Coding REST

Hint: This is really short

Why is this really short?

Because you already have all the tools to code REST!

- REST relies on HTTP and what it considers “appropriate” use of the protocol
  - Use the verbs: we know how to do this in Java (or Node, or…)
  - Set the request and response headers
  - Use proper response codes
- What about data formats?
  - Yeah there has been a lot of ongoing discussion, as we said
  - As coders though, we just need to be proficient in JSON and know what we are parsing, and look for help along the way
- What about web apps?
  - What about them? We aren’t writing the Javascript (AJAX is covered in a different class), so REST basically made our server-side life real easy
  - OK, let’s take a little detour and review the web apps case with AJAX…

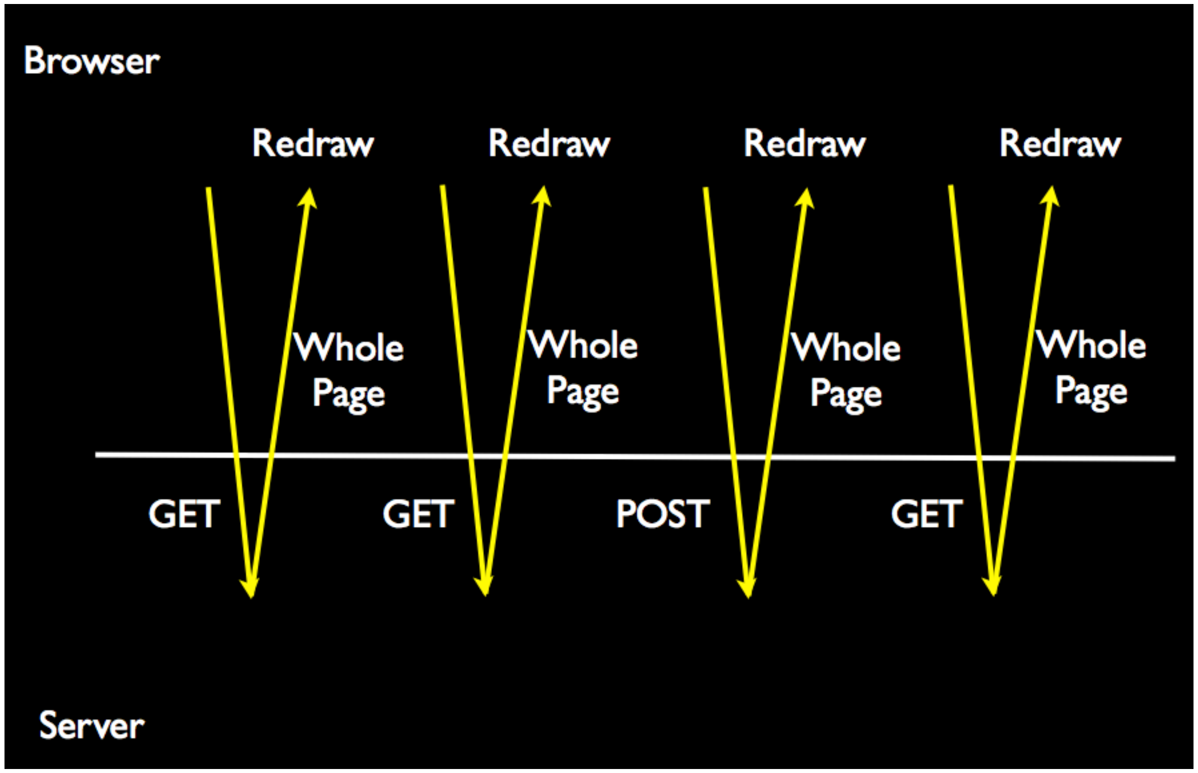
REST-driven web application development

REST+AJAX

So what does REST have to do with web apps?

