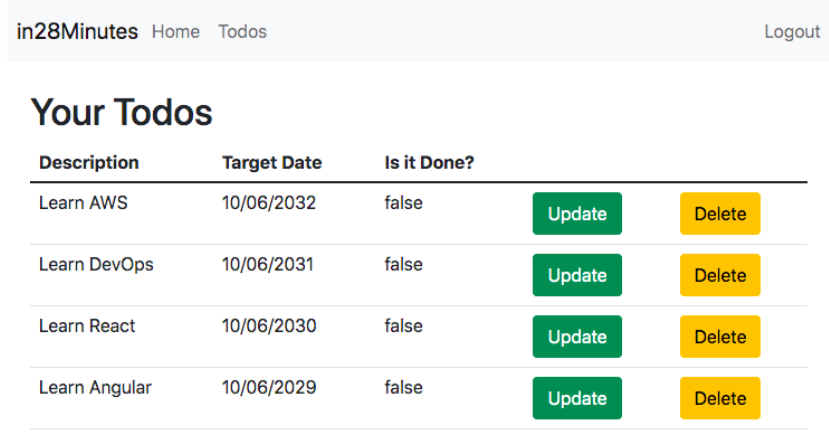
In28 Minutes

Spring Boot Master Class

Building Your First Web Application

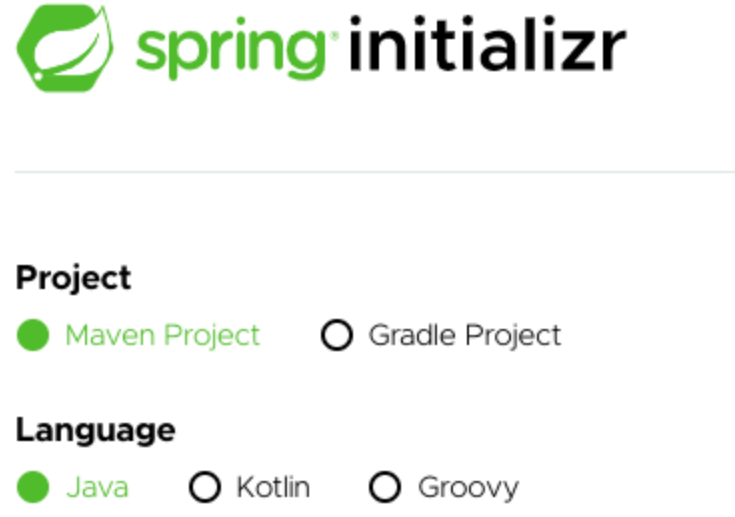
- Building **Your First Web Application** can be complex:
  - Web App concepts (Browser, HTML, CSS, Request, Response, Form, Session, Authentication)
  - Spring MVC (Dispatcher Servlet, View Resolvers, Model, View, Controller, Validations ..)
  - Spring Boot (Starters, Auto Configuration, ..)
  - Frameworks/Tools (JSP, JSTL, JPA, Bootstrap, Spring Security, MySQL, H2)
- **Goal**: Build Todo Management Web App with a **Modern Spring Boot** Approach
  - **AND** explore all concepts in a **HANDS-ON way**



Spring Initializr

In28 Minutes

- My favorite place on the internet
- Easiest way to create Spring Boot Projects
- Remember:
  - 1: SpringBoot: Use **latest released** version
    - Avoid M1,M2,M3, SNAPSHOT!
  - 2: Java: Use **latest** Version
    - Java uses 6 month release patterns
    - Spring Boot 3.0+ works on Java 17+
  - 3: Use **latest** Eclipse Java EE IDE version



