Epilogue

Web Architecture

- Ubiquitous, global on-ramp
- Connects everything to everything, based on URI-addressable resources
  - With a uniform interface
- Also provides standard coordination mechanism
  - Status codes!
- And is ambivalent about content
    - Media types!

POX

- Treats HTTP as a synchronous transport protocol
  - Great because it gets through firewalls
- But again breaks expectations
  - HTTP is not MOM!
- Misses out on all the good stuff from the Web
  - Status codes for coordination
  - Caching for performance
  - Loose coupling via hypermedia
  - Etc
- Not as good as proper message-oriented middleware
  - Which are low-latency, reliable, etc.

CRUD Services

- The simplest kind of Web-based service
- Embraces HTTP and Web infrastructure
  - Four verbs, status codes, formats
  - Cacheable!
- Can easily describe them
  - URI templates
  - WADL
- But tightly couples client and server
  - Might not be a problem in some domains

Hypermedia

- It's all about media types and link relations!
  - Describe state machines with lots of lovely links
- Constrain what you can do to resources with the uniform interface
- Loosely coupled
  - The server mints URIs to resources, clients follow them
  - Easily spans systems/domains (URIs are great!)
- Embraces the Web for robustness
  - Verbs, status codes, caching
- Design and implementation:
  - Design application protocol state machines;
  - Implement resource lifecycles;
  - Document using media types, link relation values and HTTP idioms.

Scalability

- Everything you know still applies
  - Stateless is good
  - Horizontal is good
- Yet everything you know no longer applies!
  - Text-based synchronous protocol is scalable???
- Do as little work as possible
  - Make interactions conditional
    - ETags and if-modified etc are your friends
  - Cache!

Be a Good Citizen

- Automate, test, CDCs
- Blue/green deploy for service continuity
- Warp-n-weft of project and service
- Monitor your estate, query its topology data

Don’t go nuts

- A few weeks back I talked to a prominent SI
  - Who happened to have popularised microservices
- There were some fascinating success stories
- And some total WTFs, e.g.
  - A simple web site but..
  - Implemented as half a dozen single page apps and…
  - Each backed by its own compliment of microservices
- Because microservices are not the only architectural style!

A final word from Mike Amundsen

A number of maxims have gained currency among the builders and users of microservices to explain and promote their characteristic style.

1. Make each microservice do one thing well. To do a new job, build afresh rather than complicate old microservices by adding new features.
2. Expect the output of every microservice to become the input to another, as yet unknown, microservice. Don't clutter output with extraneous information. Avoid strongly-typed or binary input formats. Don't insist on object trees as input.
3. Design and build microservices to be created and deployed early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.
4. Use testing and deployment tooling (in preference to manual efforts) to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

http://www.amundsen.com/blog/archives/1164

Retrospective

- A short retrospective on the course and address any outstanding questions or discussion topics
- Then we'll pop for a drink!

RESTful Microservices

Thanks for attending!

Dr. Jim Webber
@jimwebber