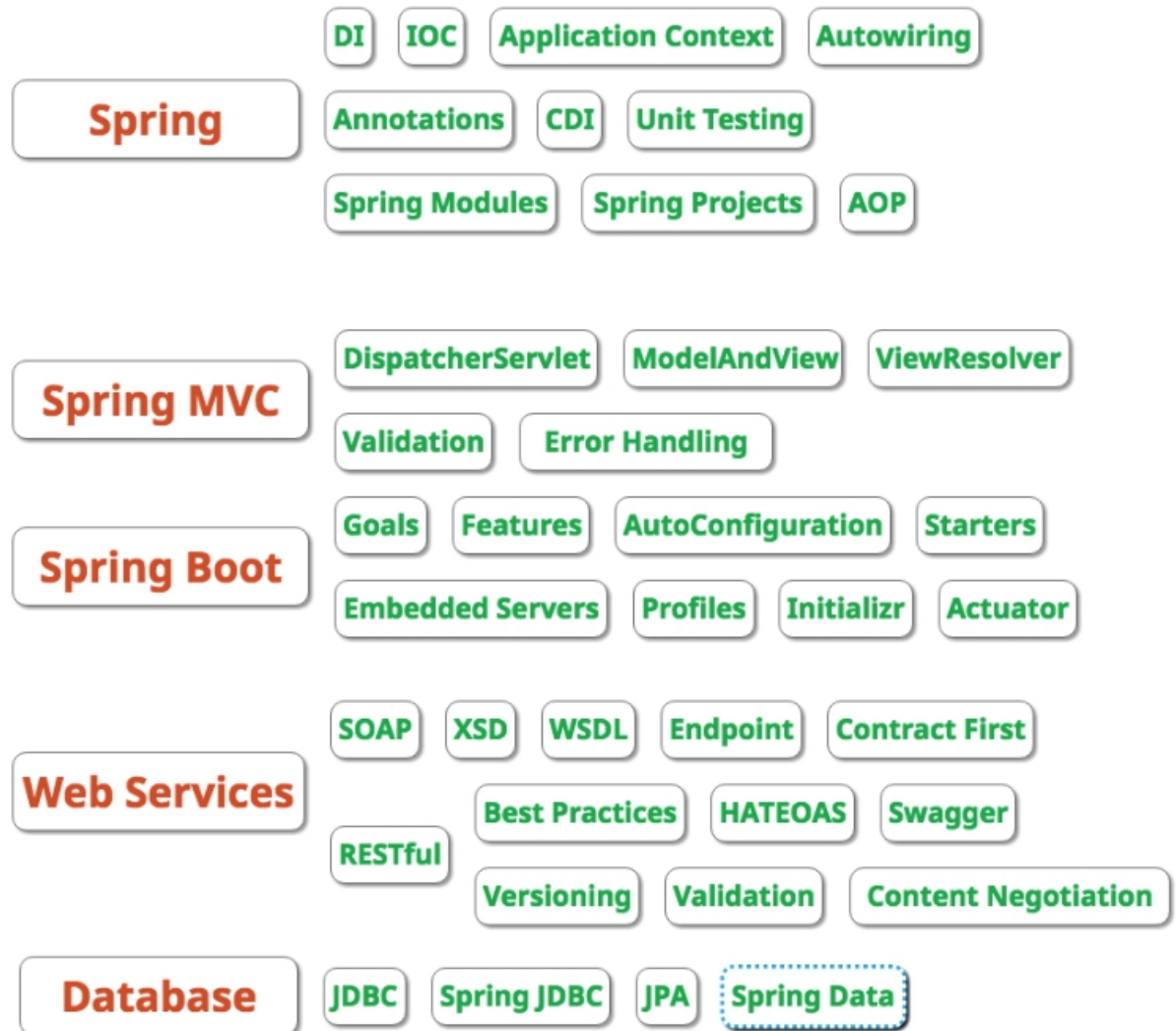
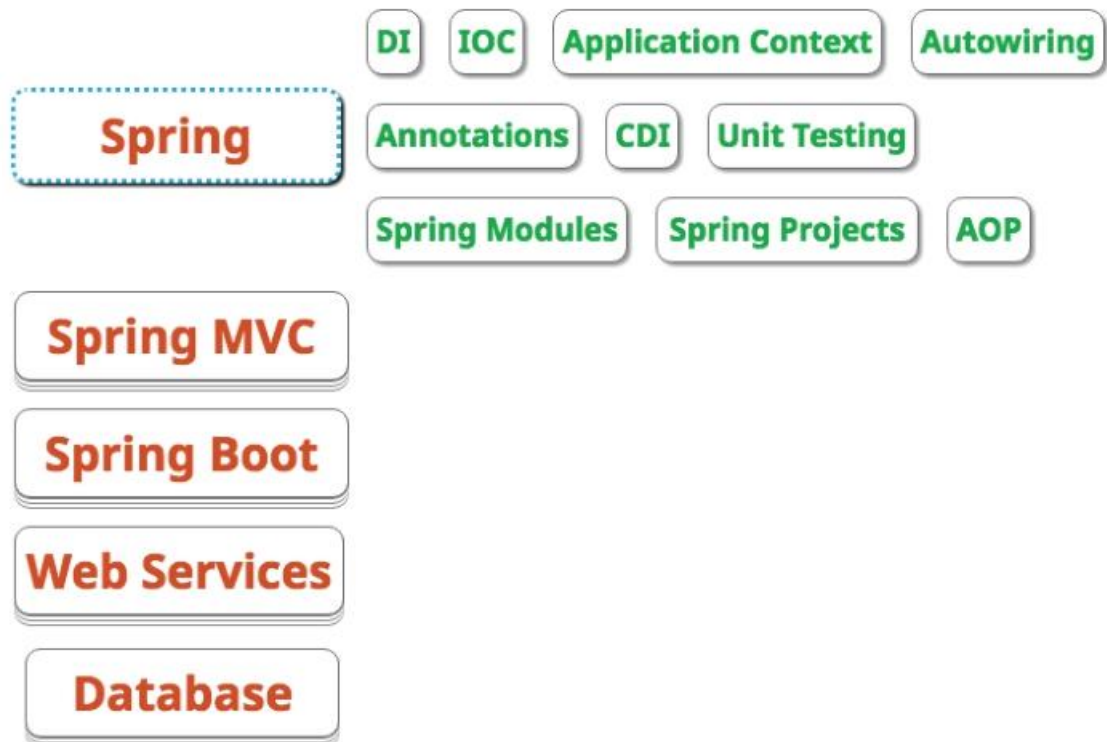
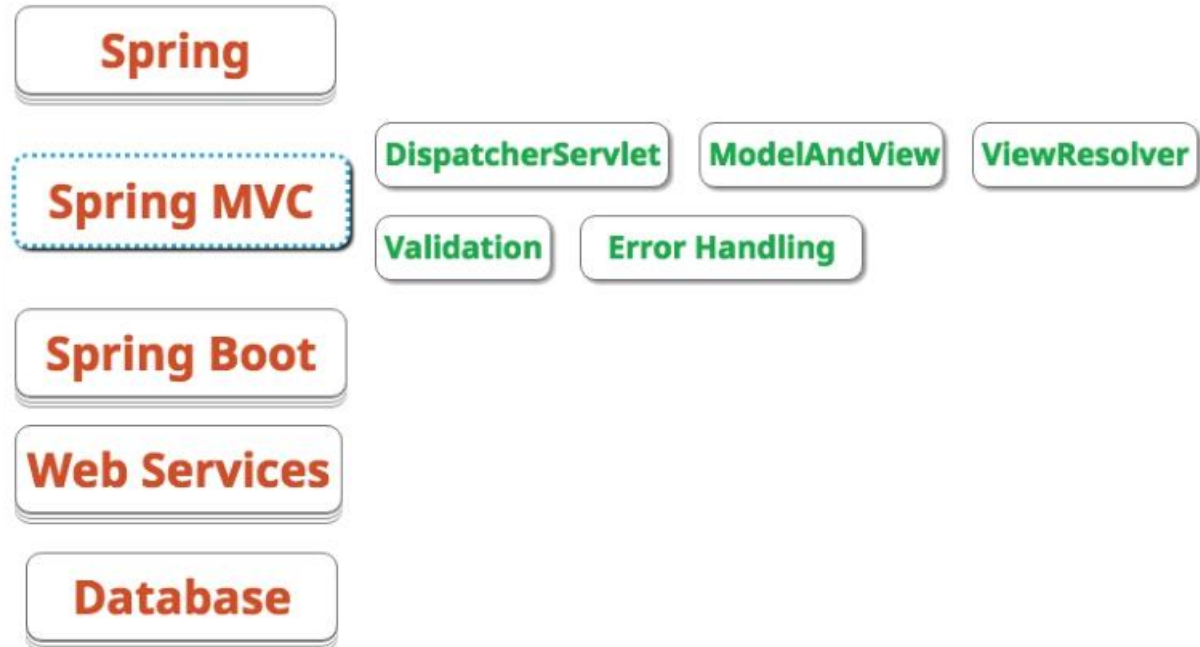


