CISC 3120

C25: Creating Web Services using the Spring Framework

Hui Chen

Department of Computer & Information Science

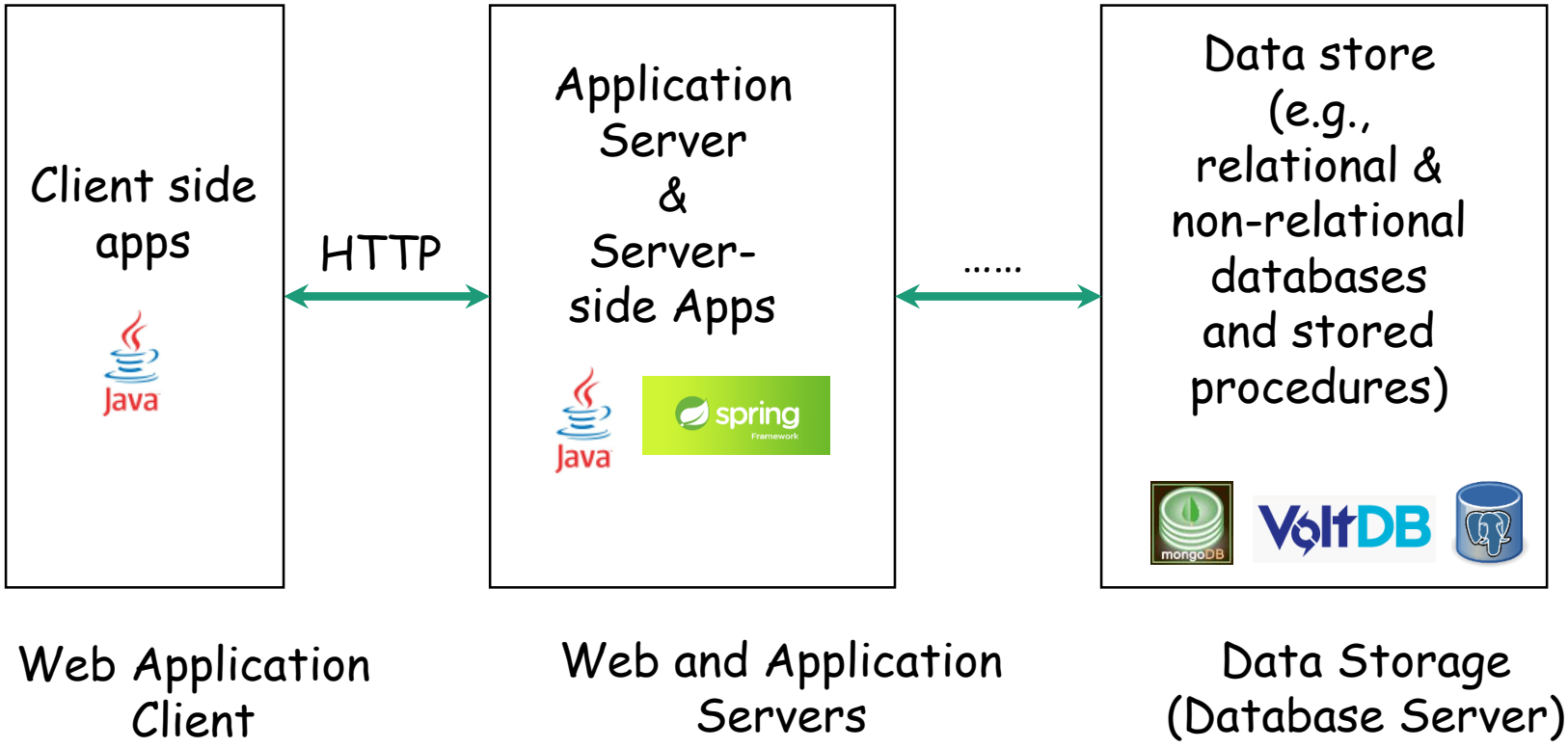
CUNY Brooklyn College

Outline

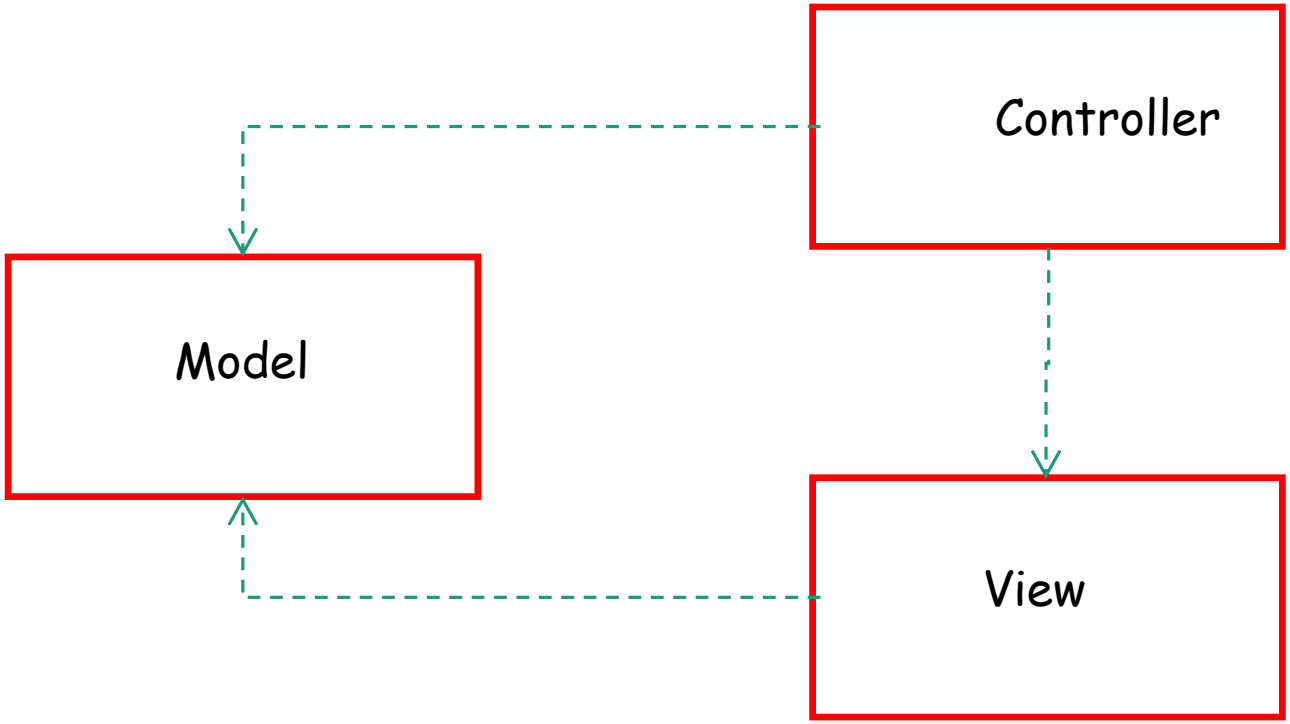
- Recap & Review
  - The Model-View-Controller (MVC) pattern
- MVC on the Web and evolution
- Spring MVC
  - Evolving Spring MVC and Web Services
- REST and RESTful Web Services
- Implementing RESTful Web Services
  - Query and update (read and write)

