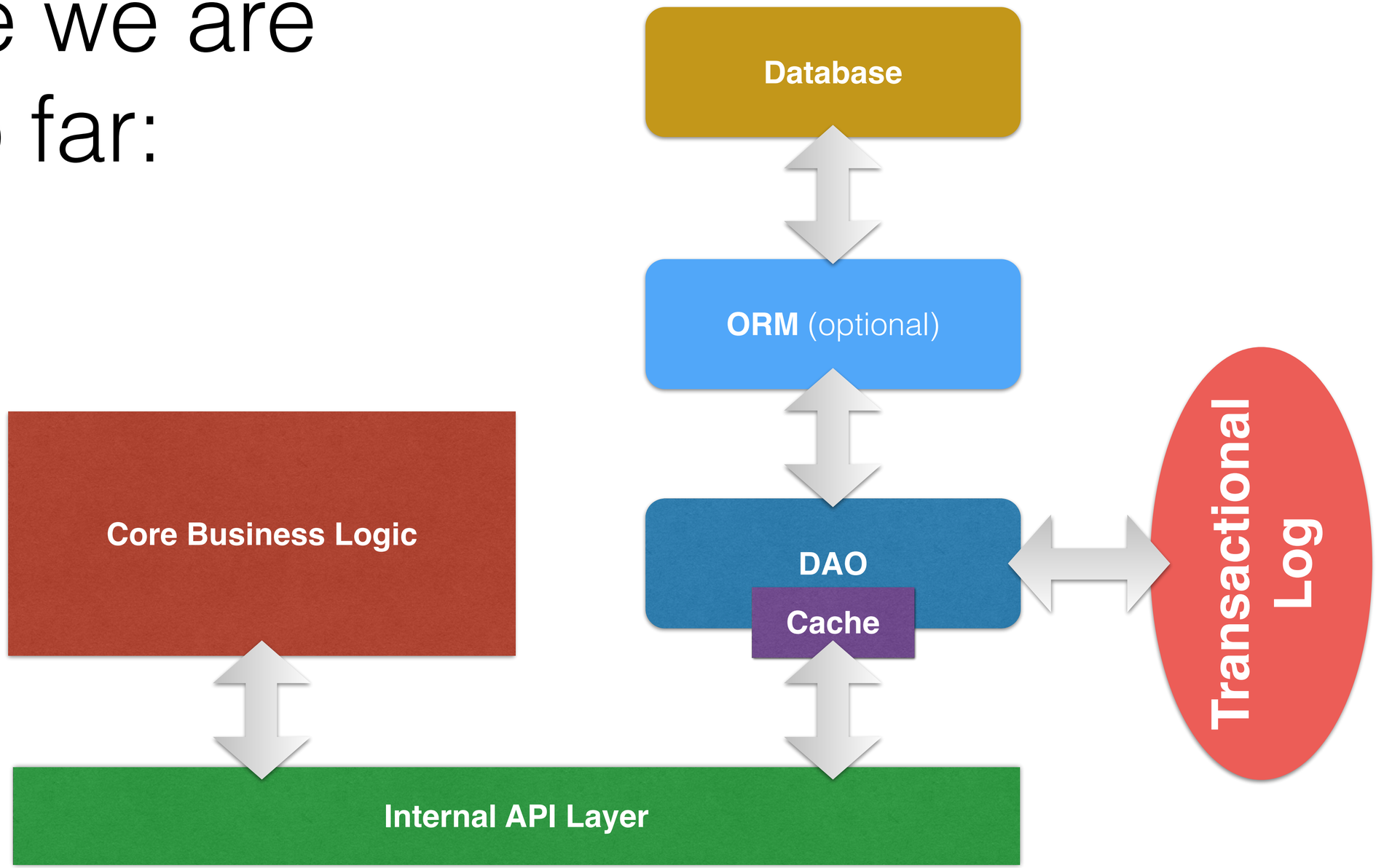
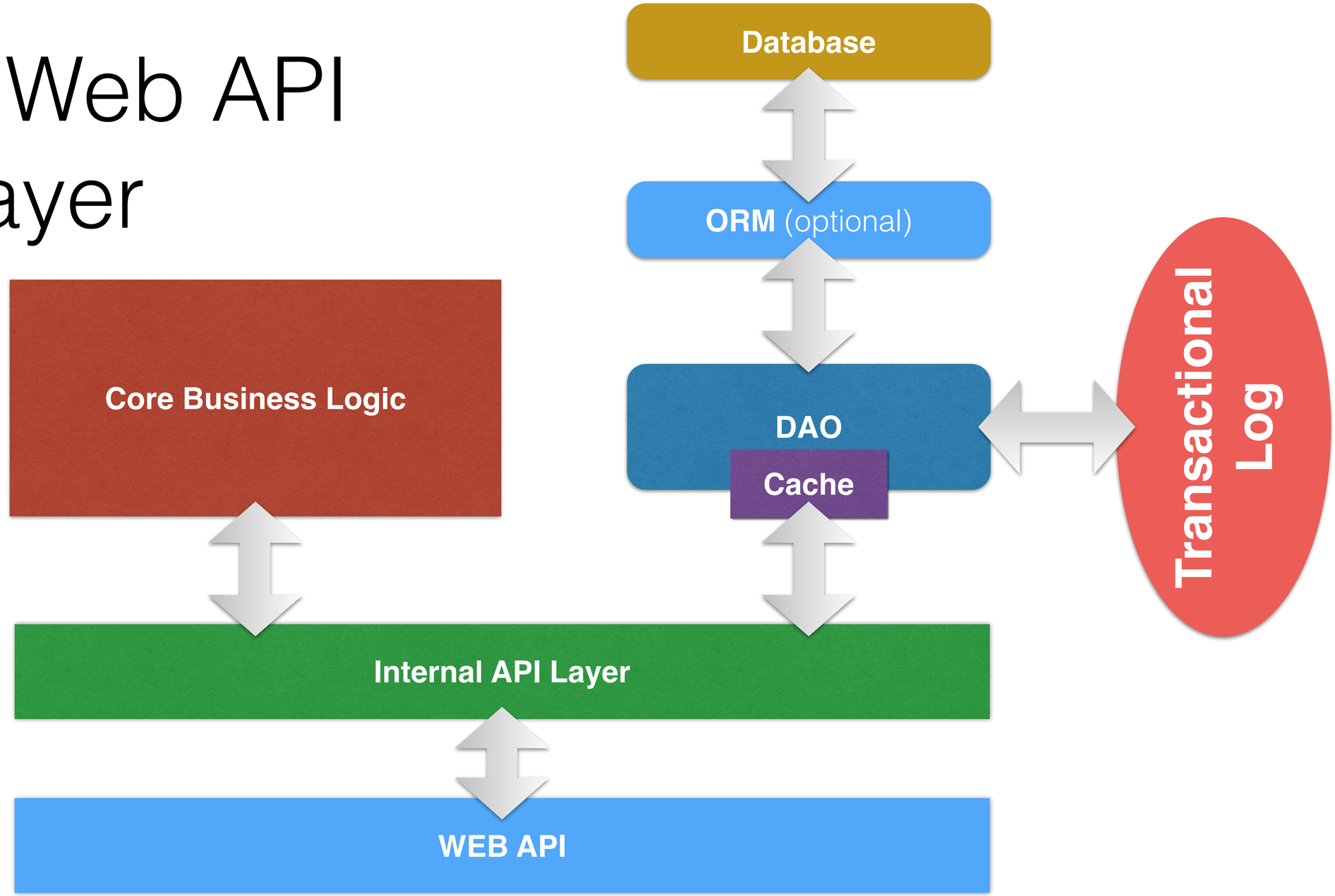
Lecture 5