Understanding Logging

In28 Minutes

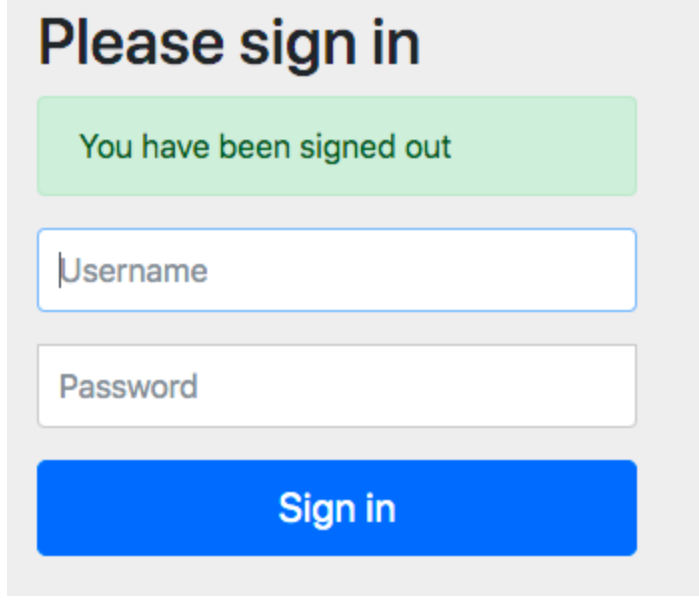
```
logging.level.some.path=debug
logging.level.some.other.path=error
logging.file.name=logfile.log

private Logger logger = LoggerFactory.getLogger(this.getClass());
logger.info("postConstruct");
```

- **Knowing what to log** is an essential skill to be a great programmer
- Spring Boot makes logging easy
  - `spring-boot-starter-logging`
- **Default**: Logback with SLF4j
- Typical Log Levels: ERROR, WARN, INFO, DEBUG, or TRACE

Session vs Request Scopes

In28 Minutes

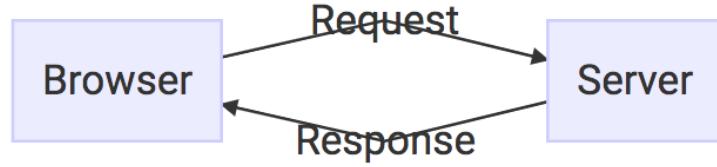
- All requests from browser are handled by our web application deployed on a server
- **Request Scope**: Active for a single request **ONLY**
  - Once the response is sent back, the request attributes will be removed from memory
  - These cannot be used for future requests
  - Recommended for most use cases
- **Session Scope**: Details stored across multiple requests
  - Be careful about what you store in session (Takes additional memory as all details are stored on server)



How does Web work?

In28 Minutes

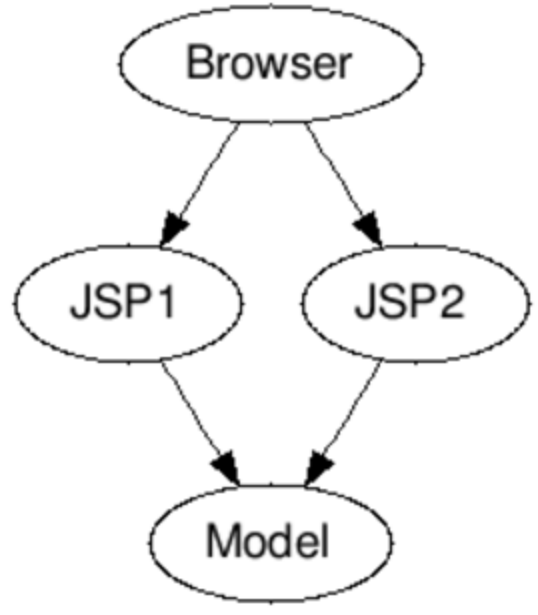
- **A:** Browser sends a request
  - HttpRequest
- **B:** Server handles the request
  - Your Spring Boot Web Application
- **C:** Server returns the response
  - HttpResponse



Peek into History - Model 1 Arch.