Web Application Architecture

- 3-tier (and n-tier)

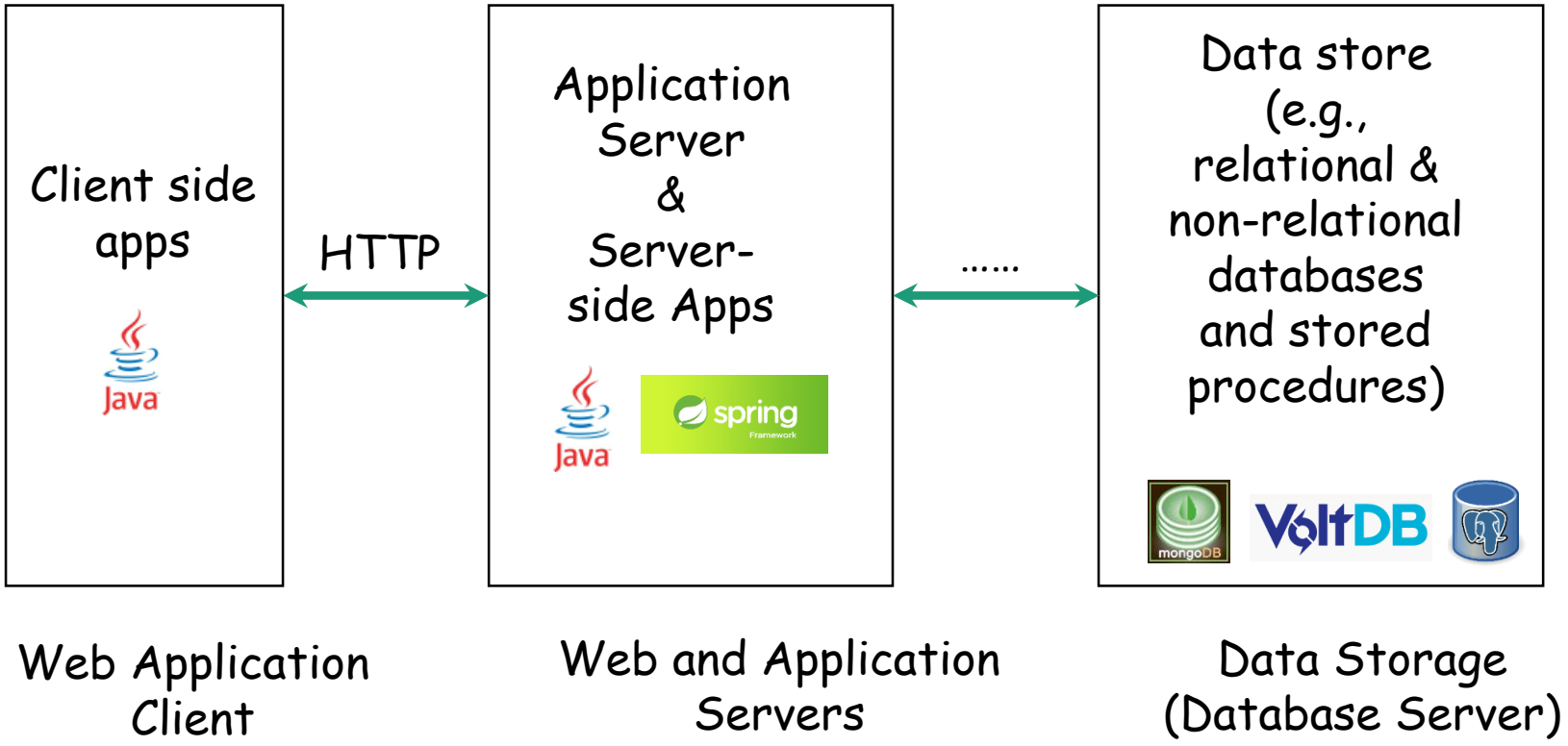


Model-View-Controller



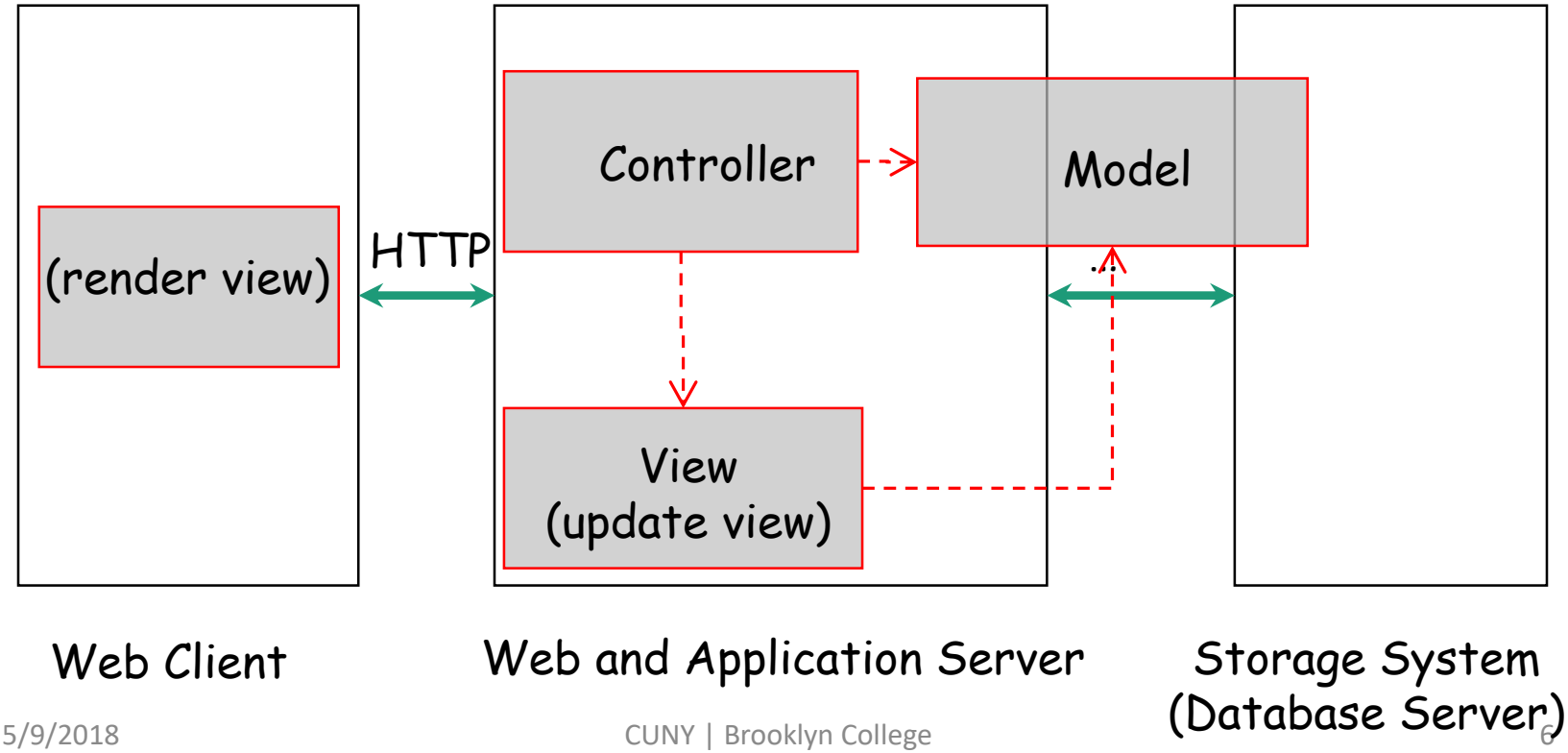
Apply MVC to the Web?

- Where are they?