CSCI 6907.12 - Full Stack Application Engineering

Where we are so far:



Today: Web API Layer



Web API Layer

- **Purpose**: To provide access to our business logic from the web.
- **Why do we need it**: Business requirement requires it. (Web is the most prevalent way to do our computing these days.)

HTTP

- Hypertext Transfer Protocol
- Runs on top of TCP layer
- Originally intended for transferring documents over the internet
- Protocol of choice for the World Wide Web
- How your application and the browser communicates
- Text based protocol

Modern HTTP

- Not for mere documents
- Any resources that are needed for application
- Latest Version HTTP2
- Most widely used version HTTP v1.1

Characteristics

- Stateless Protocol
- REQUEST <-> RESPONSE

Request

- A request line (resource and method)
- Request header fields
- An empty line
- An optional message body

Response

- A Status-Line (status code and reason message)
- Response header fields
- An empty line
- An optional message body

Request Methods

- GET - request for a resource
- HEAD - same as GET but no response body
- POST - request for a creation of new resource
- PUT - request for a modification of a resource
- DELETE - request for a deletion of a resource
- TRACE - echo the request
- OPTIONS - asks what methods are possible
- CONNECT - converts the request connection to a transparent TCP/IP tunnel
- PATCH - request for partical modification of a resource

Headers

- See https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

RESTful API

- Representational state transfer
- Uniform Interface
- Stateless
- Cacheable
- Client-Server
- Layered System

Uniform Interface

- Resource-Based
- Manipulation of Resources Through Representations
- Self-descriptive Messages
- Hypermedia as the Engine of Application State (HATEOAS)

Stateless

- The necessary state to handle the request is contained within the request itself, whether as part of the URI, query-string parameters, body, or headers.

HTTP Verbs

- GET, POST, PUT, DELETE
- Example
  - GET /profiles - all profiles
  - GET /profiles/23 - profile with the id
  - GET /profiles/23/addresses
  - GET /profiles/23/addresses/534

HTTP Verbs

- POST /profiles - create a new profile
- POST /profiles/23/addresses - create a new match
- PUT /profiles/23 - update profile
- PUT /profiles/23/addresses/543
- DELETE /profiles/23 - update profile
- DELETE /profiles/23/addresses/543

Response

- 200 OK
- 201 CREATED
- 204 NO CONTENT
- 400 BAD REQUEST
- 401 UNAUTHORIZED
- 403 FORBIDDEN
- 404 NOT FOUND
- 405 METHOD NOT ALLOWED
- 409 CONFLICT
- 500 INTERNAL SERVER ERROR

Message Type

- JSON or XML
- Distinguish with extension
  - GET /profiles/23.json
  - GET /profiles/23.xml

Good API Design Principles

- Once public, DO NOT CHANGE!
- Versioning
  - GET http://dateme.com/api/v1/profiles/23
  - GET http://dateme.com/api/v2/profiles/23
- Do one thing and one thing well!
  - Let the clients compose those
- Add filtering and limits via query parameters
  - GET http://dateme.com/api/v1/profiles?limit=100&lastname=kim

Service Oriented Architecture