/form-data
    - application/json

JSON

- JavaScript Object Notation
- String-based representation of a value
  - Serialization: converting value -> string
  - Deserialization: converting string -> value
- Easy enough for people to read
- Really designed for computers to parse
  - The *lingua franca* for transfer of (object) values via HTTP
  - Used both ways: requests and responses
- MIME type: application/json

JSON Types

- Text: a string, "…"

  `"hello", "I said \"hi\"", "3.4", ""`

- Number: integer or floating point

  `6, -3.14, 6.022e23`

- Boolean

  `true, false`

- Null

  `null`

- Array: ordered list of values, […]

  `[3, 2, 1, "go"], [[1, 3], [7, -2]]`

- Object: set of name/value pairs, {…}
  - Names are text (a double-quoted string)
  - Values are any JSON type (text, array, object…)

  `{"mascot": "Brutus", "age": 19, "nut": true}`

Example

{"current_page":1,"limit":20,"next_page":1,"previous_page":1,"results":[{"id":"GlGBIY0wAAd","joke":"How much does a hipster weigh? An instagram."},{"id":"xc21Lmbxcib","joke":"How did the hipster burn the roof of his mouth? He ate the pizza before it was cool."}],"search_term":"hipster","status":200,"total_jokes":2,"total_pages":1}

Example: Same Value


```
{
  "current_page": 1,
  "limit": 20,
  "next_page": 1,
  "previous_page": 1,
  "results": [
    {
      "id": "GlGBIY0wAAd",
      "joke": "How much does a hipster weigh? An instagram."
    },
    {
      "id": "xc21Lmbxcib",
      "joke": "How did the hipster burn the roof of his mouth? He ate the
pizza before it was cool."
    }
  ],
  "search_term": "hipster",
  "status": 200,
  "total_jokes": 2,
  "total_pages": 1
}
```

Syntax

- Very similar to hash literal in Ruby
  - Inspired by object literal in JavaScript

```
{"dept": "CSE", "class": 3901}
```

  - Spaces and newlines don't matter
- But not identical!
- Important differences
  - Keys must be strings (not symbols)
    - `"dept":` not `dept:`
  - Strings are double quoted (not single)
    - `"CSE"` not `'CSE'`
  - No comments

Example


```
{
  "current_page": 1,
  "limit": 20,
  "next_page": 1,
  "previous_page": 1,
  "results": [
    {
      "id": "GlGBIY0wAAd",
      "joke": "How much does a hipster weigh? An instagram."
    },
    {
      "id": "xc21Lmbxcib",
      "joke": "How did the hipster burn the roof of his mouth? He ate the
pizza before it was cool."
    }
  ],
  "search_term": "hipster",
  "status": 200,
  "total_jokes": 2,
  "total_pages": 1
}
```


```
data['results'][1]['id'] #=>
```

Example


```
{
  "current_page": 1,
  "limit": 20,
  "next_page": 1,
  "previous_page": 1,
  "results": [
    {
      "id": "GlGBIY0wAAd",
      "joke": "How much does a hipster weigh? An instagram."
    },
    {
      "id": "xc21Lmbxcib",
      "joke": "How did the hipster burn the roof of his mouth? He ate the pizza before it was cool."
    }
  ],
  "search_term": "hipster",
  "status": 200,
  "total_jokes": 2,
  "total_pages": 1
}
```


```
data['results'][1]['id'] #=> 'xc21Lmbxcib' (ruby)
data.results[1].id       #=> 'xc21Lmbxcib' (other languages)
```