CISC 3120

C28: The Spring Framework: WebMVC and WebFlux

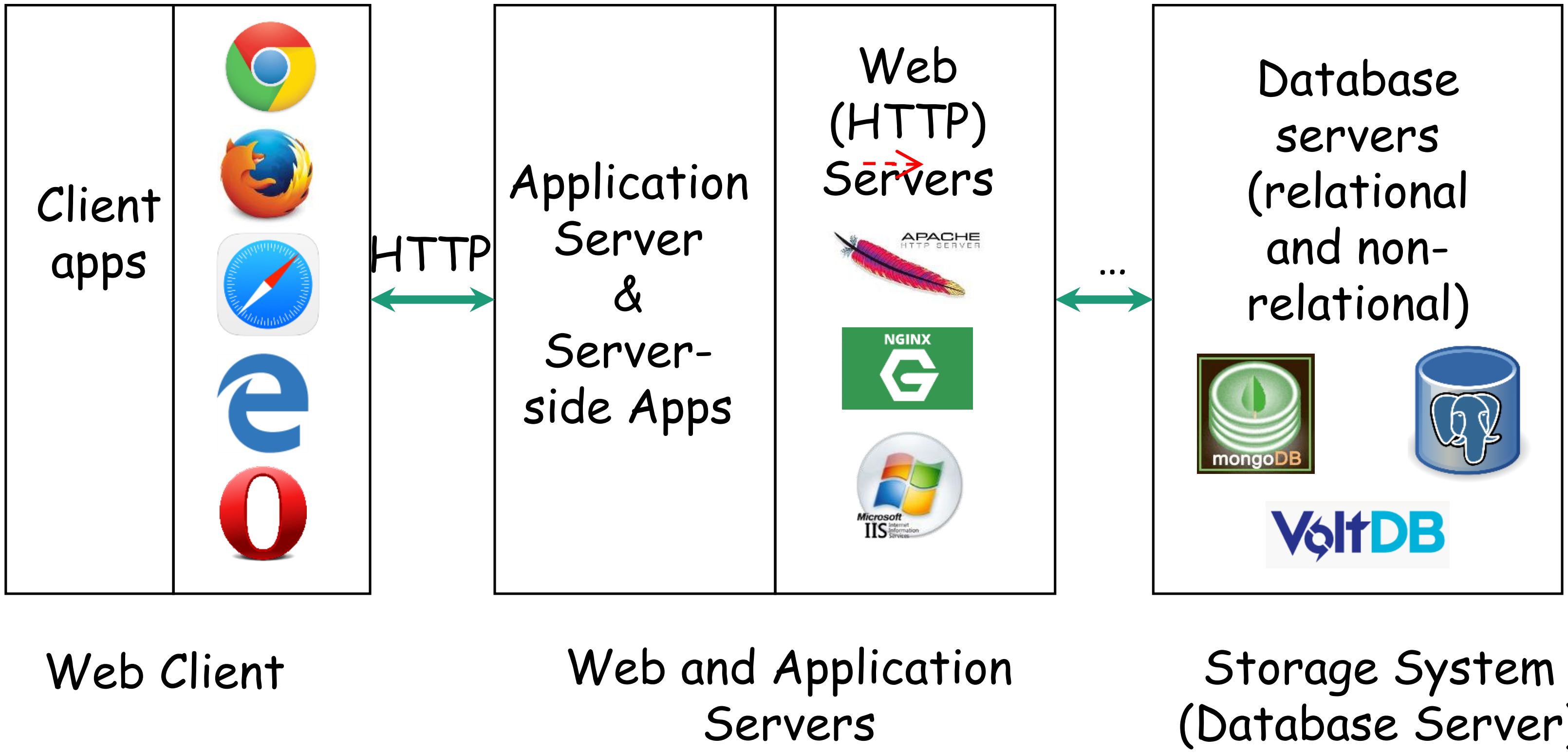
Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

