CISC 3120
C24: The Spring Framework: A Brief Introduction

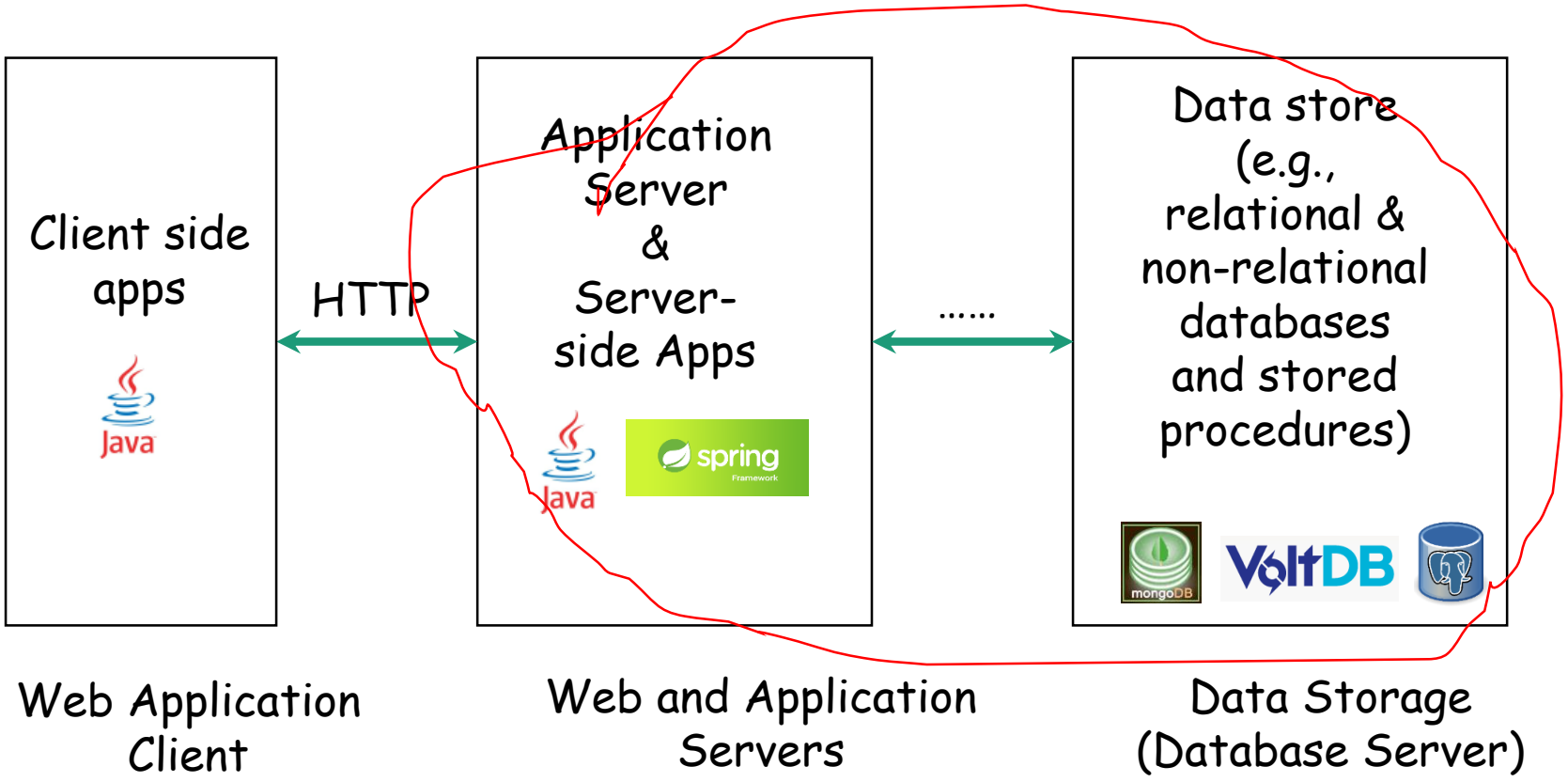
Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Web Application Architecture