In28 Minutes

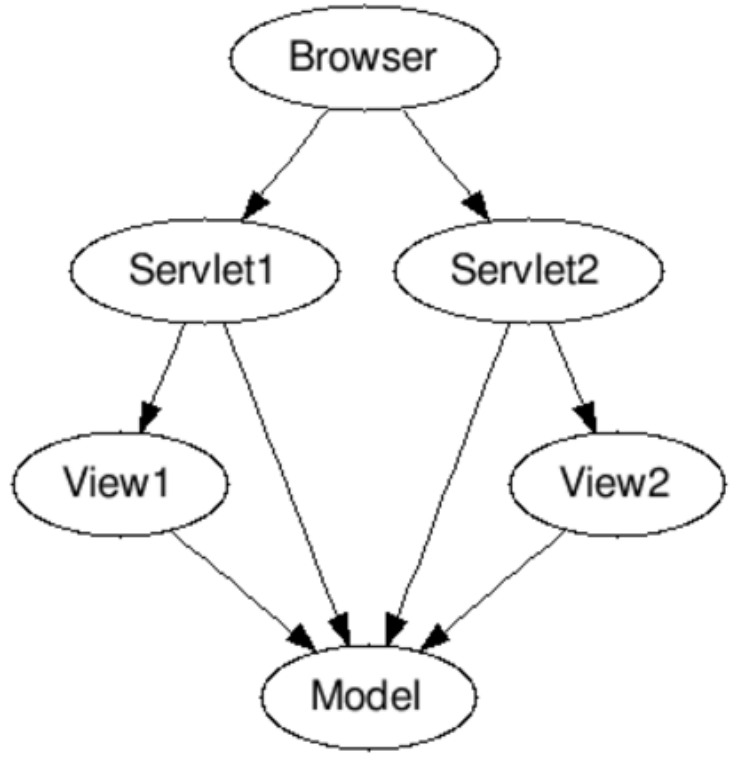
- **ALL CODE** in Views (JSPs, ...)
  - View logic
  - Flow logic
  - Queries to databases
- **Disadvantages**:
  - VERY complex JSPs
  - ZERO separation of concerns
  - Difficult to maintain



Peek into History - Model 2 Arch.

In28 Minutes

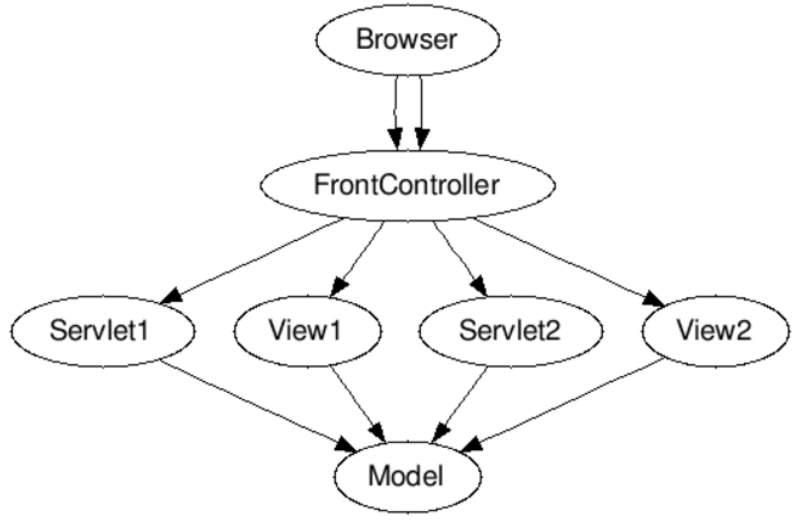
- How about separating concerns?
  - **Model**: Data to generate the view
  - **View**: Show information to user
  - **Controller**: Controls the flow
- **Advantage**: Simpler to maintain
- **Concern**:
  - Where to implement common features to all controllers?



Model 2 Architecture - Front Controller

In28 Minutes

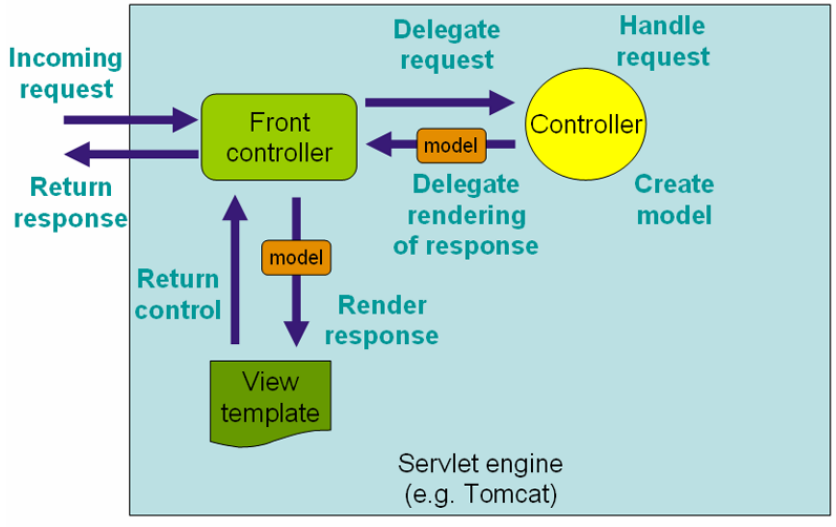
- **Concept**: All requests flow into a central controller
  - Called as **Front Controller**
- Front Controller controls flow to Controller's and View's
  - Common features can be implemented in the Front Controller



