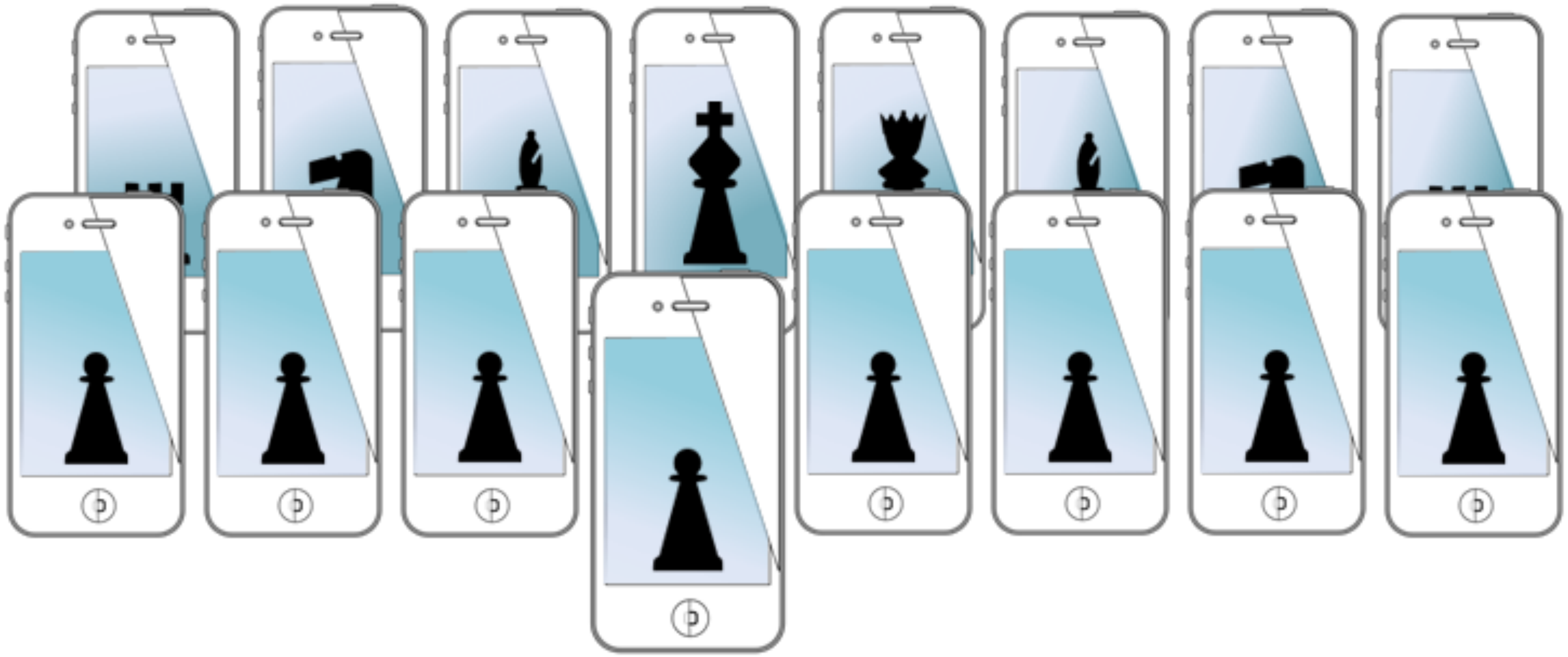
MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 10: tornado, pymongo, and http requests

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University

course logistics

- A5 is due Friday
- start to think about the final project proposal

agenda

- *finish tornado (done!)*
- *mongodb (done!)*
- http requests in iOS

working with your web server

- we want to send data to our hosted server!
  - or any server for that matter
- need to form POST and GET requests from iOS
- we will use NSURLSession