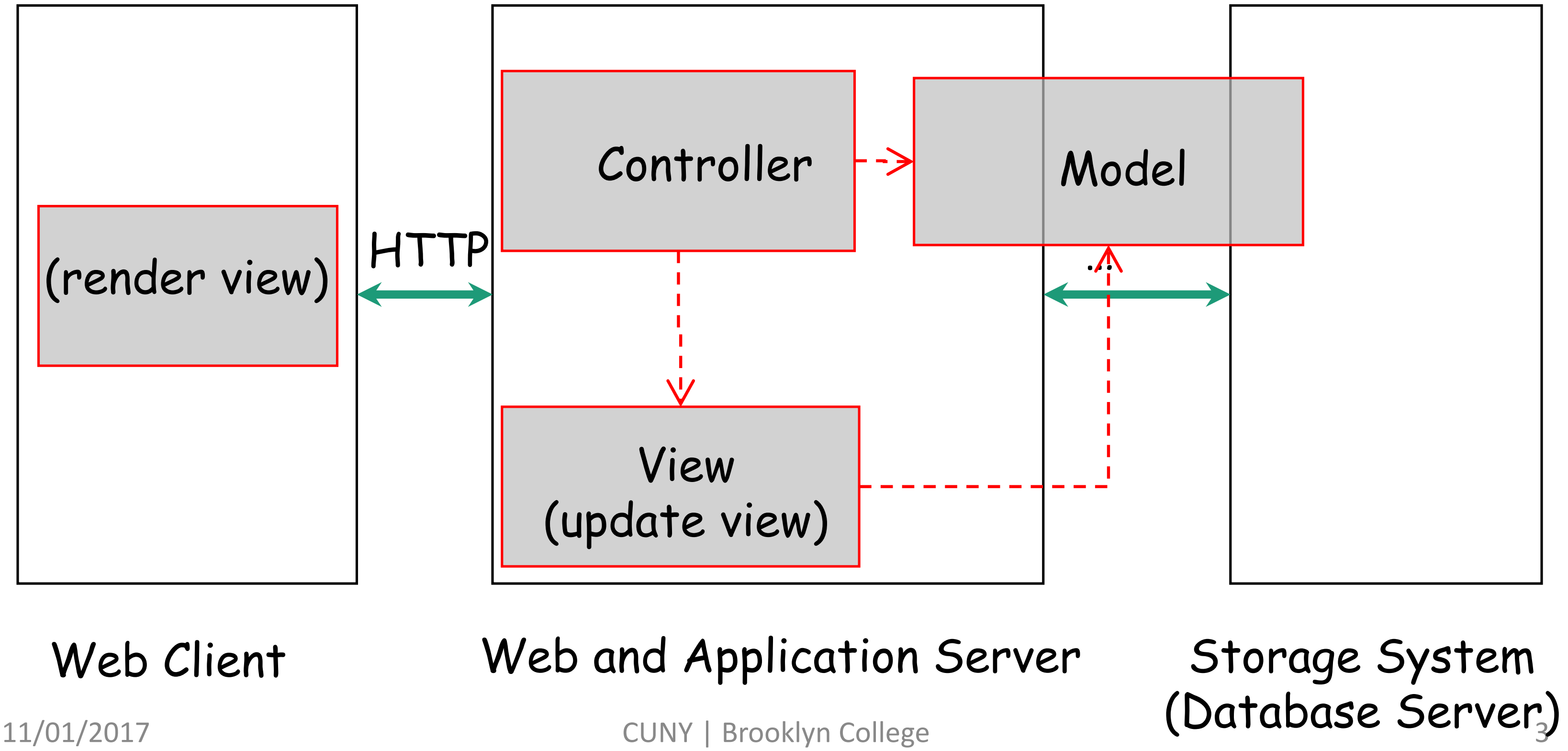
Reexamine MVC on the Web

- Where is the view updated and rendered?