Spring MVC Front Controller - Dispatcher Servlet

In28 Minutes

- **A:** Receives HTTP Request
- **B:** Processes HTTP Request
  - **B1:** Identifies correct Controller method
    - Based on request URL
  - **B2:** Executes Controller method
    - Returns Model and View Name
  - **B3:** Identifies correct View
    - Using ViewResolver
  - **B4:** Executes view
- **C:** Returns HTTP Response



Validations with Spring Boot

In28 Minutes

- 1: Spring Boot Starter Validation
  - pom.xml
- 2: Command Bean (Form Backing Object)
  - 2-way binding (todo.jsp & TodoController.java)
- 3: Add Validations to Bean
  - Todo.java
- 4: Display Validation Errors in the View
  - todo.jsp

Slides For Future

Understanding Component Scan

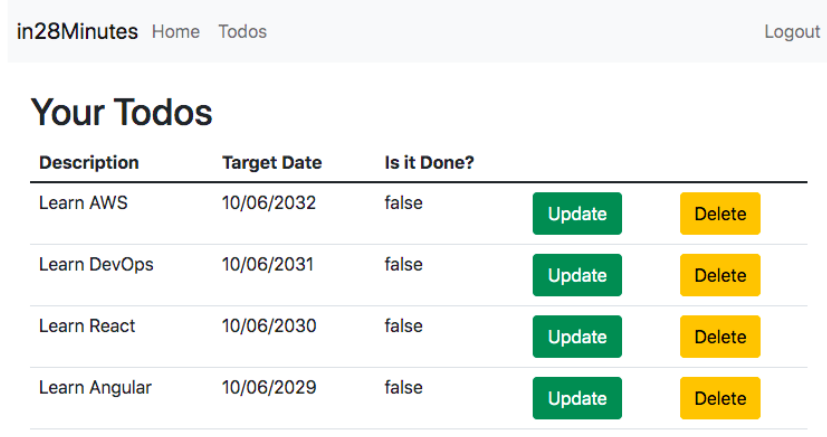
In28 Minutes

```
@SpringBootApplication
@ComponentScan("com.in28minutes.springboot.package1")
@ComponentScan({"com.in28minutes.springboot.package1"
                    ,"com.in28minutes.springboot.package2"})
```

- You can **define components** using these annotations
  - `@Component, @Controller, @Repository, @Service`
  - **How does Spring Framework know which Packages to search** for components?
    - Component Scan
- Understanding Component Scan helps you to **debug problems** (with Spring and Spring Boot)
  - `@SpringBootApplication` defines an automatic component scan
  - You can customize it by defining additional `@ComponentScan`
  - **Example Errors**:
    - My URL is not working
    - No qualifying bean of type found

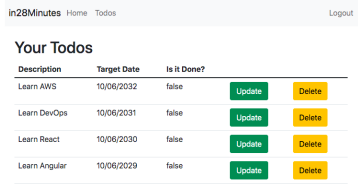
Peek into History - Jakarta EE

- **Java SE**: Java Platform, Standard Edition
  - JDK + JRE
  - Build and Run Simple Java Applications
- **Java EE (or J2EE)**: Build Web Applications
  - First released in December 1999
  - Defines Specs: Servlet, JSP, JMS, EJB, JPA, JAX-RS etc..
  - Was maintained by Oracle
- **Jakarta EE**: New form of Java EE
  - Oracle moved Java EE to Eclipse Foundation
  - AND Java EE was renamed to Jakarta EE
  - Features remain same BUT different ownership
  - Namespace move from javax to jakarta