NSURLSession

- proper way to configure a session with a server
- new format starting in iOS7
  - old way was to use `NSURLConnection`
  - before that was to use `sendAsynchronousRequest`
- you may see code for `initWithContentsOfURL`:
  - **never, never, never** use that for networking
- sessions are a huge improvement in iOS
  - and extremely powerful
  - the Stanford course talks about these (check it out)!
  - as promised, we will cover different topics than Stanford course

NSURLSession

- delegate model
- does authentication if you need it!
    - we won't use that though — who would hack our server?
- implements pause / resume
- can setup download tasks as locally saved files
    - nice for separating the data and metadata

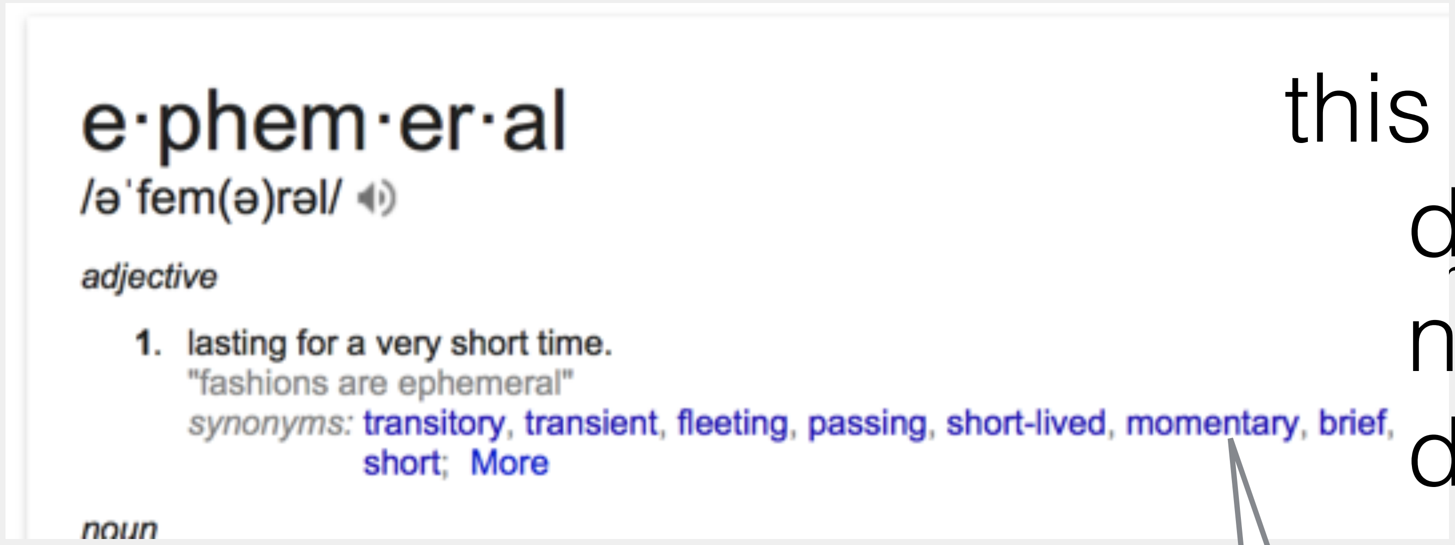
configure a session

```
@interface SMUViewController () <NSURLSessionTaskDelegate>

@property (strong,nonatomic) NSURLSession *session;

@end
```

delegation

will reuse session

```
//setup NSURLSession (ephemeral)
NSURLSessionConfiguration *sessionConfig =
    [NSURLSessionConfiguration ephemeralSessionConfiguration];
```



e·phem·er·al

/əˈfem(ə)rəl/

adjective

1. lasting for a very short time.
   "fashions are ephemeral"
   synonyms: transitory, transient, fleeting, passing, short-lived, momentary, brief, short; More

noun

this is private!!

do not cache

nfig

no cookies

do not store credentials

background queue

configure a session

- other options:

`ephemeralSessionConfiguration`

`defaultSessionConfiguration`

use global cache, cookies, and credential storage objects

`backgroundSessionConfiguration`

make my session respond to push notifications,
launch my app, if needed, handle download completion

configure a task

- tasks are common types of requests
- tied to a session
- give a way to specify URL and type of request
- will format data in specific ways to handle response from server
- we will use a "completion handler"
  - more involved downloads allow use of delegates
    - progress indicators
    - completion indicators

NSURLSessionDataTask

- common to use for GET requests
- uses blocks for completion — or could use… ?

```
dataTaskWithURL:completionHandler:

 NSURL *getUrl = [NSURL URLWithString: [baseURL stringByAppendingString:query]];

 NSURLSessionDataTask *dataTask = [self.session dataTaskWithURL:getUrl
                  completionHandler:^(NSData *data,
                                      NSURLResponse *response,
                                      NSError *error) {
                      NSLog(@"%@",response);
                      NSLog(@"%@",[[NSString alloc] initWithData: data
                                                encoding:NSUTF8StringEncoding]);

                  }];
     [dataTask resume]; // start the task
```

NSString

NSString

convert bytes to string

must call, or stays suspended

NSURLDownloadTask

- sub-class of NSURLSessionDataTask
- many delegate methods for getting progress

```
    NSURLSessionDownloadTask *downloadTask = [self.session downloadTaskWithURL:[NSURL
URLWithString:@"someurlfordownloadingimages.com/coolimage"]
         completionHandler:^(NSURL *location, NSURLResponse *response, NSError *error) {
                if(!error)
                {
                    UIImage *img = [UIImage imageWithData:[NSData dataWithContentsOfURL:location]];
                }
            }];
    [downloadTask resume];
```

could use delegate instead of completion handler

```
-(void)URLSession:(NSURLSession *)session
      downloadTask:(NSURLSessionDownloadTask *)downloadTask
didFinishDownloadingToURL:(NSURL *)location

-(void)URLSession:(NSURLSession *)session
      downloadTask:(NSURLSessionDownloadTask *)downloadTask
      didWriteData:(int64_t)bytesWritten
totalBytesWritten:(int64_t)totalBytesWritten
totalBytesExpectedToWrite:(int64_t)totalBytesExpectedToWrite
```

NSURLUploadTask

- common to use for POST requests
- need to setup your own HTTP request (not just URL)

```
uploadTaskWithRequest:fromData:completionHandler



// create a custom HTTP POST request
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:postUrl];
[request setHTTPMethod:@“POST"];

NSData *imageData = UIImageJPEGRepresentation(image, 0.25);

NSURLSessionUploadTask *uploadTask =
[self.session uploadTaskWithRequest:request
                           fromData:imageData
          completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

          }];
```

could be any data

could use delegate methods

NSURLDataTask

- can also use this for PUT or POST requests
- highly similar to uploadTask
    - but you don't get the delegate methods for progress

```
// create a custom HTTP POST request
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:postUrl];
[request setHTTPMethod:@"POST"];

NSData *imageData = UIImageJPEGRepresentation(image, 0.25);
[request setHTTPBody:imageData];

NSURLSessionDataTask *dataTask =
[self.session dataTaskWithRequest:request
         completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

          }];
```

JSON serialization

- parse in tornado

```
import json

class JSONPostHandler(BaseHandler):
   def post(self):
    '''Parse some posted data
    '''
    data = json.loads(self.request.body)
```