**Spring**

**Interview Questions**







**Spring**

**Spring MVC**

**Spring Boot**

**Web Services**

**Database**

Goals

Features

AutoConfiguration

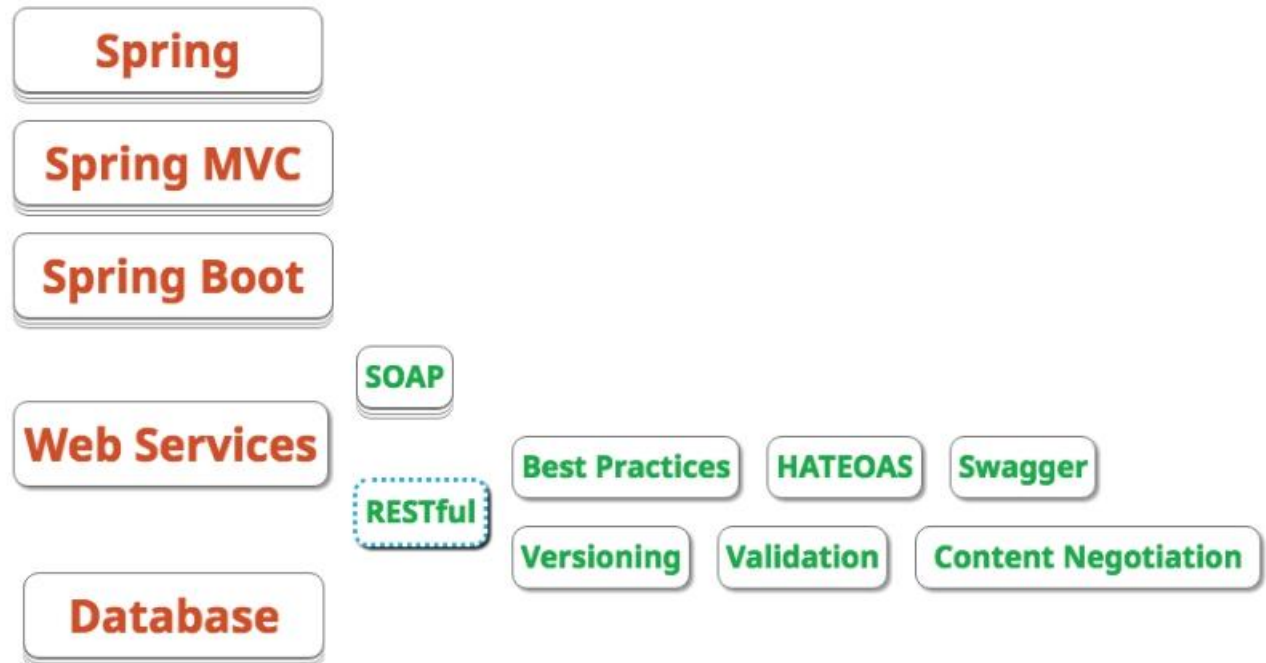
Starters

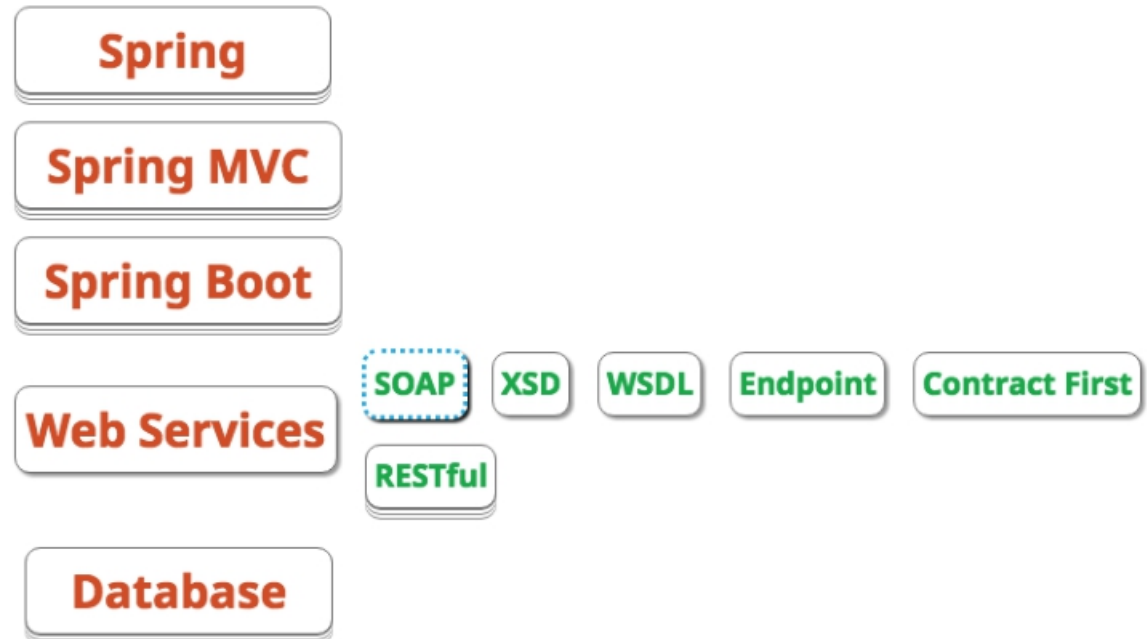
Embedded Servers

Profiles

Initializr

Actuator





**Spring**

**Spring MVC**

**Spring Boot**

**Web Services**

**Database**

JDBC

Spring JDBC

JPA

Spring Data



# Questions

- What is loose coupling?
- What is a Dependency?
- What is IOC (Inversion of Control)?
- What is Dependency Injection?
- Can you give few examples of Dependency Injection?
- What is Auto Wiring?