Client apps
HTTP
Application Server & Server-side Apps
Web (HTTP) Servers
APACHE HTTP SERVER
NGINX
Microsoft IIS
…
Database servers (relational and non-relational)
mongoDB
VoltDB
Web Client
Web and Application Servers
Storage System (Database Server)

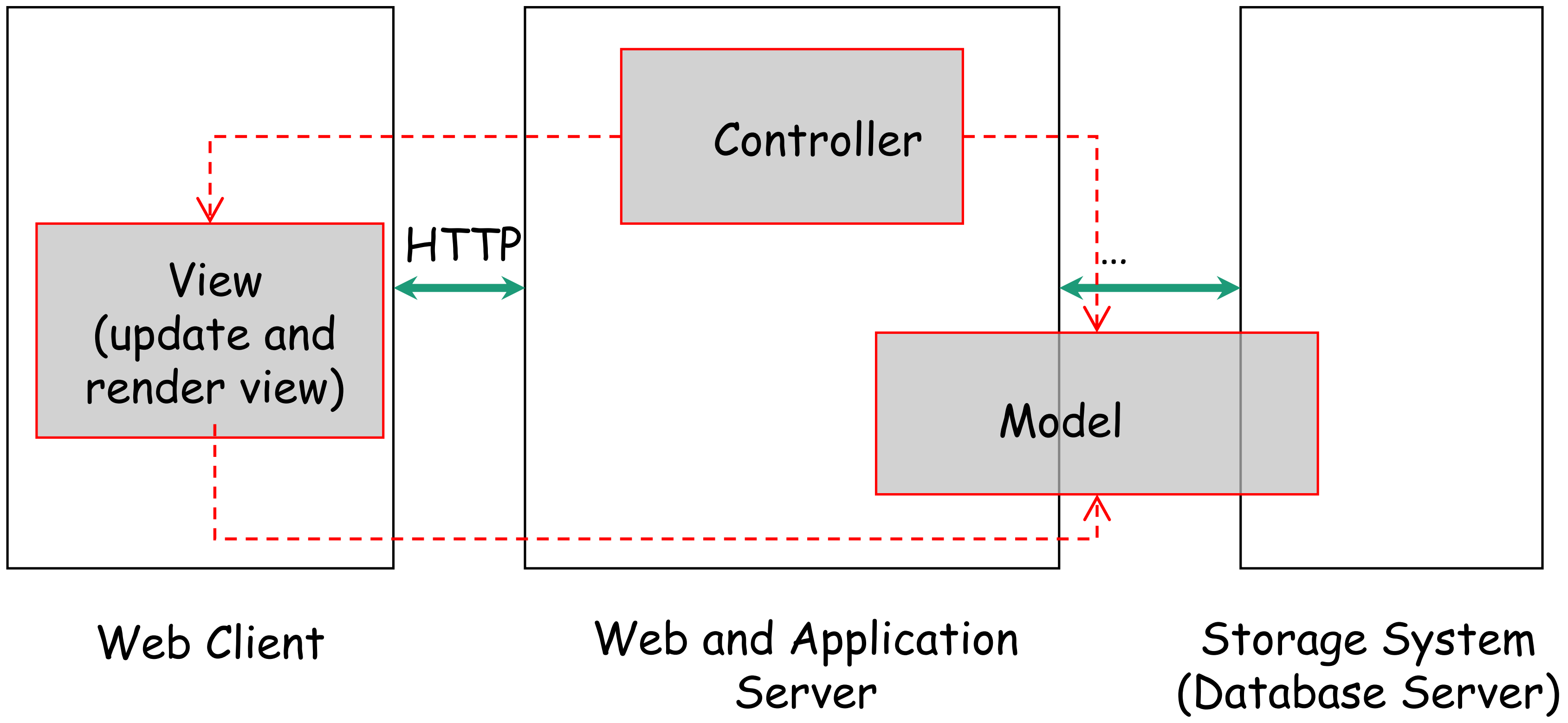
Update and Render View

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



Update and Render View

- Update and render view at the client



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
- Coined by Roy Fielding in his PhD dissertation
- 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