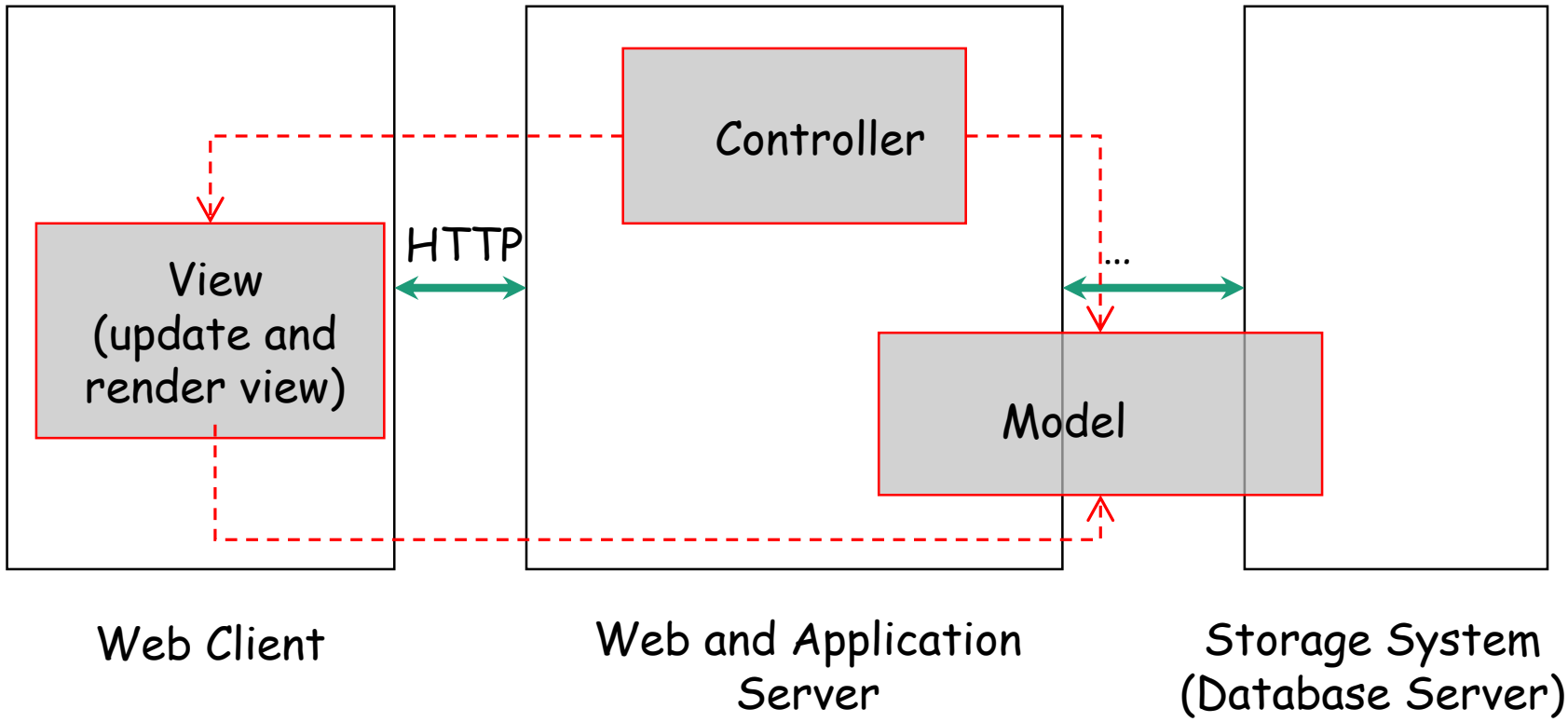
Update and Render View

- Update view at the server & render view at the client



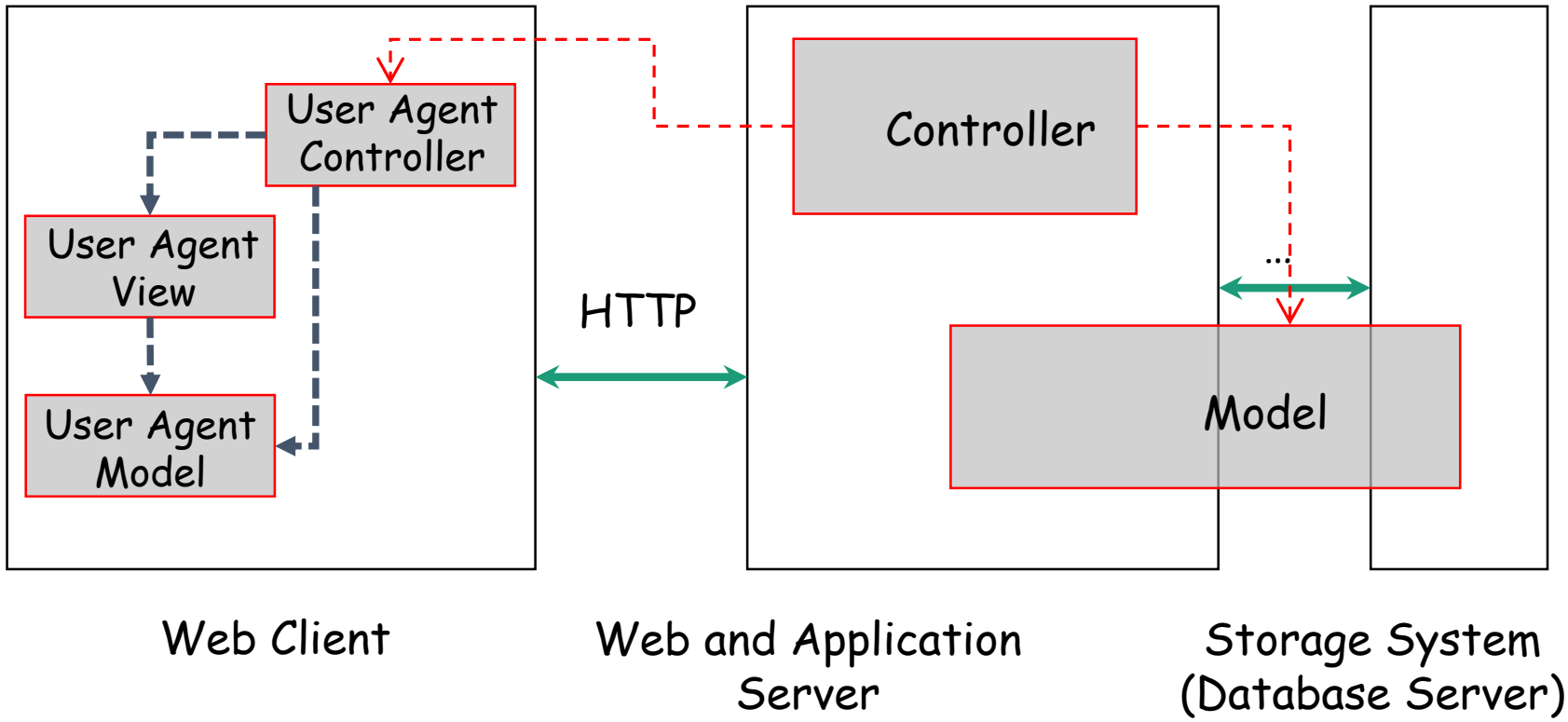
Update and Render View

- Update and render view at the client



Update and Render View: MVC and MVC …

- Update and render view at the client



Web MVC: Evolution

- Initially, update view at the server
  - Constant network connectivity
    - Wired connectivity, stationary stations
  - Few interactions, tolerate high latency
    - Page-based update, synchronous request-response
- Evolve to update view at the client
  - Intermittent network connectivity
    - Wireless connectivity, mobile stations
  - Many interactions, expect low latency
    - Component-based update, asynchronous request-response
  - Often, RESTful web services & feature-rich client

REST

- Representation State Transfer
- Architecture style for networked-base applications
  - Define a set of constraints
  - Commonly used in the Web applications