# Tight Coupling

```
public class TodoBusinessService {  
    TodoDataServiceImpl dataService = new TodoDataServiceImpl();  
  
public class ComplexAlgorithmImpl {  
    BubbleSortAlgorithm bubbleSortAlgorithm  
        = new BubbleSortAlgorithm();  
}
```

# Loose Coupling

@Component

```
public class TodoBusinessService {
```

@Autowired

```
    TodoDataService dataService; // = new TodoDataService()
```

```
    public TodoBusinessService(TodoDataService dataService) {  
        this.dataService = dataService;  
    }
```

```
public interface TodoDataService {  
    List<String> retrieveTodos(String user);  
}
```

# Loose Coupling

@Component

```
public class ComplexAlgorithmImpl {
```

@Autowired

```
    private SortAlgorithm sortAlgorithm;
```

```
public interface SortAlgorithm {  
    public int[] sort(int[] numbers);  
}
```

```
public class QuickSortAlgorithm implements SortAlgorithm
```

```
{ public class BubbleSortAlgorithm implements SortAlgorithm  
  
{
```

# Dependency Injection

- We use Spring Framework to instantiate beans and wire dependencies

```
ComplexAlgorithmImpl binarySearch =  
    new ComplexAlgorithmImpl(new QuickSortAlgorithm());
```

```
@Component
```

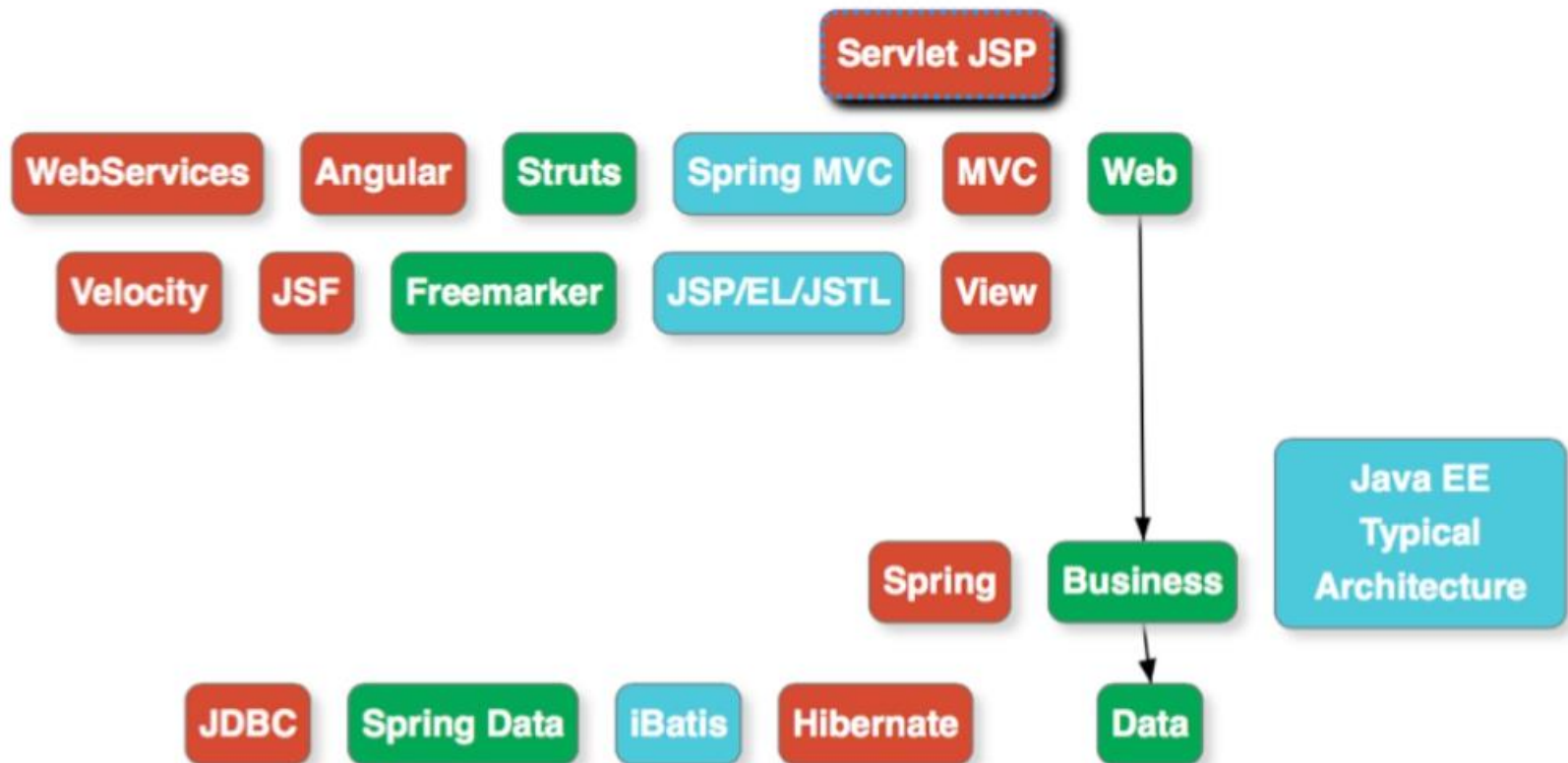
```
public class ComplexAlgorithmImpl {
```

```
    @Autowired
```

```
    private SortAlgorithm sortAlgorithm;
```

# Inversion of Control

```
public class ComplexAlgorithmImpl {  
  
    BubbleSortAlgorithm bubbleSortAlgorithm  
        = new BubbleSortAlgorithm();  
  
    @Component  
    public class ComplexAlgorithmImpl {  
  
        @Autowired  
        private SortAlgorithm sortAlgorithm;
```



# Questions

- What are the important roles of an IOC Container?
- What are Bean Factory and Application Context?
- Can you compare Bean Factory with Application Context?
- How do you create an application context with Spring?



# IOC Container

- Find Beans
- Wire Dependencies
- Manage Lifecycle of the Bean

# IOC Container

@Component

```
public class ComplexAlgorithmImpl {
```

@Autowired

```
    private SortAlgorithm sortAlgorithm;
```

@Component

```
public class QuickSortAlgorithm implements SortAlgorithm {
```

```
public interface SortAlgorithm {  
    public int[] sort(int[] numbers);  
}
```

# Application Context

- Bean Factory++
  - Spring's AOP features
  - I18n capabilities
  - WebApplicationContext for web applications etc

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/be
                        http://www.springframework.org/schema
                        http://www.springframework.org/schema
                        http://www.springframework.org/schema
</beans>
```

```
ApplicationContext context =
    new ClassPathXmlApplicationContext(
    new String[] { "BusinessApplicationContext.xml",
                  "Other-Configuration.xml" });
```

```
@Configuration  
class SpringContext {  
}
```

```
ApplicationContext ctx =  
    new AnnotationConfigApplicationContext(  
        SpringContext.class);
```

# Unit Tests

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = JavaTestContext.class)
public class DependencyInjectionJavaContextExamples {

    @RunWith(SpringJUnit4ClassRunner.class)
    @ContextConfiguration(locations = { "/TestContext.xml" })
    public class TodoBusinessTest {
```

# Questions

- How does Spring know where to search for Components or Beans?
- What is a component scan?
- How do you define a component scan in XML and Java Configurations?
- How is it done with Spring Boot?

# Java Configuration

```
@Configuration
```

```
@ComponentScan(basePackages =  
    { "com.in28minutes.spring.example1.businessservice",  
      "com.in28minutes.spring.example1.dataservice.stub" })  
class SpringContext {  
}
```



# XML Configuration

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/xsi"
  <context:component-scan
    base-package="com.in28minutes.example"/>
</beans>
```

# Questions

- What does @Component signify?
- What does @Autowired signify?
- What's the difference Between @Controller, @Component, @Repository, and @Service Annotations in Spring?

# Notes

- @Component - Spring should manage the bean.
- @Autowired - Spring should find the matching bean and wire the dependency in.