Consider our server-side web app pattern:

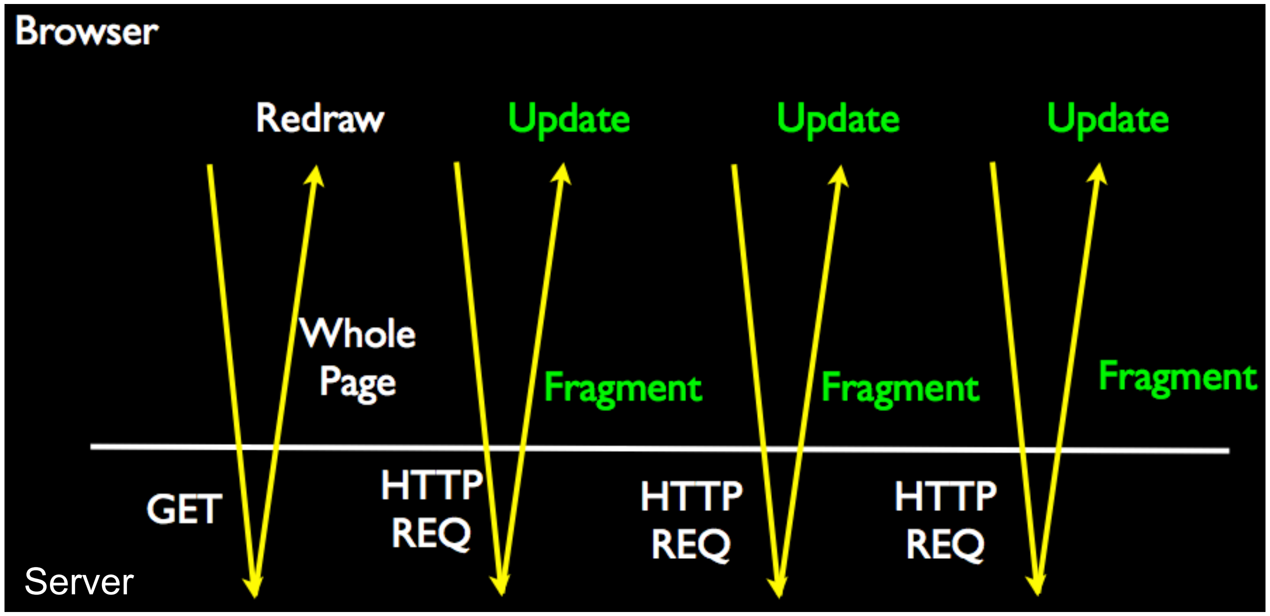
- A user takes some action like a click on a link or button
- The browser makes a TCP/IP connection to the web server
- The browser sends a POST or GET request
- The server sends back a rendered page to display to the user
- Repeat the Request-Response Cycle...



XMLHttpRequest and AJAX

By 1999, Microsoft wanted to move some of the processing of web pages from the web server to the web browser

- The idea was instead of sending whole pages of HTML to the browser, send out the data to be displayed as XML and then produce presentation in JavaScript in the browser
- Originally a Microsoft innovation - other browsers soon adopted the idea and it became a defacto standard with a little variation between browsers
- It soon became clear that this could send *anything* - not just XML back and forth between a browser and client



AJAX Motivation

Standard Request/Response

- Each click presents a whole new screen

AJAX

- Each action sends data and receives results in the background.
- The browser typically gets back a fragment of HTML, XML, or JSON which is used to update a portion of the screen using the browser document model

Why is this better?

- Well, sometimes it isn't ☺
- But it allows us to move presentation and perhaps some small amount of business logic to the client
  - Scales the server better
  - Allows for personalized, dynamic page composition
  - Let the local environment decide what to do with its responses (*systems* are on the *servers*, *apps* are on the *clients*)

REST and AJAX Summary

REST

- Lot of confusion as to what REST really is; a lot of people think that "RESTful URIs" means they are implementing REST.
  - No examples of a true REST frameworks and no compliance tests to gauge RESTfulness.
  - True REST is a lot of work; may be overkill unless application is expected to be long-term and work across multiple domains and organizations.
    - Sacrifices efficiency for scalability, interoperability, and generality; multiple requests to get one piece of data.
- Uses a proven and successful architecture: the World Wide Web
  - Why not use something that already works?
  - It's lightweight compared to other alternatives like SOAP.