REST Constraints

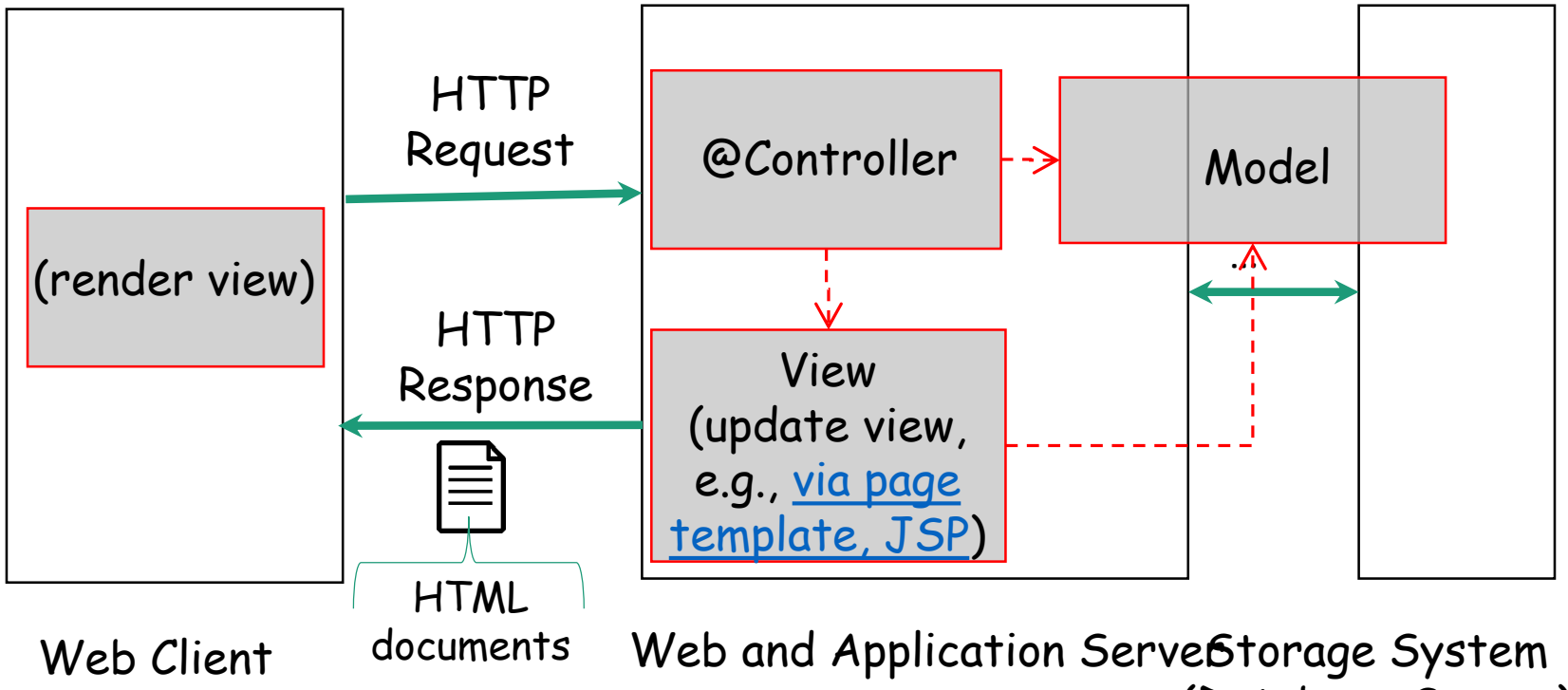
- All important resources are identified by one resource identifier mechanism
  - e.g., URI
  - "everything is a resource on the web"
- Access methods have the same semantics for all resources
- Resources are manipulated through the exchange of representations, e.g., URI
- Representations are exchanged via self-descriptive messages, e.g., HTTP messages
- Hypertext as the engine of application state
  - HTTP is stateless
  - Maintaining state via hypertext messages

Questions

- Apply MVC to the Web?
- Evolution of the Web
  - Feature rich Web applications
- REST and RESTful Web services

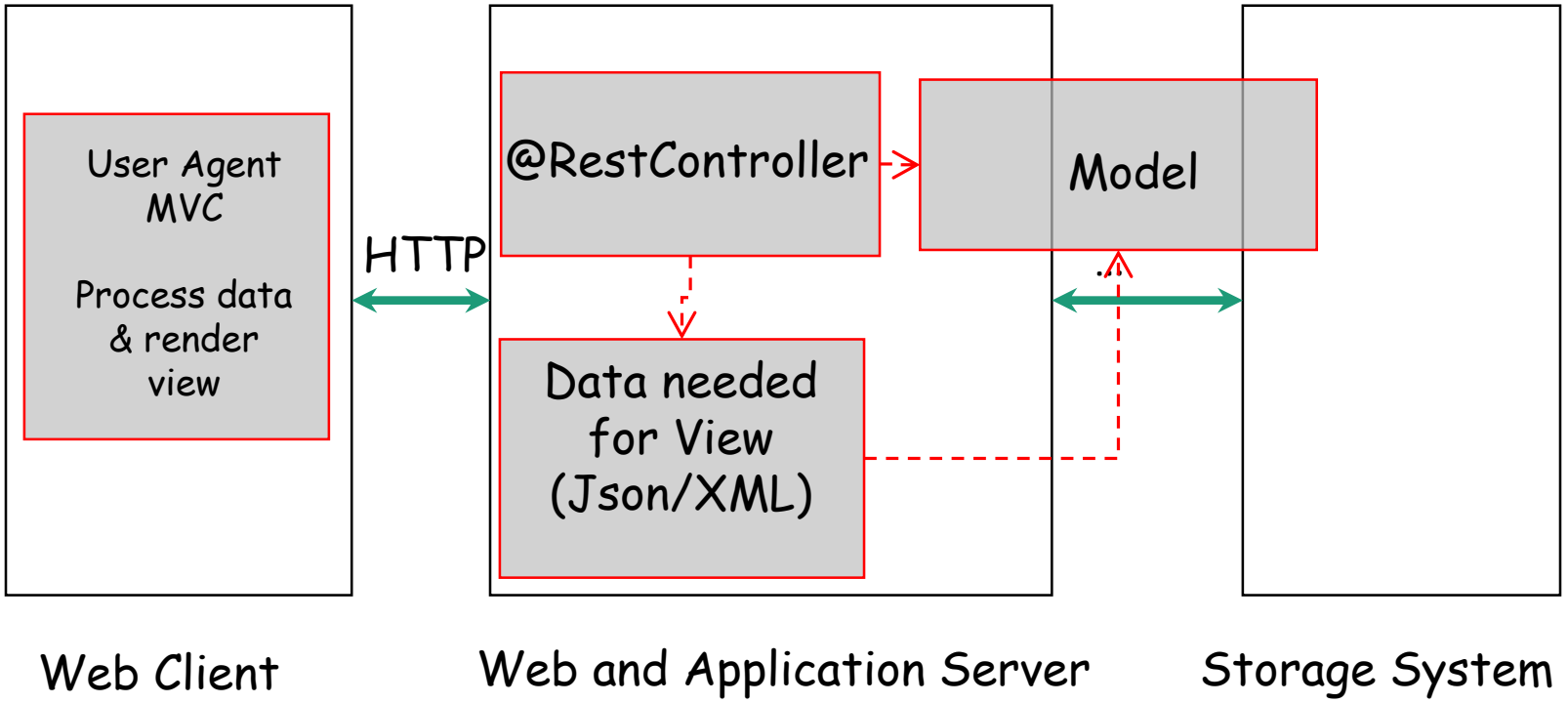
Design with Spring Controller: User Agent for Rendering Views