CRUD

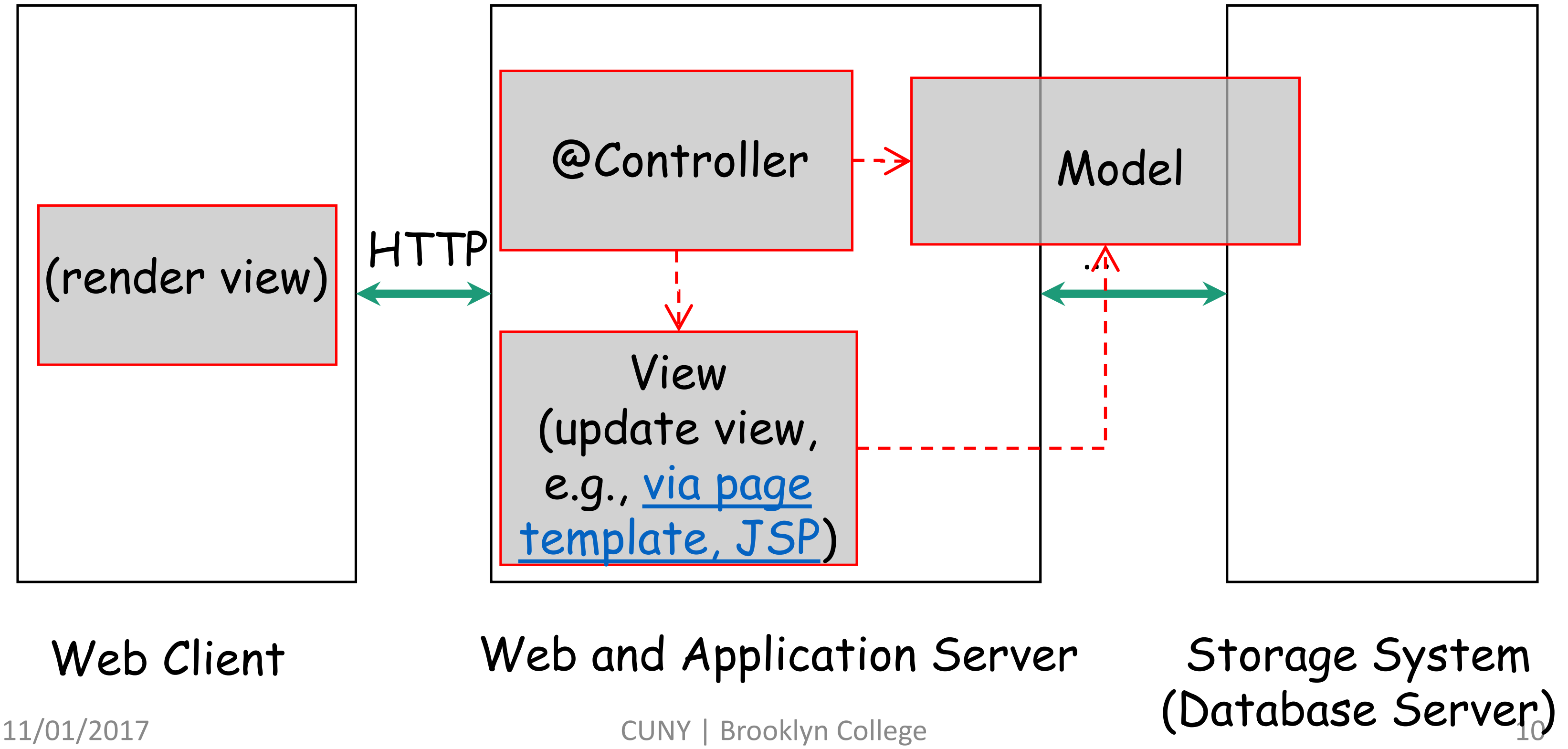
- Resources reside on storage
- Basic functions of persistent storage
  - Create, read, update, and delete
- Map to HTTP requests
  - Examples
    - PUT /addresses/1
    - GET /addresses/1
    - POST /addresses
    - DELETE /addresses/1