```
@Component  
public class ComplexAlgorithmImpl {
```

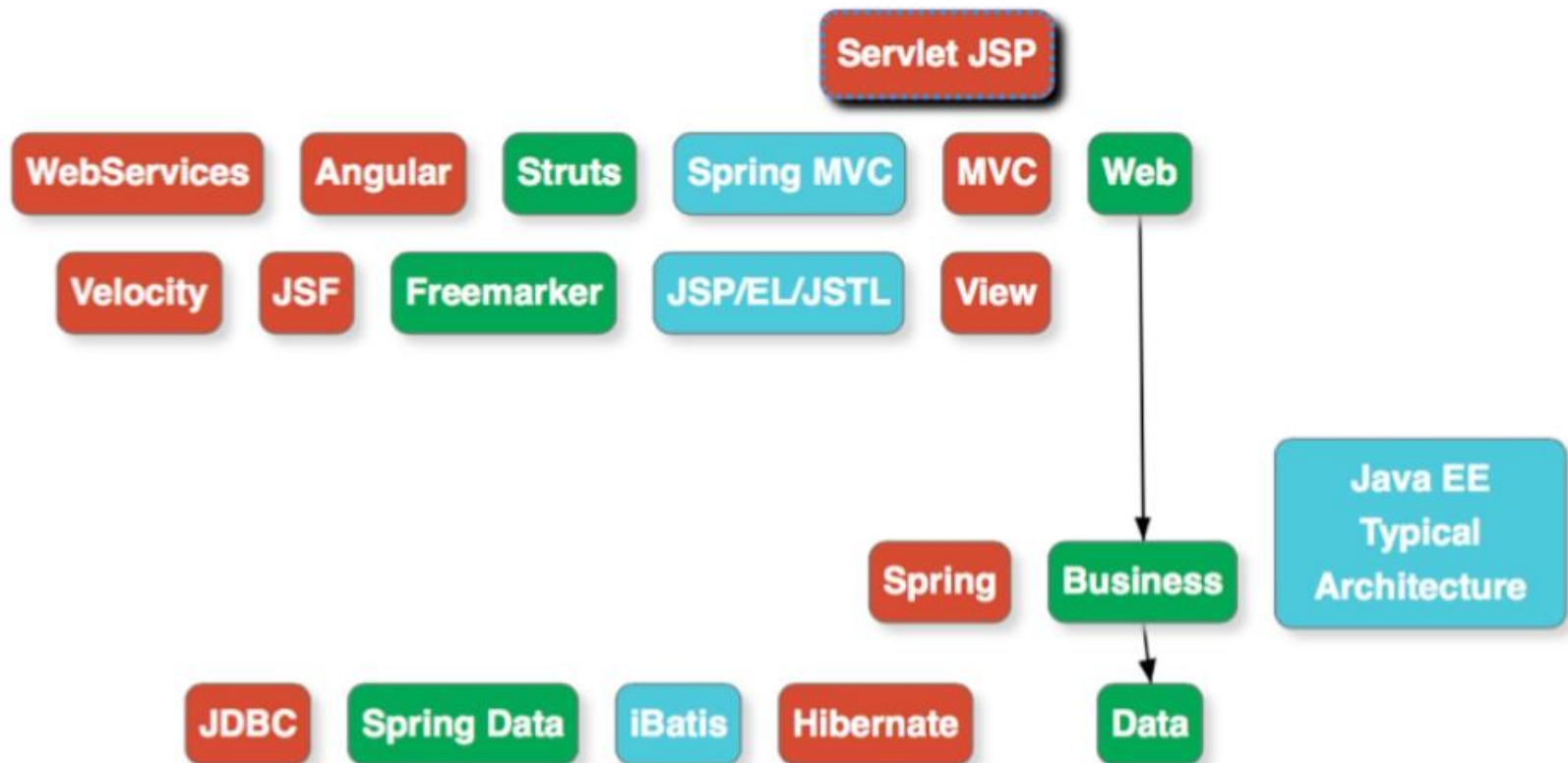
```
    @Autowired  
    private SortAlgorithm sortAlgorithm;
```

```
public interface SortAlgorithm {  
    public int[] sort(int[] numbers);  
}
```

```
@Component  
public class QuickSortAlgorithm implements SortAlgorithm {
```

# Notes

- @Component - Generic Component
- @Repository - encapsulating storage, retrieval, and search behavior typically from a relational database
- @Service - Business Service Facade
- @Controller - Controller in MVC pattern



# Questions

- What is the default scope of a bean?
- Are Spring beans thread safe?
- What are the other scopes available?
- How is Spring's singleton bean different from Gang of Four Singleton Pattern?

# Notes

- The singleton scope is the default scope in Spring.
- The Gang of Four defines Singleton as having one and only one instance per ClassLoader.
- However, Spring singleton is defined as one instance of bean definition per container.



# Other Scopes

- singleton - One instance per Spring Context
- prototype - New bean whenever requested
- request - One bean per HTTP request. Web-aware Spring ApplicationContext.
- session - One bean per HTTP session. Web-aware Spring ApplicationContext.

# Examples

@RequestScope

@Component

```
public class RequestScopedBean {
```

@SessionScope

@Component

```
public class SessionScopedBean {
```

```
<bean id="someBean" class="com.in28minutes.SomeBean"  
      scope="prototype"/>
```

# Questions

- What are the different types of dependency injections?
- What is setter injection?
- What is constructor injection?
- How do you choose between setter and constructor injections?

# Setter Injection

@Component

```
public class TodoBusinessService
```

```
{ TodoDataService dataService;
```

@Autowired

```
public void setDataService
```

```
(TodoDataService dataService) {
```

```
    this.dataService = dataService;
```

```
}
```

//Through Reflection

@Component

```
public class TodoBusinessService {
```

@Autowired

```
TodoDataService dataService;
```

# Constructor Injection

@Component

```
public class TodoBusinessService {
```

```
    TodoDataService dataService;
```

@Autowired

```
public TodoBusinessService(TodoDataService dataService) {  
    super();  
    this.dataService = dataService;  
}
```

# Constructor vs Setter Injection

- <https://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/>
- The Spring team generally advocates constructor injection as it enables one to implement application components as immutable objects and to ensure that required dependencies are not null.

# Constructor vs Setter Injection

- Furthermore constructor-injected components are always returned to client (calling) code in a fully initialized state.
  - As a side note, a large number of constructor arguments is a bad code smell.

# Constructor vs Setter Injection

- Constructor Injection for Mandatory Dependencies
- Setter Injection for Optional Dependencies



# Questions

- What are the different options available to create Application Contexts for Spring?
- What is the difference between XML and Java Configurations for Spring?
- How do you choose between XML and Java Configurations for Spring?

# XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/be
                        http://www.springframework.org/schema
                        http://www.springframework.org/schema
                        http://www.springframework.org/schema
</beans>
```

```
ApplicationContext context =
    new ClassPathXmlApplicationContext(
    new String[] { "BusinessApplicationContext.xml",
                  "Other-Configuration.xml" });
```

# Java

```
@Configuration  
class SpringContext {  
}
```

```
ApplicationContext ctx =  
    new AnnotationConfigApplicationContext(  
        SpringContext.class);
```

# **XML vs Java Configuration**

- Today, with use of annotations, the things that are specified in a Application Context are at bare minimum.
- There is little to choose between these as long as the configuration is at a bare minimum.

# XML vs Java Configuration

- With Spring Boot, we are slowly moving towards complete Java Configuration.

```
package com.in28minutes.spring.basics.springin5steps;
```

```
@SpringBootApplication
```

```
public class SpringIn5StepsApplication {
```

# Questions

- How does Spring do Autowiring?
- What are the different kinds of matching used by Spring for Autowiring?

# Autowiring

- byType
- byName
- constructor - similar to byType, but through constructor

# by Type - Class or Interface

```
@Component
```

```
public class ComplexAlgorithmImpl {
```

```
    @Autowired
```

```
    private SortAlgorithm sortAlgorithm;
```

```
public interface SortAlgorithm {  
    public int[] sort(int[] numbers);  
}
```

```
@Component
```

```
public class QuickSortAlgorithm implements SortAlgorithm {
```



# by Name

@Component

```
public class ComplexAlgorithmImpl {
```

@Autowired

```
    private SortAlgorithm quickSortAlgorithm;
```

```
public interface SortAlgorithm {  
    public int[] sort(int[] numbers);  
}
```

@Component

```
public class QuickSortAlgorithm implements SortAlgorithm {
```

@Component

```
public class BubbleSortAlgorithm implements SortAlgorithm {
```

# Questions

- How do you debug problems with Spring Framework?
  - NoUniqueBeanDefinitionException
  - NoSuchBeanDefinitionException
- What is @Primary?
- What is @Qualifier?