- @Controller annotate a Web Controller



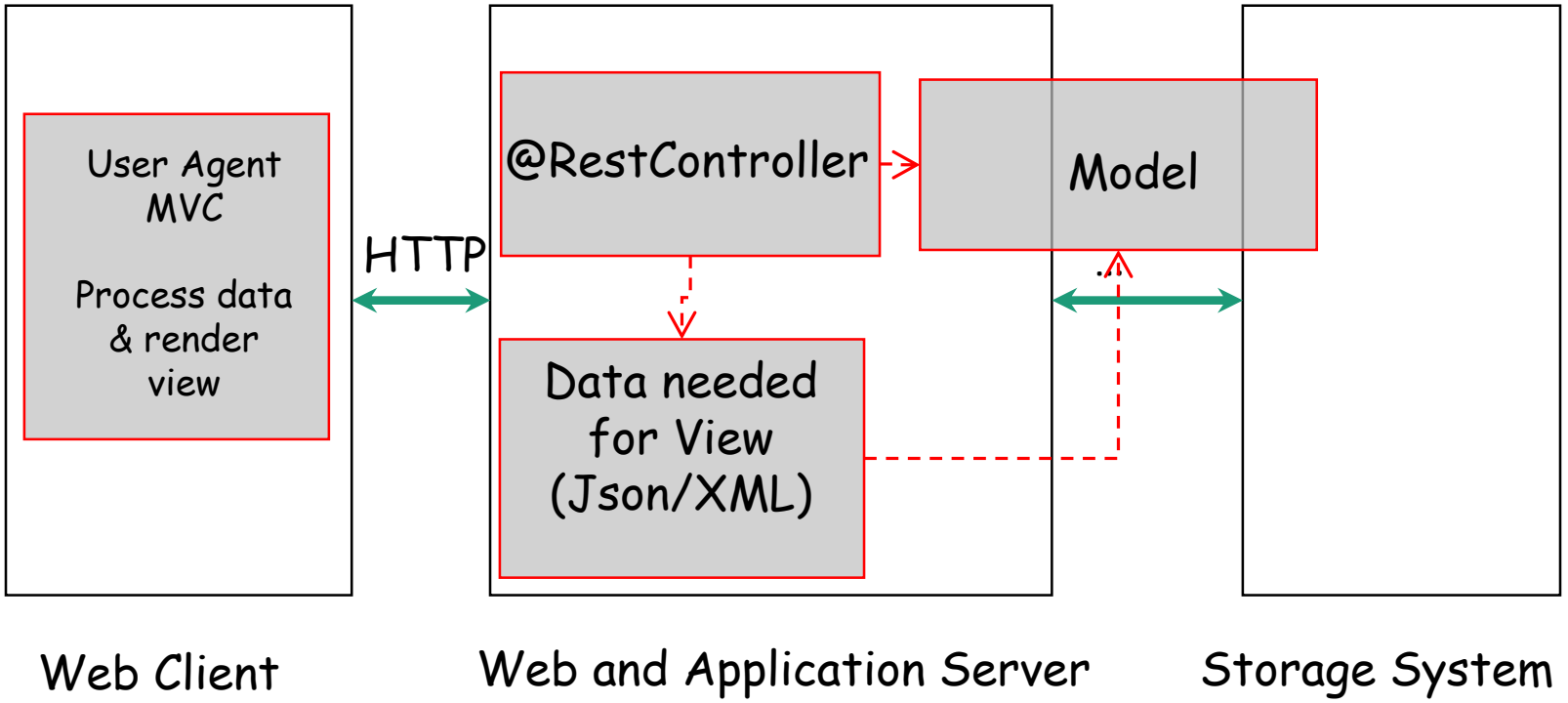
Design with Spring Controller: User Agent Processing Data