- 3-tier (and n-tier)



Web Development: Some Challenges

- User agents have different set of capabilities
- Lots of libraries, technologies, languages, and APIs
- Distributed on the network

Web Application Framework

- Collection of applications, libraries, and APIs to support large web application development
- Example
  - For Java
    - The Spring framework
    - Apache Wicket
    - …

The Spring Framework

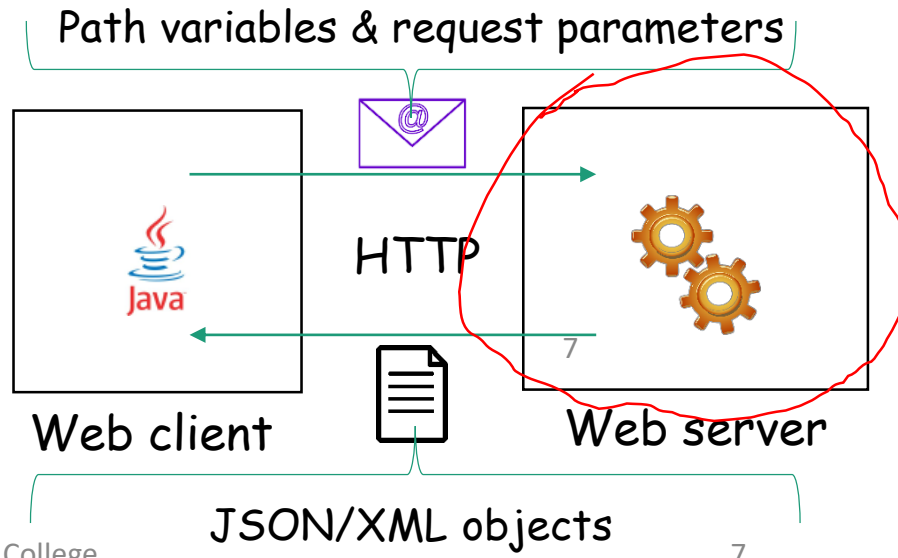
- Aimed at building multiple tier applications in Java
- Incorporate many design patterns
  - Factory pattern
  - MVC pattern
  - …
- Support automated testing

Main Components

- Core: IoC container, Events, Resources, i18n, Validation, Data Binding, Type Conversion, SpEL, AOP.
- Testing: Mock objects, TestContext framework, Spring MVC Test, WebTestClient.
- Data Access: Transactions, DAO support, JDBC, ORM, Marshalling XML.
- Web Servlet: Spring MVC, WebSocket, SockJS, STOMP messaging.
- Web Reactive: Spring WebFlux, WebClient, WebSocket.
- Integration: Remoting, JMS, JCA, JMX, Email, Tasks, Scheduling, Cache.
- Languages: Kotlin, Groovy, Dynamic languages.