# No matching Components

@Component

```
public class ComplexAlgorithmImpl {
```

@Autowired

```
    private SortAlgorithm sortAlgorithm;
```

```
public interface SortAlgorithm {  
    public int[] sort(int[] numbers);  
}
```

```
public class QuickSortAlgorithm implements SortAlgorithm
```

```
{ public class BubbleSortAlgorithm implements SortAlgorithm  
  
{
```

UnsatisfiedDependencyException:  
Error creating bean with name 'binarySearchImpl':  
Unsatisfied dependency expressed through field 'sortAlgorithm'  
nested exception is org.springframework.beans.factory.:  
No qualifying bean of type  
'com.in28minutes.spring.basics.springin5steps.SortAlgorithm'  
available:  
expected at least 1 bean which qualifies as autowire candidate  
Dependency annotations:  
{ @org.springframework.beans.factory.annotation.Autowired  
(required=true) }

# Typically problems

- @Component missing
- or @ComponentScan not defined properly

# Exception - Two matching Components

@Component

```
public class ComplexAlgorithmImpl {
```

@Autowired

```
    private SortAlgorithm sortAlgorithm;
```

```
public interface SortAlgorithm {  
    public int[] sort(int[] numbers);  
}
```

@Component

```
public class QuickSortAlgorithm implements SortAlgorithm {
```

@Component

```
public class BubbleSortAlgorithm implements SortAlgorithm {
```

Field sortAlgorithm in  
springin5steps.BinarySearchImpl required a single  
bean, but 2 were found:

- bubbleSortAlgorithm: defined in file  
[BubbleSortAlgorithm.class]
- quickSortAlgorithm: defined in file  
[QuickSortAlgorithm.class]

# Primary

@Component

@Primary

```
public class BubbleSortAlgorithm implements SortAlgorithm {
```



# Qualifier

```
@Component
```

```
public class ComplexAlgorithmImpl {
```

```
    @Autowired
```

```
    @Qualifier("mainAlgorithm")
```

```
    private SortAlgorithm sortAlgorithm;
```

```
@Component
```

```
@Qualifier("mainAlgorithm")
```

```
public class BubbleSortAlgorithm implements SortAlgorithm {
```

# Questions

- What is CDI (Contexts and Dependency Injection)?
- Does Spring Support CDI?
- Would you recommend to use CDI or Spring Annotations?

# CDI

- Java EE Dependency Injection Standard (JSR-330)
- Spring Supports most annotations
  - @Inject (@Autowired)
  - @Named (@Component & @Qualifier)
  - @Singleton (Defines a scope of Singleton)

# Questions

- What are the major features in different versions of Spring?
- What are new features in Spring Framework 4.0?
- What are new features in Spring Framework 5.0?

# Notes

- Spring 2.5 made annotation-driven configuration possible.
- Spring 3.0 made great use of the Java 5 improvements in language.

# Spring 4.0

- First version to fully support Java 8 features.
- Minimum version of Java to use Spring 4 is Java SE 6.
- Introduced `@RestController` annotation
- Spring 4.1 supports JCache (JSR-107) annotations

# Spring 5.0

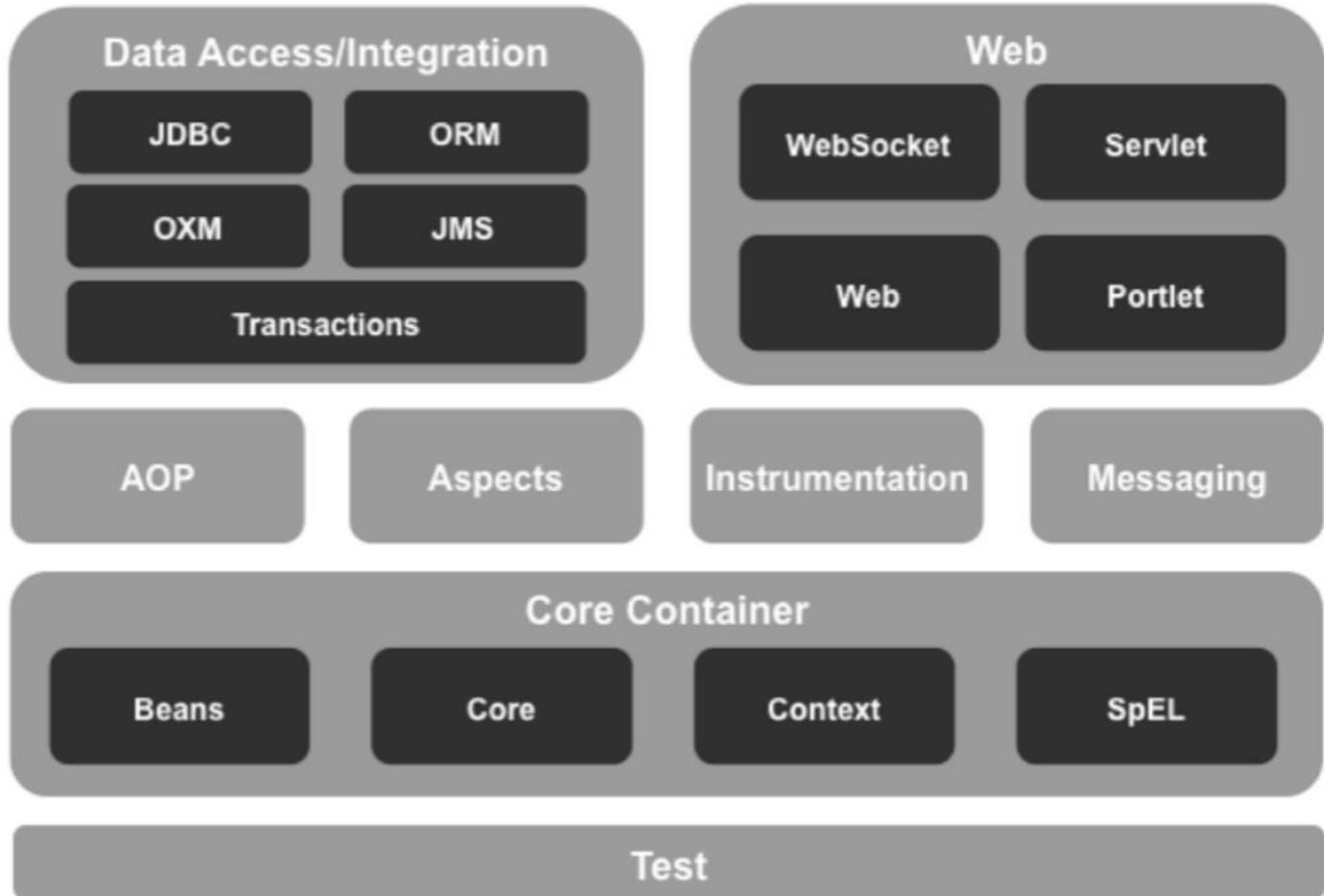
- Functional web framework
- Support for Jigsaw (Java Modularity)
- Support for reactive programming
- Support for Kotlin

# Questions

- What are important Spring Modules?



# Spring Modules



# Questions

- What are important Spring Projects?

# Spring Projects

- Spring Boot
- Spring Cloud
- Spring Data
- Spring Integration
- Spring Batch
- Spring Security
- Spring HATEOAS
- Spring Web Services
- Spring Session

# Questions

- What is the simplest way of ensuring that we are using single version of all Spring related dependencies?

# Use a BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-framework-bom</artifactId>
      <version>5.0.0.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

```
<dependencies>
  <dependency> <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
  </dependency>
  <dependency> <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
  </dependency>
</dependencies>
```

# Questions

- Name some of the design patterns used in Spring Framework?

# Design Patterns in Spring

- Front Controller - Dispatcher Servlet
- Prototype - Beans
- Dependency Injection
- Factory Pattern - Bean Factory & Application Context
- Template Method
  - `org.springframework.web.servlet.mvc.AbstractCon`

# Questions

- What are some of the important Spring annotations you have used?