HTTP Status Code

- 1XX: informational
  - e.g., 101
- 2XX: success
  - e.g., 200
- 3XX: redirection
  - e.g., 301
- 4XX: client error
  - e.g., 404
- 5XX: server error
  - e.g., 503

Design with Spring Controller

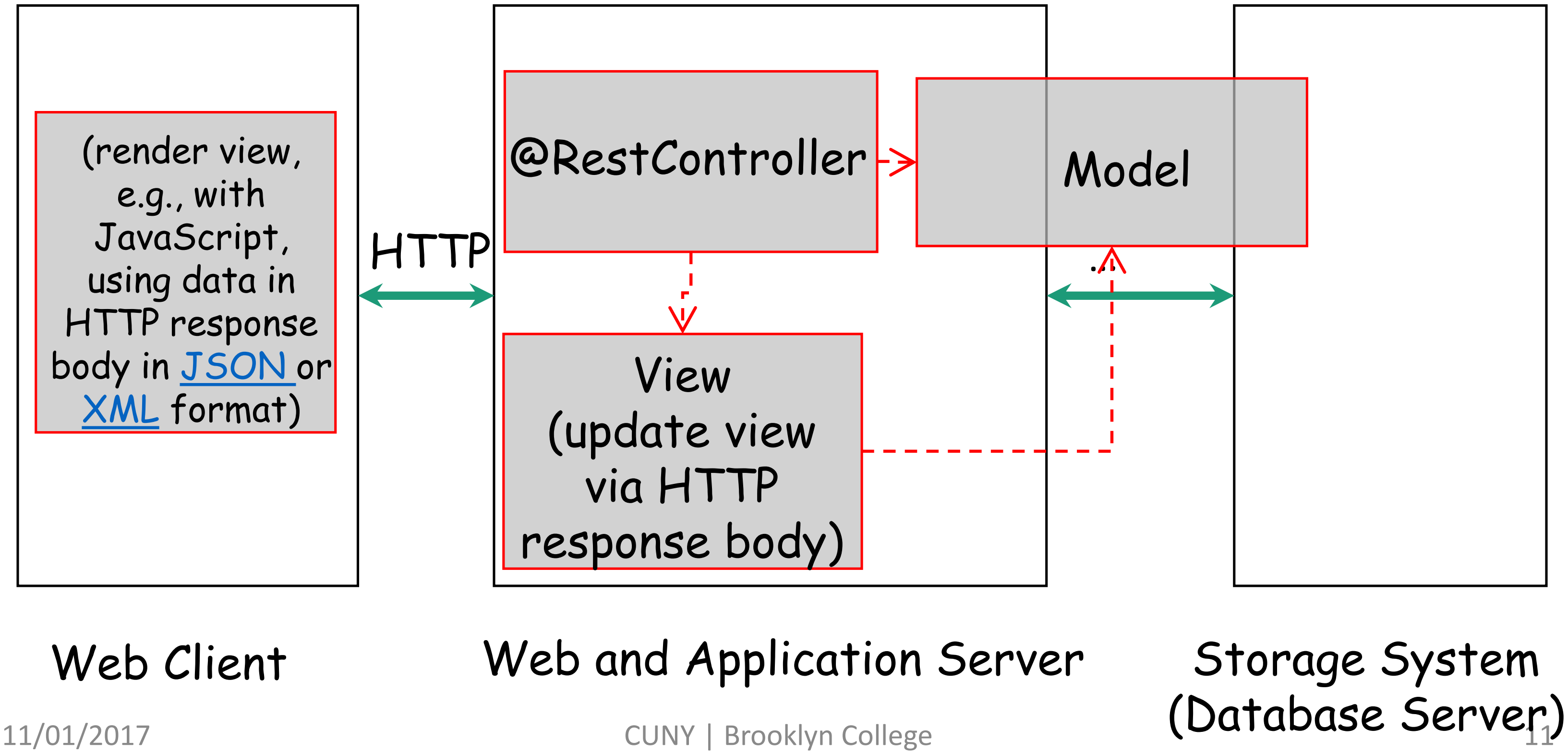
- @Controller annotate a Web Controller (often bind it with a web Model)