REST + AJAX:

- REST is useful w/out AJAX, but together they are "like peas & carrots"
- REST, AJAX, and HTML5 mean this is the app pattern moving forward
- Server-side developers need to think RESTfully or "service-ly"

A Note on CORS

What's the problem?

- The trend in web app development is to load an app *container* in the browser and have it be responsible for rendering & routing (see: MVC).
- AJAX in the webpage may want to speak to REST APIs from *locations that were not the origin* server
  - But this will not be allowed by default by the browser due to security constraints (*same origin policy,* how does it know it is not a rogue injection?)
  - Note that resources other than Javascript (images, stylesheets) are just fine

Solution: CORS (Cross-Origin Resource Sharing)

- Basically, a web application should know what REST APIs it will use, and therefore should indicate what is required.

Cross-origin Resource Sharing (CORS):

- Browser includes an Origin header in an OPTIONS request (pre-flight)
- Server provides Access-Control-Allow-Origin response header indicating what domains are allowed (or * for all)
- It is up to the browser to enforce the policy

http://www.w3.org/TR/cors/

Back to our regularly scheduled program(ming)

CODING FOR REST

Aren’t you going to help us at all Dr. Gary?

- Well, you *do* know HTTP by now
  - But yeah, there are some frameworks out there
- If you live in REST-land, be aware of a few things:

1. REST API Design Tools
   - Swagger2 and swag-edit, API Blueprint, RESTlet Studio, etc.
2. Choose a response format:
   - Content-formats: XML or JSON to start. Then you have to decide:
     – Do I need a schema?
     – Do I need semantic metadata?
   - Web linking: RFC8288 or HAL by default I’d say, but there are a lot of other options, including roll-your-own
3. Frameworks (and framework extensions)
   - (Java) Spring MVC Rest extensions like @RestController
   - Jersey (Java’s reference implementation we will use)
   - More Java: Restlet, RestEasy, Jello – lots more
   - Node: restify, express-based middleware, hapi-js middleware, and about a zillion modules you can get via npm

OK, here is a (very) little help - Jersey

Jersey is Java’s reference implementation of JAX-RS, the Java standard for rest services.

Summary:

- Provides servlet that routes requests to classes in a *provider* package
- Classes use *annotations* to determine the subpaths they respond to, what media types they produce and consume, and to specify what HTTP verbs they respond to with what methods
  - Example class annotations:
    - @Path(“/foo/”) – respond to subpath “foo” under our web context
    - @Produces(“<mime type>”) – convenience class MediaType provided
  - Example method annotations:
    - @Path, @Produces, @Consumes - like the above scoped to a method
    - @PathParam – uses URL part after subpath as one or more params in {}
    - @QueryParam – uses query string params with a @DefaultValue option
- Method return values are mapped to media type Content Types via a “best guess” by Jersey, with fine-grained control possible (stay tuned)

What about the semantic formats stuff?

REVISIT THIS SLIDE AFTER WE TALK ABOUT THESE NEXT WEEK

REST Coding Summary

REST is an architectural style based on conceptual underpinnings of the modern web

- As such, there is no "one way" to do REST
- Nor is there a specific tool or technology
    - There are some standards that can help you out, but REST is more an "ideal" than it is a "technology"

Of course there are many different tools, libraries, and frameworks that can help you out

- You don't actually need to do any of them; you can use your low-level language support for HTTP, and code in a "RESTful" way
- But like most things, the machinery ends up getting defaulted in libraries, frameworks, and tools that help developers work faster and with less defects – customization is always fun tho'!
- In this class we will use Jersey for JAX-RS and some Jackson for JAXB to facilitate REST