# Annotations

- @Component, @Service, @Repository, @Controller
- @Autowired
- @Primary
- @Qualifier
- @Configuration

# Questions

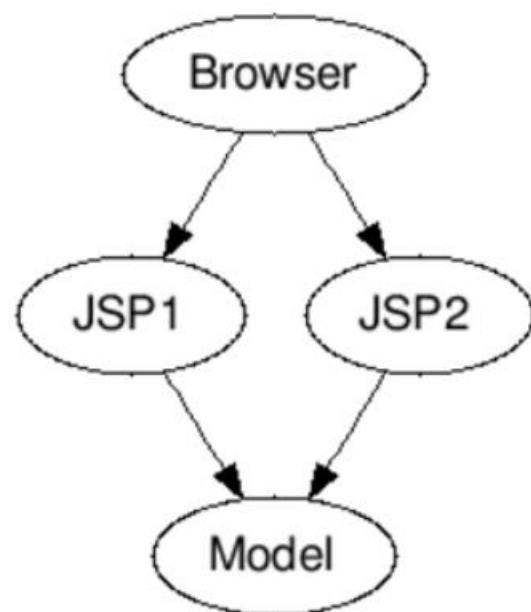
- What do you think about Spring Framework?
- Why is Spring Popular?
- Can you give a big picture of the Spring Framework?

- Architecture - Flexible & No Restrictions
- Design - Loosely Coupled
- Code - Easy Unit Testing
- Features - Dependency Injection, IOC Container(Bean Factory & Application Context), Auto wiring, Component Scan
- Spring Modules
- Spring Projects

**MVC**

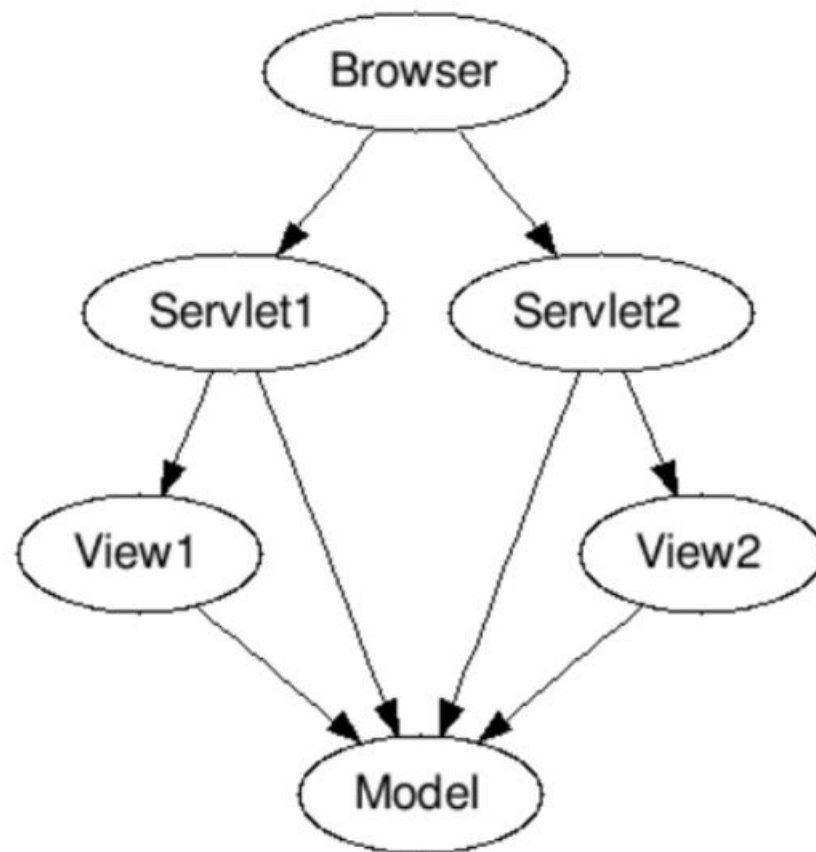
# Questions

- What is Model 1 architecture?



# Questions

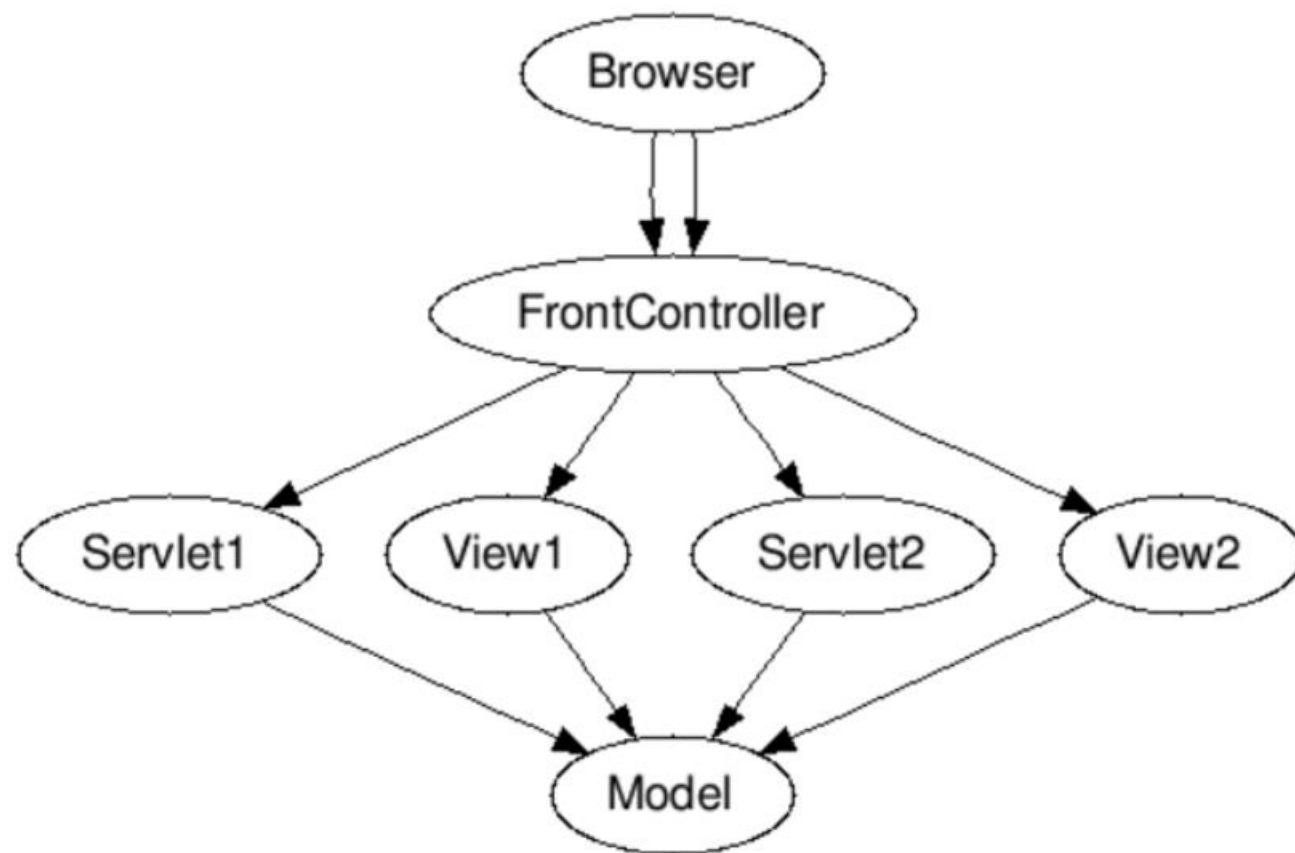
- What is Model 2 architecture?





# Questions

- What is Model 2 Front Controller architecture?