- @Controller annotates a Web Controller, @ResponseBody annotates response type



Design with Spring Controller: REST and User Agent

- @RestController combines @Controller and @ResponseBody

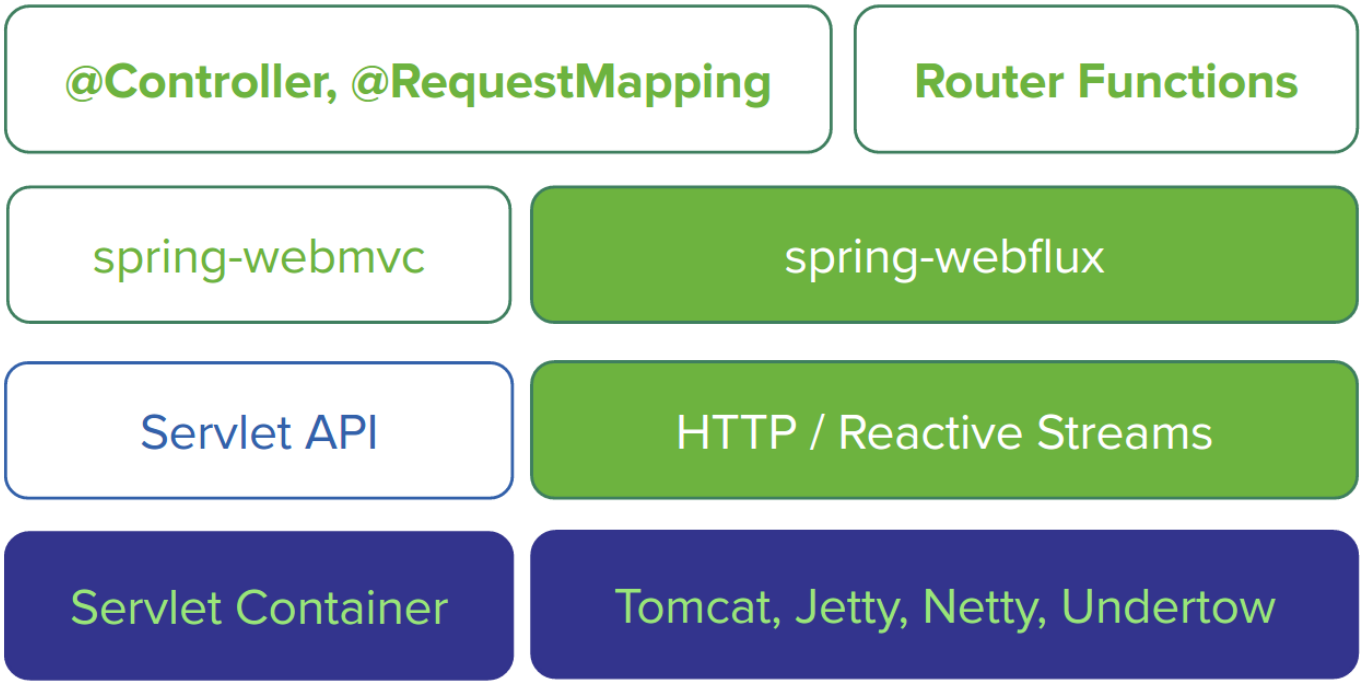


Approaches in the Spring Framework

- Spring WebMVC
  - The original Spring Web framework built on the Servlet API from inception of the Spring framework.
- Spring WebFlux
  - The new Spring Web framework introduced in the Spring Framework 5.0

The Spring Framework version 5

- The Spring Framework is evolving as the Web development.



Spring WebMVC

- A MVC implementation on the Web
- A central controller that processes and dispatches all HTTP requests
- Problem
  - Synchronous (Filter and Servlet)
  - Blocking (getParameter and getPart methods)

Spring WebFlux

- Support a non-blocking web stack
- Handle concurrency with a small number of threads
- Scale with less hardware resources

Reactive Spring Web

- WebFlux support reactive programming
  - Applications are built around reacting to change, e.g.,
    - Network component reacting to I/O events
    - UI controller reacting to mouse events.
- Non-blocking is reactive
  - Reacting to notifications as operations complete or data becomes available

Questions

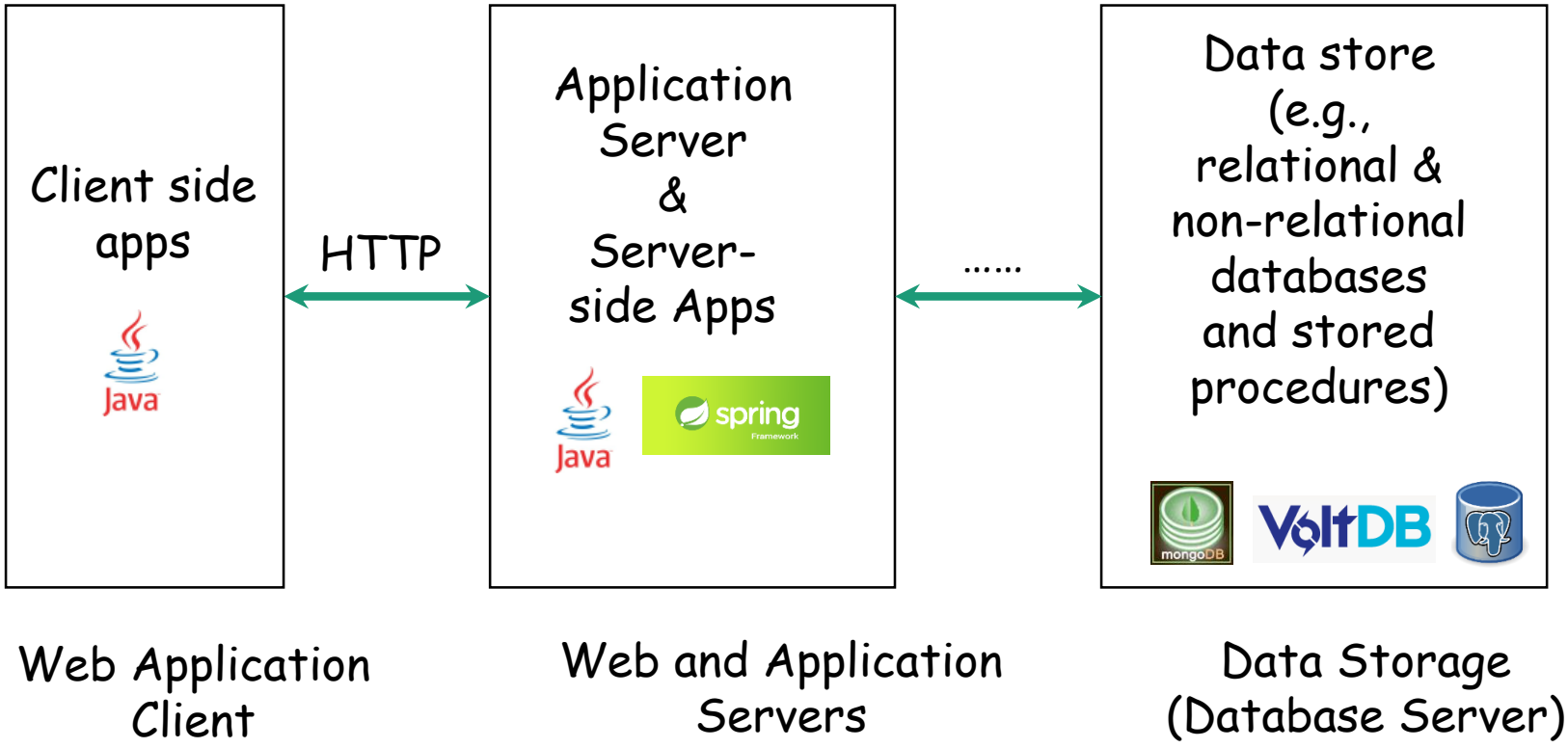
- The Spring Framework
- Overview of creating RESTful services in the Spring Framework