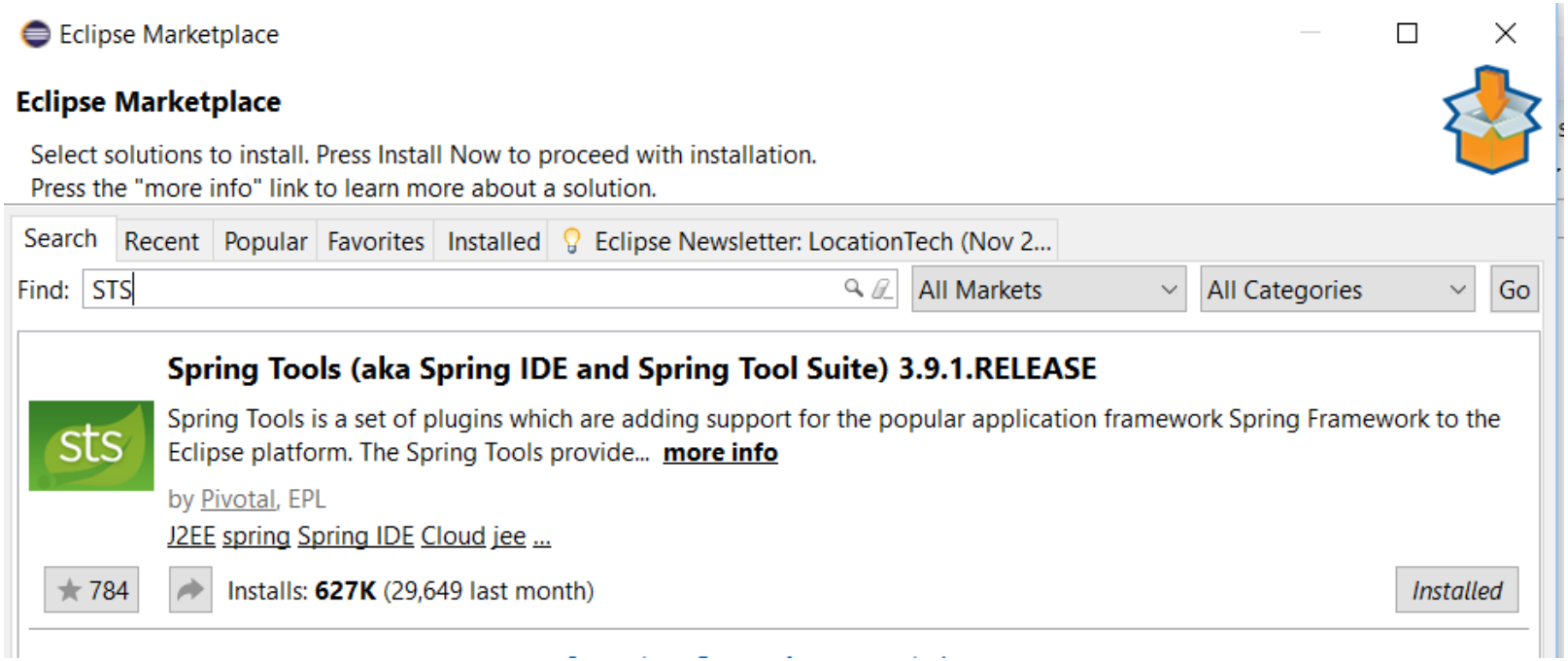
How do we begin?

- Design and build simple Web services
  - Design API
  - Design Web service
  - Test and design a full application (user agent + Web service)