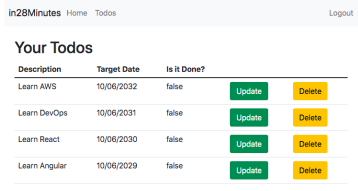
Quick Review : Web App with Spring Boot

- **HTML:** Hyper Text Markup Language
  - Tags like html, head, body, table, link are part of HTML
- **CSS:** Cascading Style Sheets
  - Styling of your web page is done using CSS
  - We used Bootstrap CSS framework
- **JavaScript:** Do actions on a web page
  - Example: Display a Date Popup (Bootstrap Datepicker)
- **JSTL**: Display dynamic data from model
  - `<c:forEach items="${todos}" var="todo">`
- **Spring form tag library**: Data binding-aware tags for handling form elements
  - `<form:form method="post" modelAttribute="todo">`



Quick Review : Web App with Spring Boot

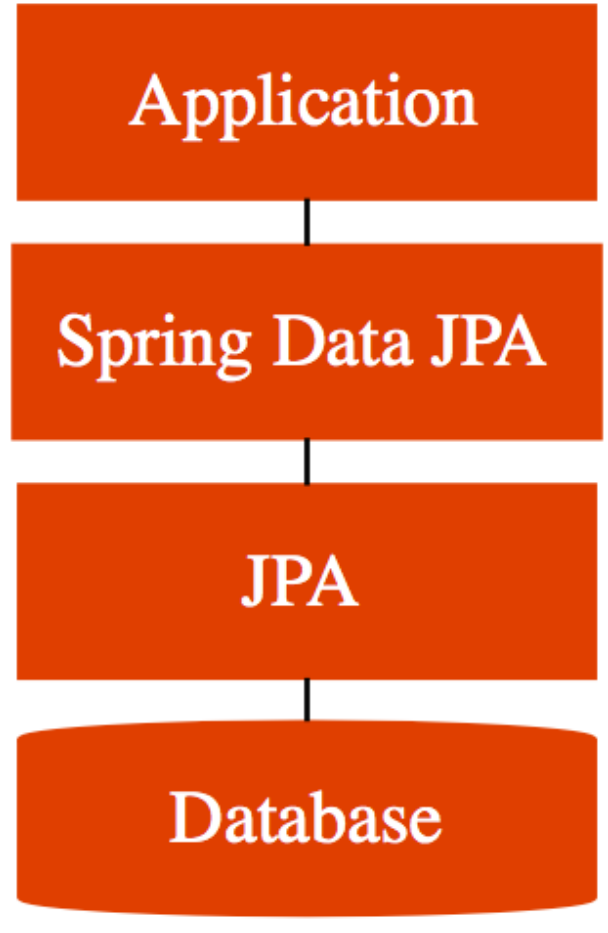
- **DispatcherServlet**: All requests flow into a central controller (Front Controller)
  - **View**: Show information to user
  - **Controller**: Controls the flow
  - **Model**: Data to generate the view
- **Spring Boot Starters**: Fast track building apps
  - Spring Boot Starter Web
  - Spring Boot Starter Validation
  - Spring Boot Starter Security
  - Spring Boot Starter Data JPA



Spring Boot Auto Configuration Magic - Data JPA

In28 Minutes

- We added Data JPA and H2 dependencies:
  - Spring Boot Auto Configuration does some magic:
    - Initialize JPA and Spring Data JPA frameworks
    - Launch an in memory database (H2)
    - Setup connection from App to in-memory database
    - Launch a few scripts at startup (example: `data.sql`)
- **Remember** - H2 is in memory database
  - Does NOT persist data
  - Great for learning
  - BUT NOT so great for production
  - Let's see how to use MySQL next!



Spring and Spring Boot Release Cycles

In28 Minutes

- What is the **difference** between these?
  - 4.2.0 (SNAPSHOT)
  - 4.4.5 (M3)
  - 4.4.0
- Release Number: MAJOR.MINOR.FIX
- Spring and Spring Boot **Release Cycle**:
  - SNAPSHOT (versions under development) > Mile Stones > Released Version
- **Recommendation** - Do NOT use SNAPSHOTs or M1 or M2 or M3
  - Prefer released versions!

Introduction to Functional Programming

In28 Minutes

- **Functional Programming**: Essential Skill for Java Programmers today
  - Lambda Functions
  - Streams
  - Filters
- **Goal**: Intro to Functional Programming
  - 7 Steps
  - ~ 30 minutes
  - Warning: **Quick** 30 minutes section

References

In28 Minutes

| Topic | Reference | |--|:--| |A|B|