Query and Update

- Two primary types of operations
  - Query
    - Retrieve data from the Web services based on the input from the user agent
  - Update
    - Update data at the Web services using the input from the user agent

Persistent Data Store

- Necessary to store data



Basic Data Store: CRUD

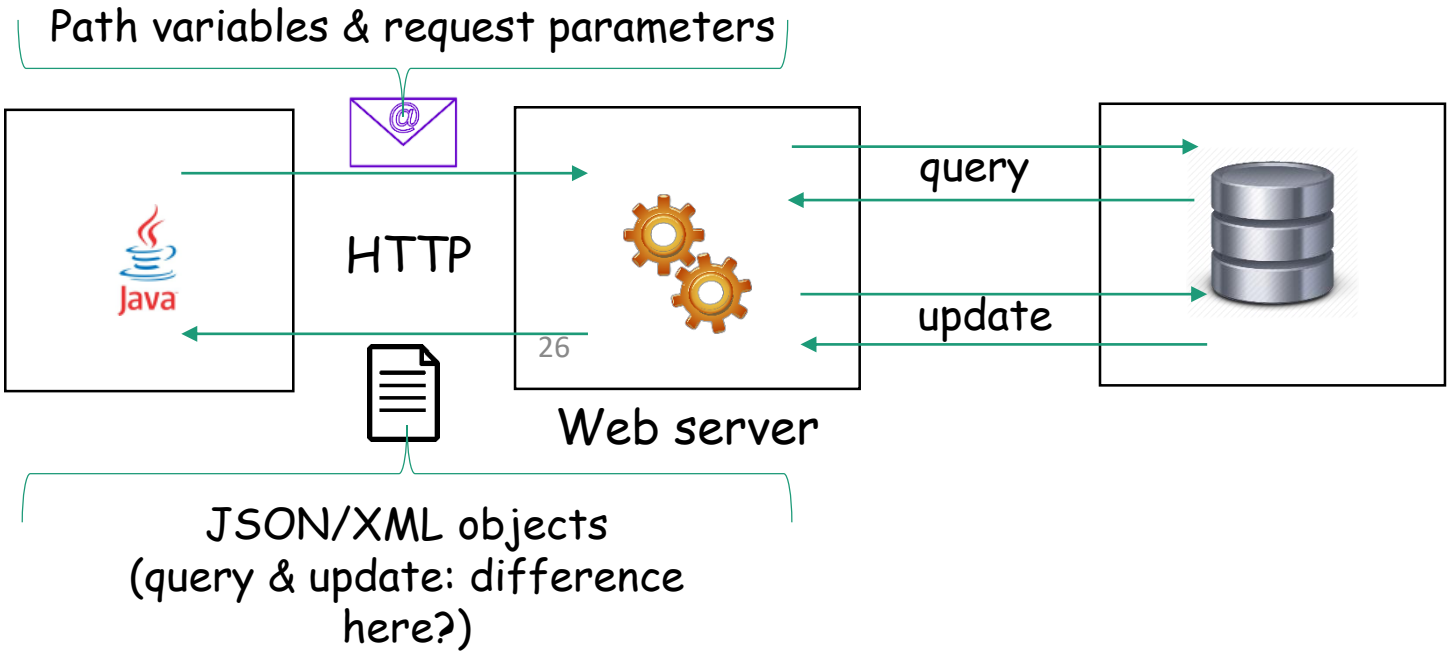
- Basic functions of persistent storage
  - Create, read, update, and delete
- Map to HTTP requests
  - Examples
    - PUT /addresses/1
    - GET /addresses/1
    - POST /addresses
    - DELETE /addresses/1

Example Data Store: MongoDB

- Document-like persistent storage
- Basic operations based key-value pairs

Example with Persistent Data Store

- Passing little data (e.g., primitive values or Strings)
- Passing lots of data (e.g., an object)



Questions

- Persistent data stores
- Examples
  - Passing small amount of data
  - Passing lots of data
  - Query and update (read and write)

Example Application with Persistent Data Store

- RESTful Web service using Spring Framework
- MongoDB for data storage

Example: GpaMongoWebService

- pom.xml
  - Examine the dependencies
- Application configuration
  - application.properties (where is it?)
    - Port number, SSL certificate, and database
- GpaMongoWebService
  - Created by the Spring Suite Tools
    - Mongo repositories
- GpaController
  - Mapping between Web APIs and methods

Example: GpaMongoWebService: Persistent Data Store

- Package edu.cuny.brooklyn.web.data
  - Built upon Spring's CRUD repositories
    - Queries implementations are automatically created by the Spring framework
    - However the method signatures must conform with the Spring convention, e.g.,
      - findBy…, findAll…, findAllBy…
- A few Spring Framework interfaces
  - CrudRepository, PagingAndSortingRepository, MongoRepository, SimpleMongoRepository

Example: GpaMongoWebService: Error Handling

- Package edu.cuny.brooklyn.web.exception
  - Return error message in JSON

Example: GpaMongoWebService: Model and Data

- Package edu.cuny.brooklyn.web.service
- Package edu.cuny.brooklyn.web.data

Eye-Ball Testing

- Testing without implementing a User Agent
  - Designing and implementing a User Agent can be time consuming
- Recommend cURL
  - Sample scripts provided for both Windows and OS X/Linux
  - Run scripts from command line
  - Create scripts by revising scripts for your own work
    - API
    - data

Further Reading

- Roy T. Fielding, Richard N. Taylor, Justin R. Erenkrantz, Michael M. Gorlick, Jim Whitehead, Rohit Khare, and Peyman Oreizy. 2017. Reflections on the REST architectural style and "principled design of the modern web architecture" In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (ESEC/FSE 2017). ACM, New York, NY, USA, 4-14. DOI: https://doi.org/10.1145/3106237.3121282
- Spring Framework Documentation & Guide
  - "Understanding REST"
  - "Building a RESTful Web Service"
  - "Serving Web Content with Spring MVC"
  - "Reactive Programming with Spring 5.0 M1"
  - "Notes on Reactive Program Part I and Part II"

Questions

- Understand the big picture
- Create Web applications in Java
- Example Web services design and implementation
- Eye-ball testing