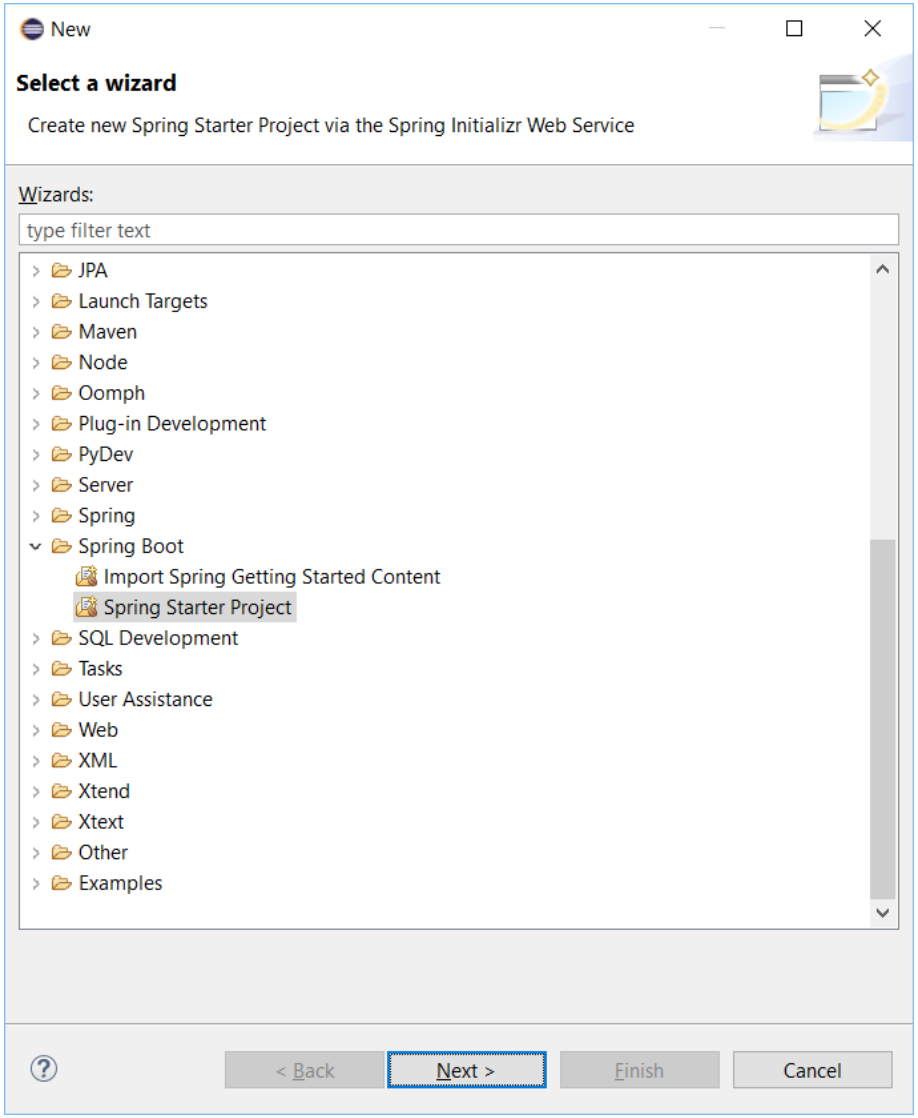
Spring Tool Suite (STS)