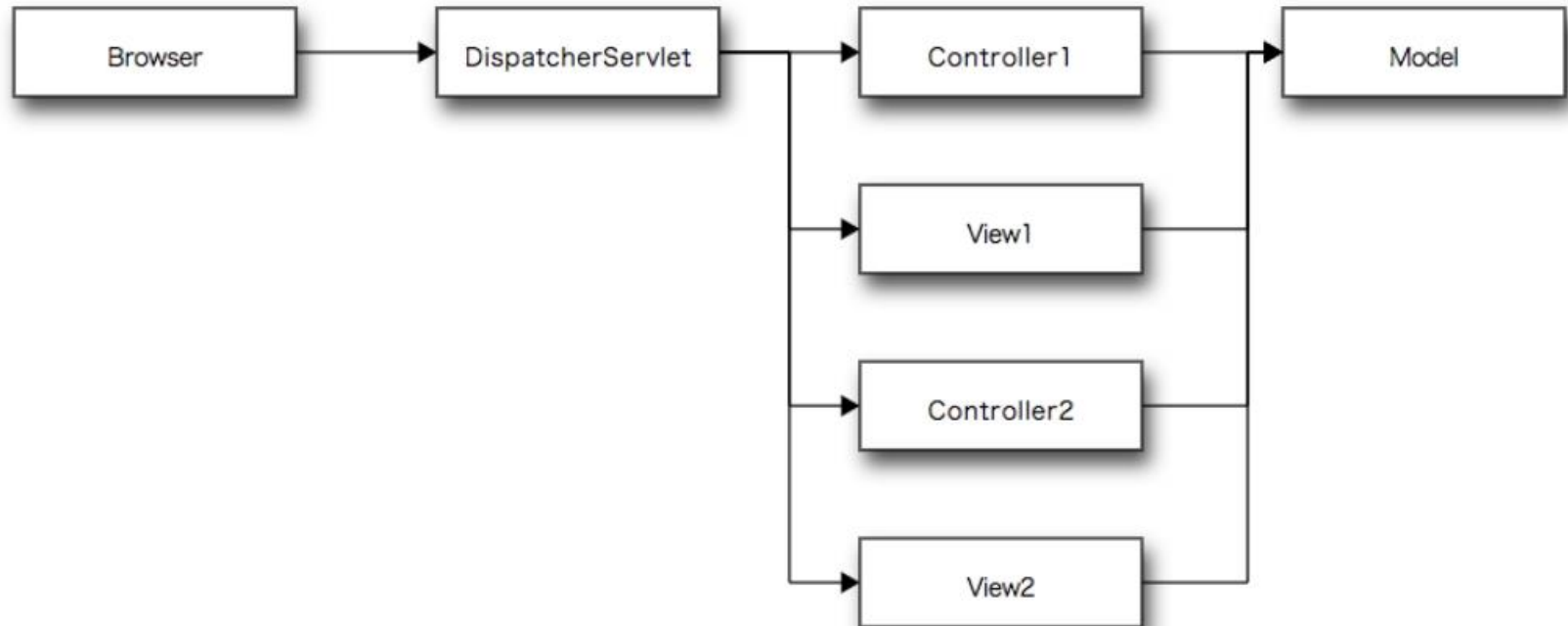


# Questions

- Can you show an example controller method in Spring MVC?
- Can you explain a simple flow in Spring MVC?
- What is a ViewResolver?
- What is Model?
- What is ModelAndView?
- What is a RequestMapping?

# Controller Method

```
@RequestMapping(value = "/list-todos",  
    method = RequestMethod.GET)  
public String listTodos(ModelMap model) {  
    model.addAttribute("todos",  
        service.retrieveTodos(retrieveLoggedInUserName()));  
    return "list-todos";  
}
```



# ViewResolver

```
<bean
  class=
  "org.springframework.web.servlet.view.InternalResourceViewRe
  <property name="prefix">
    <value>/WEB-INF/views/</value>
  </property>
  <property name="suffix"> <value>.jsp</value>
  </property>
</bean>
```

# Model vs ModelAndView

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public String showLoginPage(ModelMap model) {
    model.put("name", "in28Minutes");
    return "welcome";
}
```

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public ModelAndView showLoginPage() {
    ModelAndView mv = new ModelAndView();
    mv.addObject("name", "in28Minutes");
    mv.setViewName("welcome");
}
```

# Questions

- What is Dispatcher Servlet?
- How do you set up Dispatcher Servlet?



```
<servlet> <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherSer
  <init-param>
    <param-name>contextConfigLocation</param-name> <param-
    value>/WEB-INF/todo-servlet.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping> <servlet-name>dispatcher</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

# Questions

- What is a form backing object?
- How is validation done using Spring MVC?
- What is BindingResult?
- How do you map validation results to your view?
- What are Spring Form Tags?

# Show New Todo Page

```
@RequestMapping(value = "/add-todo",  
                  method = RequestMethod.GET)  
public String showTodoPage(ModelMap model) {  
  
    model.addAttribute("todo", new Todo(0,  
        retrieveLoggedInUserName(), "",  
        new Date(), false));  
    return "todo";  
  
}
```

# todo.jsp

```
<form:form method="post" commandName="todo">
  <fieldset>
    <form:label path="desc">Description</form:label>
    <form:input path="desc" type="text"/>
    <form:errors path="desc"/>
  </fieldset>
  <fieldset>
    <form:label path="targetDate">Target Date</form:label>
    <form:input path="targetDate" type="text" />
    <form:errors path="targetDate"/>
  </fieldset>
  <input type="submit" value="Submit" />
</form:form>
```

# Add Todo

```
@RequestMapping(value = "/add-todo", method = RequestMethod.POST)
public String addTodo(ModelMap model, @Valid Todo todo,
                        BindingResult result) {
    if (result.hasErrors()) {
        return "todo";
    }
    service.addTodo(retrieveLoggedInUserName(), todo.getDesc(),
                    false);
    model.clear();
    return "redirect:list-todos";
}
```

# Todo.java

```
public class Todo {  
  
    private int id;  
    private String user;  
  
    @Size(min = 6, message = "Enter at least 6 characters")  
    private String desc;
```

# Questions

- What is a Path Variable?
- What is a Model Attribute?
- What is a Session Attribute?

# Path Variable

- <http://localhost:8080/todos/1>

```
@RequestMapping(value = "/todos/{id}")  
public Todo retrieveTodo(@PathVariable int id) {  
    return service.retrieveTodo(id);  
}
```



```
@ModelAttribute  
public void addAttributes(Model model) {  
    model.addAttribute("options",  
        Arrays.asList(  
            "Option 1", "Option 2", "Option 3" ));  
}
```

# Model Attribute

- Indicates the purpose of that method is to add one or more model attributes.
- Invoked before @RequestMapping methods.
- Used to fill the model with commonly needed attributes
  - Drop down values for form

# @SessionAttributes

- List the names of model attributes which should be transparently stored in the session or some conversational storage.

```
@SessionAttributes("name")  
public class TodoController {
```

# Questions

- What is a init binder?
- How do you set default date format with Spring?

@InitBinder

```
protected void initBinder(WebDataBinder binder)
{ SimpleDateFormat dateFormat = new
SimpleDateFormat("dd/MM/
binder.registerCustomEditor(Date.class, new CustomDateEdit
dateFormat, false));
}
```

# Questions

- How do you implement common logic for Controllers in Spring MVC?
- What is a Controller Advice?
- What is `@ExceptionHandler`?

# Questions

- How to handle exceptions for web applications?
- What are the important things to think about when implementing Exception Handling?
- How do you implement Specific Error Handling for a Spring MVC Controller?

# ExceptionHandlerController

@ControllerAdvice

```
public class ExceptionController {  
  
    private Log logger = LogFactory.getLog(ExceptionController.class)  
  
    @ExceptionHandler(value = Exception.class)  
    public String handleException  
        (HttpServletRequest request, Exception ex) {  
        logger.error("Request " + request.getRequestURL()  
            + " Threw an Exception", ex);  
        return "error";  
    }  
}
```



# Questions

- Why is Spring MVC so popular?

# Spring MVC

- Clear Separation of Concerns
  - DispatcherServlet
  - ViewResolver
  - View
  - Model

**We will discuss more "Spring  
MVC" questions in the section on  
RESTful Webservices.**

# Spring Boot

# Questions

- What is Spring Boot?
- What are the important Goals of Spring Boot?
- What are the important Features of Spring Boot?

# Why Spring Boot?

- Spring based applications have a lot of configuration.
- When we use Spring MVC, we need to configure component scan, dispatcher servlet, a view resolver, web jars(for delivering static content) among other things.

# Why Spring Boot?

- World is moving towards Microservices.
- We do not have a lot of time to set up 100 microservices.

# Spring Boot Goals

- Quick Start to Spring
- Be opinionated
- Non functional features
- No code generation



# Spring Boot Features

- Auto Configuration
- Spring Boot Starter Projects
- Spring Boot Actuator
- Embedded Server

# Questions

- Compare Spring Boot vs Spring?
- Compare Spring Boot vs Spring MVC?

# Spring

Most important feature of Spring Framework is Dependency Injection. At the core of all Spring Modules is Dependency Injection or IOC Inversion of Control.