- parse in iOS

```
NSDictionary *responseData = [NSJSONSerialization JSONObjectWithData:data
                                 options: NSJSONReadingMutableContainers
                                   error: &error];
```

the output in both scenarios is a dictionary

NSDictionary

- serialize in iOS

```
NSData *requestBody=[NSJSONSerialization dataWithJSONObject:jsonUpload
                                     options:NSJSONWritingPrettyPrinted
                                       error:&error];
```

tornado + iOS demo

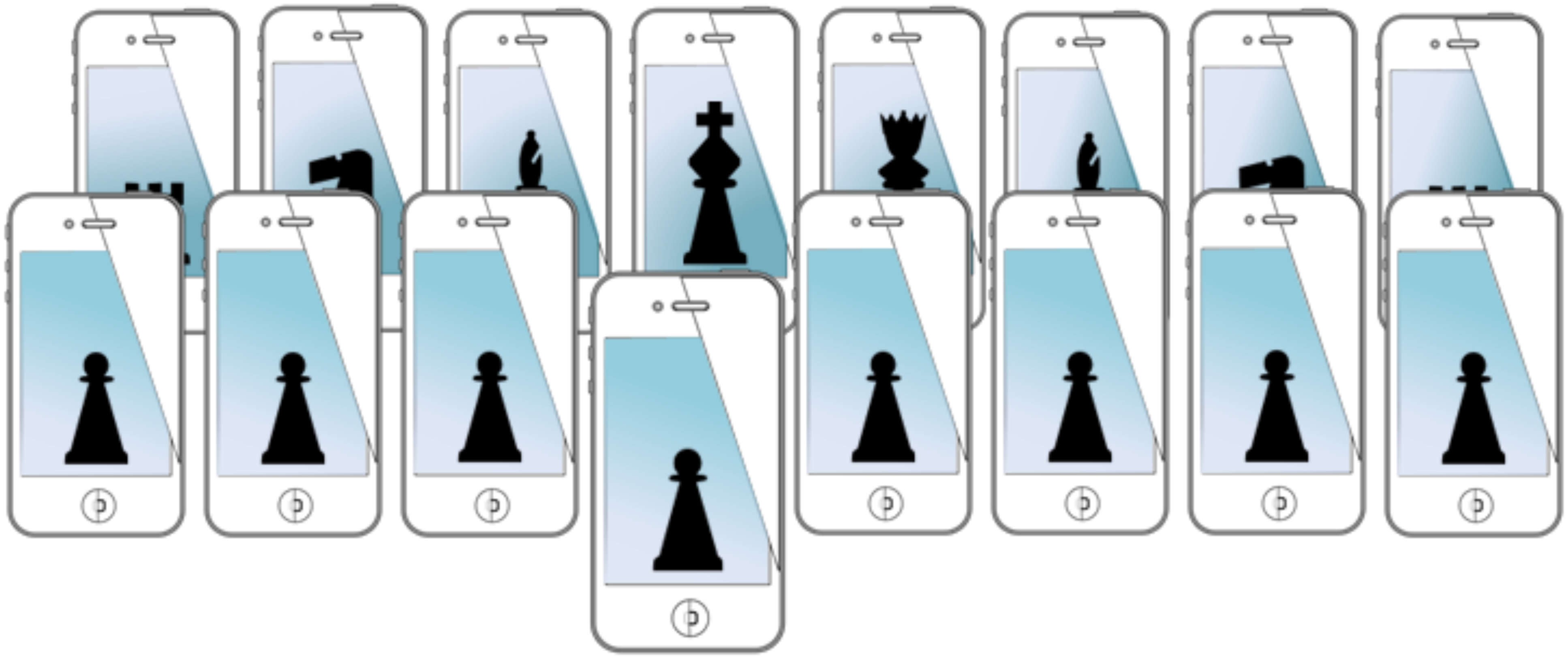
- send a GET request, handle query in tornado
- do POST with GET-like query
- do POST with JSON in, JSON out

we just learned objective c syntax, but…
I rewrote the examples in Swift!

for next time…

- next time: basics of machine learning
  - machine learning as a service
    - install scikit-learn (already there with anaconda)
- next next time: **in-class-assignment** with our own restful API
  - using ML and networking to do cool things…
  - some code has changed since filming, but mostly the same

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 10: tornado, pymongo, and http requests

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University