- Eclipse → Help → Eclipse Marketplace → Search "STS" → Install



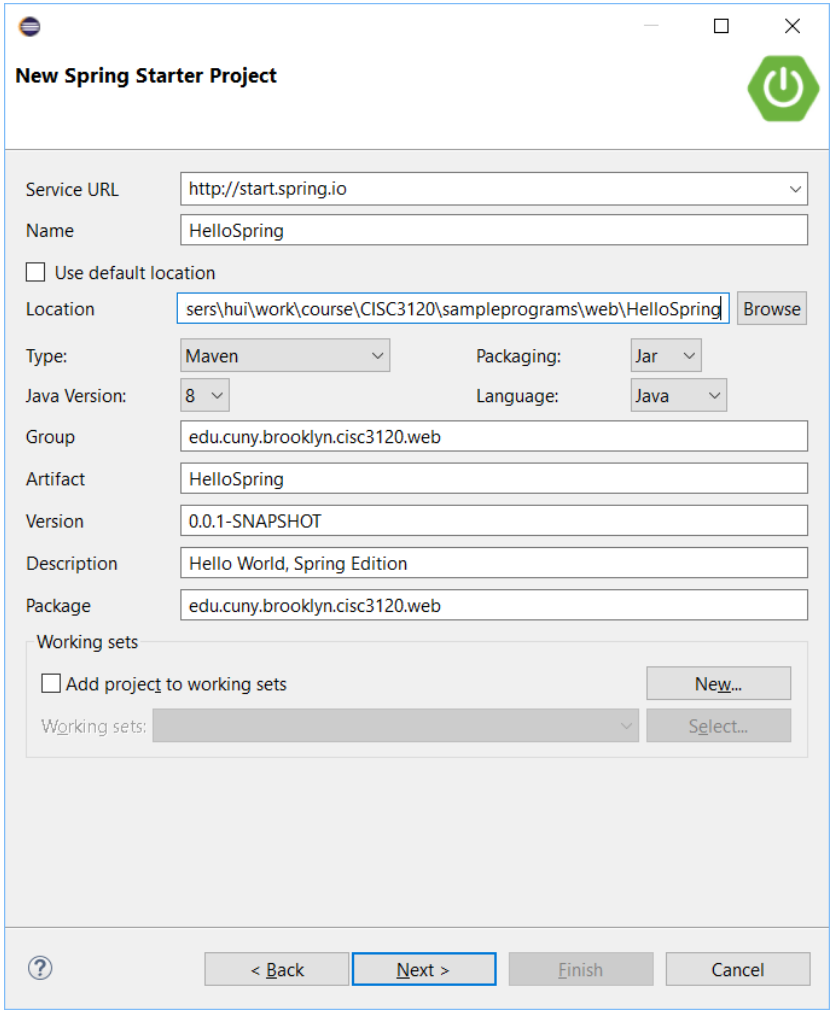
Hello, Spring

- Create the project
- Spring Boot → Spring Starter Project



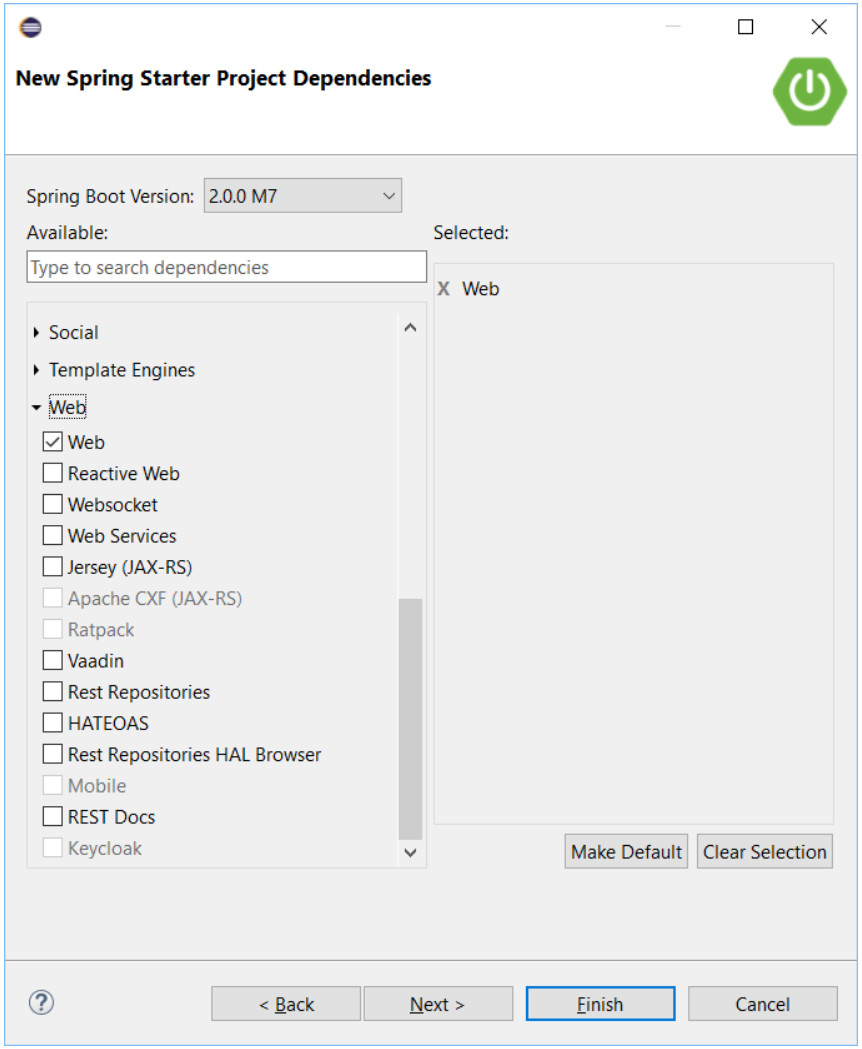
Set up Maven Project

- In this class (CISC 3120), we use Maven.
- Other choices of build automation system are also available, such as, Gradle.



