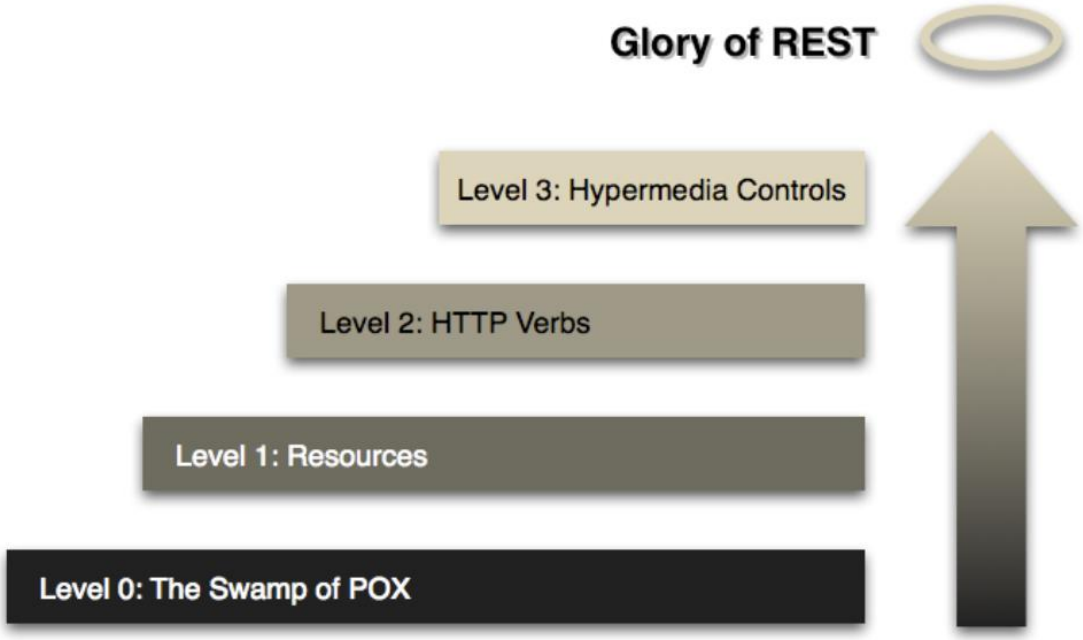
In this lecture, we will discuss…

- REST Introduction
- RESTful Services: Design Principles

REST: Introduction

- **RE**presentational **S**tate **T**ransfer
- Resource Instance(s) are identified by URI (Uniform Resource Indicator)
  - http://www.movieservice.com/movie/:id
  - http://www.movieservice.com/movie/12345
- Introduced by Roy Fielding in 2000

“Glory of REST”: Richardson Maturity Model



Source: http://martinfowler.com/articles/richardsonMaturityModel.html

REST: Web Services

- Stateless
- Expose directory structure-like URIs
- Supports multiple formats but JSON/XML – most popular formats.

Representations

- Represents a resource (Movie)
- A resource can contain other resources (Movie → Roles)
- Representation does not restrict representation format – XML/JSON
- JSON is ideal for web pages (RoR/Ajax)

HTTP Protocol

- GET - retrieve a resource
- POST - create a resource
- PATCH – update partial resource
- PUT - change the state of a source or to update it
- DELETE - remove a resource
- HEAD – similar to GET but no message body

Stateless

- ✧ Stateful
  - /movies/getNextPage
  - server needs to store previous page
- ✧ Stateless
  - /movies?offset=25&limit=4
  - /movies?page=3

Uniform Resource Indicator (URI)

http://www.movieservice.com/movies/12345

http://www.movieservice.com/movies/12345/roles

http://www.movieservice.com/movies/12345/roles/100

- Lower case
- Underlying technology can change

Resource Representations

Common MIME Types

MIME-Type	Content-Type
JSON	application/json
XML	application/xml
HTML (XHTML)	application/xhtml

- Custom Type - application/vnd+company.category+xml

Summary

- HTTP-based web - predominant WS design model
- Simplicity - Most are in the level2 to 3 level
- "Truly RESTful" services only when you - add solid support for state, links to the use of URIs, methods, and exchangeable content

What's Next?

- REST Web Services - Resources