(render view)
HTTP
@Controller
Model
View (update view, e.g., via page template, JSP)
Web Client
Web and Application Server
Storage System (Database Server)

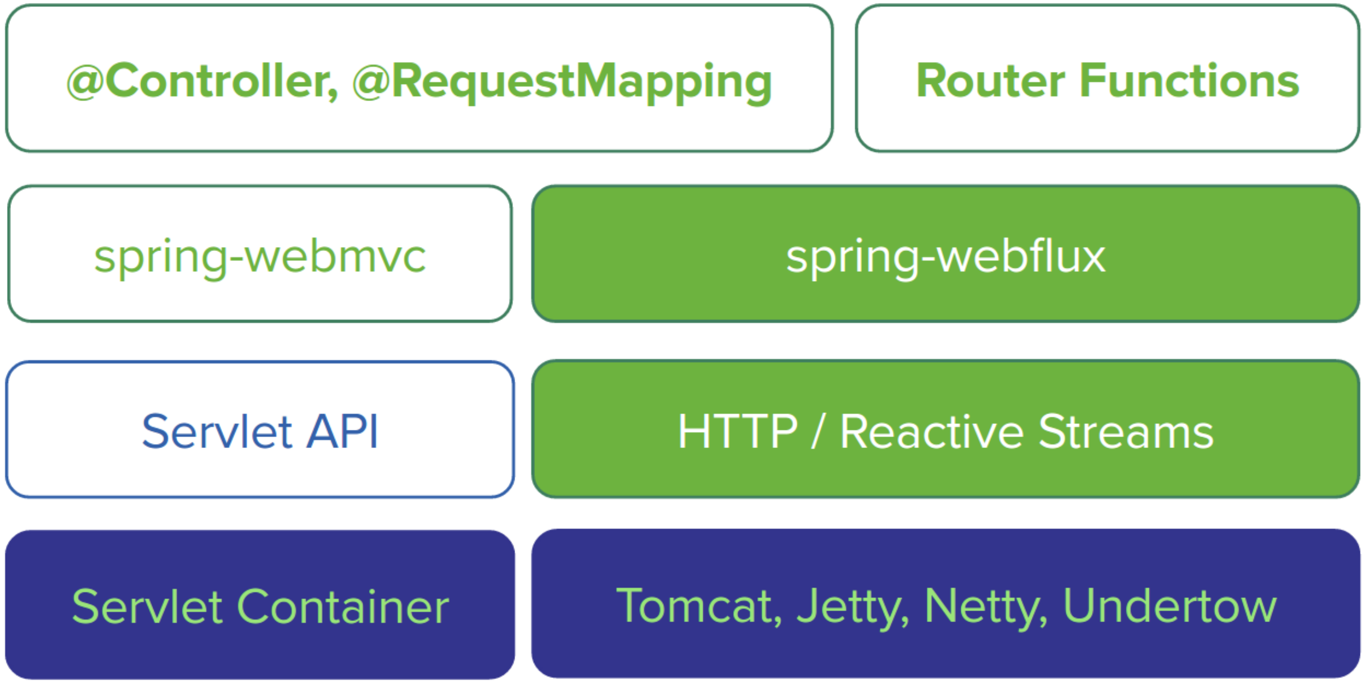
Design with Spring RestController

- @RestController annotate a RESTful Web Controller (bind it with HTTP response body)


(render view, e.g., with JavaScript, using data in HTTP response body in JSON or XML format)
HTTP
@RestController
Model
View (update view via HTTP response body)
Web Client
Web and Application Server
Storage System (Database Server)

The Spring Framework version 5

- The Spring Framework is evolving as the Web development.



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

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

HelloSpring and HelloSpringFlux

- Compare the two implemtantions
  - HelloSpring
  - HelloSpringFlux
  - Both at the web directory of the sampleprograms repository

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"