Select Dependencies

- Spring has many components.
- Select necessary dependencies
  - Recommend
    - Web
    - DevTools
    - Actuators



Controller and Runner

- Refer to the Starter Guide for additional discussion: https://spring.io/guides/gs/spring-boot/
- Add the HelloController class
- Add the commandLineRunner method to HelloSpringApplication class
  - Not necessary, just to provide the means to investigate the application

Inspect Network State in the OS

- Upon running the application, inspect network status on the host

```
C:> netstat -anp tcp

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING
```

Inspect Network State from Spring Server App Output

- Upon running the application, inspect application output

```
Mapped "{[/HttpUrlConnectionApiDoc]}" onto …… HelloController.apiDoc()
Mapped "{[/apiDoc]}" onto …… HelloController.apiDoc(java.lang.String)
Mapped "{[/hello]}" onto …… HelloController.hello()
Mapped "{[/error]}" onto ……
Mapped "{[/error],produces=[text/html]}" onto ……
Mapped "{[/actuator/health],methods=[GET],produces=[application/vnd.spring-
boot.actuator.v2+json || application/json]}" onto ……
Mapped "{[/actuator/info],methods=[GET],produces=[application/vnd.spring-
boot.actuator.v2+json || application/json]}" onto ……
Mapped "{[/actuator],methods=[GET],produces=[application/vnd.spring-boot.actuator.v2+json ||
application/json]}" onto ……
```

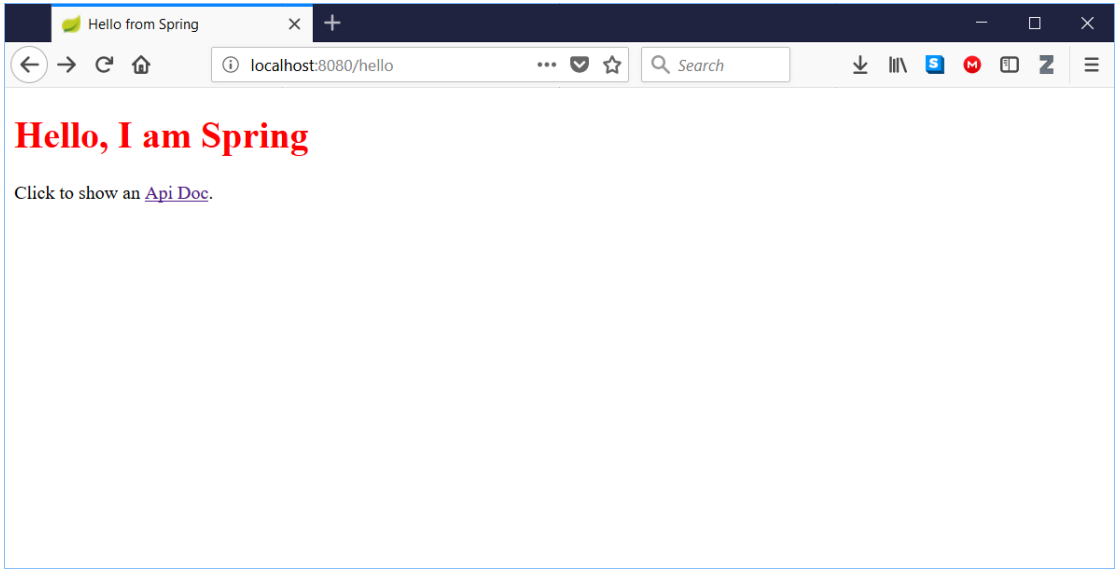
Inspect Included Components

- Upon running the application, inspect printPropertiesOfBeans method's output

```
Let's inspect the beans provided by Spring Boot:
auditEventRepository
auditEventsEndpoint
auditEventsJmxEndpointExtension
auditEventsWebEndpointExtension
auditListener
……
```

Invoke the Web Service

- Point a user agent to the URL
  - http://localhost:8080/hello
  - Or a few other URLs based on the mappings



JUnit Test in Spring

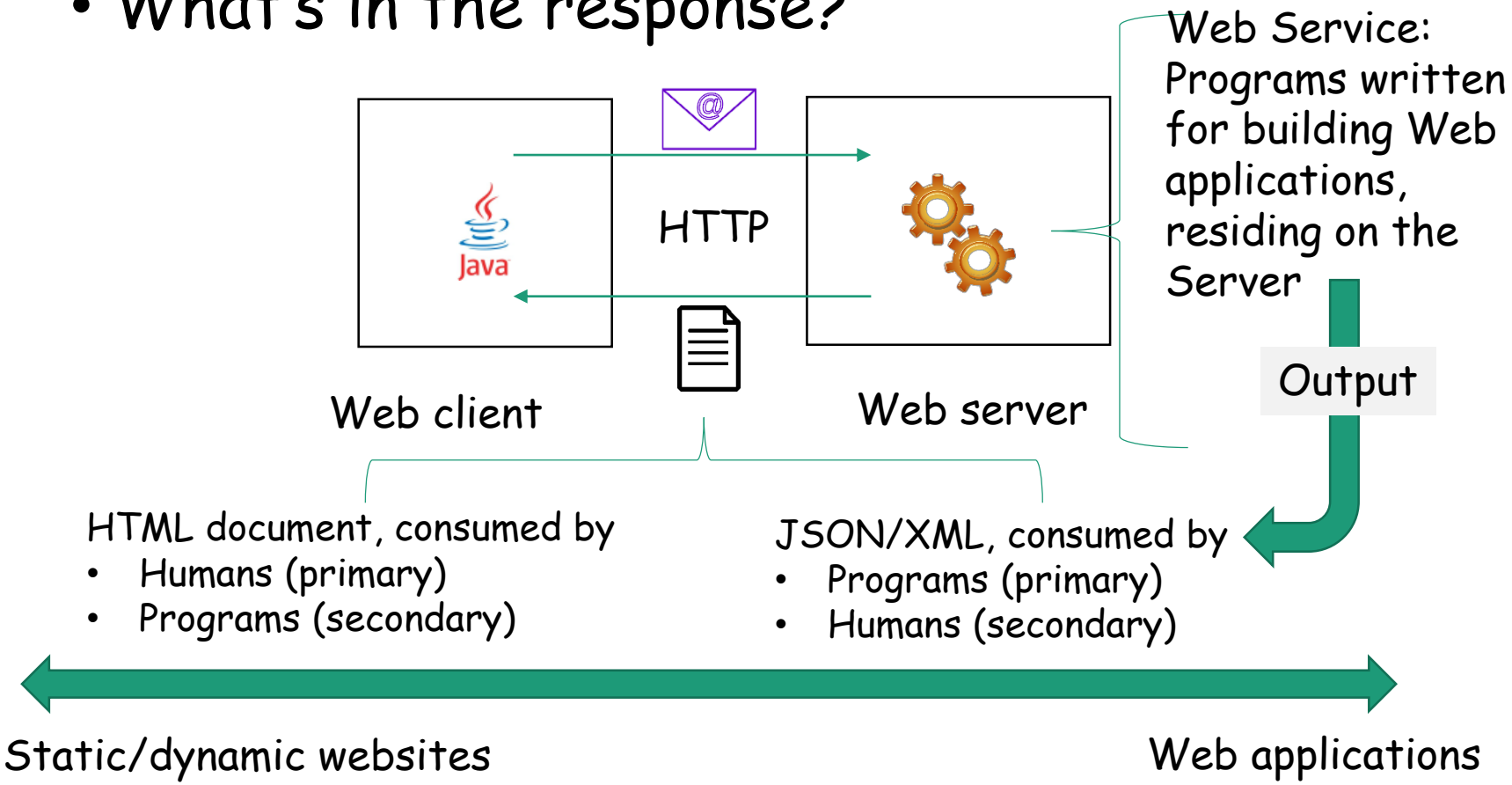
- Spring supports automatic unit testing
- See HelloControllerTest.java
- Discuss later

Questions?

- Building a simple Web server application
  - The server side of the application
- Using the Spring framework with the Spring Tool Suite (STS) in Eclipse
  - Software installation
  - Project setup
  - Run application
  - Examine application status

Recall: Web Service and User Agent

- What's in the response?



Building RESTful Web Service

- Define API
  - Determine path variables or request parameters
    - What arguments/data to pass to the Web service (class and method)
  - Determine return value (as JSON object or array)

URL for the Web Service

- Consider the Web API as
  - http://our-host/greeting?name=value

JSON Object to Return

- As an example, we expect the Web service returns,

```
{
    "id": 1,
    "content": "Hello, World!"
}
```

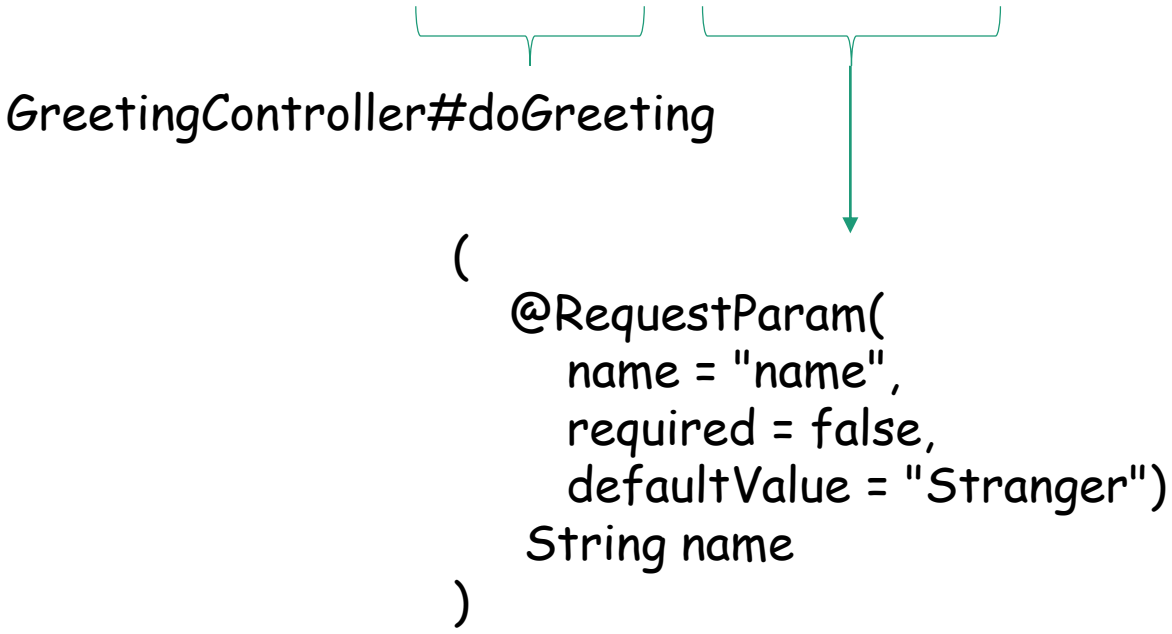
- Create a class represent this type of objects
  - The Greeting class

Create a Spring MVC Controller

- Also called the Resource Controller
- Name the class as GreetingController
  - Annotate the class with @Controller
  - Map the URL of the Web API
    - http://our-host/greeting
  - To a method in the controller
    - doGreeting(…)
  - By
    - @GetMapping("/greeting")

Passing Arguments the Remote Method

- URL of the Web API
  - http://our-host/greeting?name=value



Return JSON Object

- Annotate the method with @ResponseBody
  - The method should not return a "view" (a HTML page)
  - Instead, it should write the returned object (the Greeting object) to the response body
  - Spring can be configured to convert the object to some format, e.g., JSON
    - The Maven project setup determines the output should be a JSON object/array

Examining the Web API

- Let's try
  - URL
    - http://localhost:8080/greeting
    - http://localhost:8080/greeting?name=Jane%20Doe
    - http://localhost:8080/greeting?name=%22Jane%20Doe%22
- How about write a Java client?

Questions?

- Hello, World in Spring Framework
  - RESTful web service
  - Design a Web API
    - URL
    - Passing arguments
    - Return objects
- Return JSON instead of HTML