```
@RestController
public class WelcomeController {

    private WelcomeService service = new WelcomeService();

    @RequestMapping("/welcome")
    public String welcome() {
        return service.retrieveWelcomeMessage();
    }
}
```

# With Spring

```
@Component
```

```
public class WelcomeService {  
    //Bla Bla Bla  
}
```

```
@RestController
```

```
public class WelcomeController {
```

```
    @Autowired
```

```
    private WelcomeService service;
```

```
    @RequestMapping("/welcome")
```

```
    public String welcome() {  
        return service.retrieveWelcomeMessage();  
    }
```

# Problems Spring Solves

- Problem 1 : Duplication/Plumbing Code
- Problem 2 : Good Integration with Other Frameworks.

# Spring MVC

Spring MVC Framework provides decoupled way of developing web applications. With simple concepts like Dispatcher Servlet, ModelAndView and View Resolver, it makes it easy to develop web applications.

# Spring Boot

- Eliminates all configuration needed by Spring and Spring MVC and auto configures it
  - No need for `@ComponentScan`. Default Component Scan.
  - No need to configure `DispatcherServlet`
  - No need to configure a Data Source, Entity Manager Factory or Transaction Manager.



# Spring Boot Thinking

Can we bring more intelligence into this? When a spring mvc jar is added into an application, can we auto configure some beans automatically?

# Spring Boot Thinking

Spring Boot looks at

- Frameworks available on the CLASSPATH
- Existing configuration for the application.

Based on these, Spring Boot provides  
Auto Configuration.

# Questions

- What is the importance of `@SpringBootApplication`?

# SpringBootApplication

```
@SpringBootApplication @EnableAutoConfiguration  
@ComponentScan  
public @interface SpringBootApplication {
```

# Questions

- What is Auto Configuration?
- How can we find more information about Auto Configuration?

Spring Boot looks at a) Frameworks available on the CLASSPATH b) Existing configuration for the application. Based on these, Spring Boot provides basic configuration needed to configure the application with these frameworks. This is called Auto Configuration.

# Application Startup Log

Mapping servlet: 'dispatcherServlet' to [/]

Mapped "{[/error]}" onto public org.springframework.http.Respo  
<java.util.Map<java.lang.String, java.lang.Object>> org.spring  
BasicErrorController.error(javax.servlet.http.HttpServletRequestReque

Mapped URL path [/webjars/\* \*] onto handler of type  
[class org.springframework.web.servlet.resource.ResourceHttpRe

# Implementation

- spring-boot-autoconfigure.jar
- To get more details
  - Turn on Debug logging  
`logging.level.org.springframework:DEBUG`
  - Use Spring Boot Actuator



# Questions

- What is an embedded server? Why is it important?
- What is the default embedded server with Spring Boot?
- What are the other embedded servers supported by Spring Boot?

# Embedded Server

- Server is embedded as part of the deployable - jar.
- Removes the need to have the server pre-installed on the deployment environment.
- Default is Tomcat.
- Spring Boot also supports Jetty and Undertow.

# Switching to Jetty

```
<dependency> <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-web</artifactId>  
  <exclusions>  
    <exclusion>  
      <groupId>org.springframework.boot</groupId>  
      <artifactId>spring-boot-starter-tomcat</artifactId>  
    </exclusion>  
  </exclusions>  
</dependency>  
<dependency> <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-jetty</artifactId>  
</dependency>
```

# Questions

- What are Starter Projects?
- Can you give examples of important starter projects?

# Spring Boot Documentation

Starters are a set of convenient dependency descriptors that you can include in your application. You get a one-stop-shop for all the Spring and related technology that you need, without having to hunt through sample code and copy paste loads of dependency descriptors.

# Spring Boot Documentation

For example, if you want to get started using Spring and JPA for database access, just include the spring-boot-starter-data-jpa dependency in your project, and you are good to go.

# Starters

- spring-boot-starter-web-services - SOAP WebServices
- spring-boot-starter-web - Web & RESTful applications
- spring-boot-starter-test - Unit, Integration Testing
- spring-boot-starter-jdbc - Traditional JDBC spring-
- boot-starter-hateoas - HATEOAS features

# Starters

- spring-boot-starter-security - Authentication and Authorization using Spring Security
- spring-boot-starter-data-jpa - Spring Data JPA with Hibernate
- spring-boot-starter-cache - Enabling Spring Framework's caching support
- spring-boot-starter-data-rest - Expose Simple REST Services using Spring Data REST



- spring-boot-starter-actuator - To use advanced features like monitoring & tracing to your application out of the box
- spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-starter-tomcat - To pick your specific choice of Embedded Servlet Container
- spring-boot-starter-logging - For Logging using logback
- spring-boot-starter-log4j2 - Logging using Log4j2
-

# Questions

- What is Starter Parent?
- What are the different things that are defined in Starter Parent?
- How does Spring Boot enforce common dependency management for all its Starter projects?

# Starter Parent

```
<parent>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-parent</artifactId>  
  <version>2.0.0.RELEASE</version>  
</parent>
```

# Inside Starter Parent

```
<parent> <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-dependencies</artifactId>  
  <version>2.0.0.RELEASE</version>  
  <relativePath>../../spring-boot-dependencies</relativePath>  
</parent>
```

```
<java.version>1.6</java.version>  
<resource.delimiter>@ </resource.delimiter> <!-- delimiter that  
<project.build.sourceEncoding>UTF-8</project.build.sourceEncod  
<project.reporting.outputEncoding>UTF-8</project.reporting.out  
<maven.compiler.source>${java.version}</maven.compiler.source>  
<maven.compiler.target>${java.version}</maven.compiler.target>
```

# Inside Spring Boot Dependencies

```
<ehcache3.version>3.1.1</ehcache3.version>
```

```
...
```

```
<h2.version>1.4.192</h2.version>
```

```
<hamcrest.version>1.3</hamcrest.version>
```

```
<hazelcast.version>3.6.4</hazelcast.version>
```

```
<hibernate.version>5.0.9.Final</hibernate.version> <hibernate-  
validator.version>5.2.4.Final</hibernate-validator.
```

```
<jackson.version>2.8.1</jackson.version>
```

```
.....
```

```
<jersey.version>2.23.1</jersey.version> <spring-  
security.version>4.1.1.RELEASE</spring-security.versio
```

```
<tomcat.version>8.5.4</tomcat.version> <xml-  
apis.version>1.4.01</xml-apis.version>
```

# Questions

- What is Spring Initializr?

Spring Initializr <http://start.spring.io/> is  
great tool to bootstrap your Spring  
Boot projects.

# Questions

- What is application.properties?
- What are some of the important things that can be customized in application.properties?



application.properties is used to  
configure your Spring Boot Application

# Example Configuration

```
logging.file=  
logging.level.*= spring.autoconfigure.exclude=  
spring.profiles.active= server.error.path=/error  
server.port=8080  
spring.http.converters.preferred-json-mapper=jackson
```

# Questions

- How do you externalize configuration using Spring Boot?
- How can you add custom application properties using Spring Boot?
- What is `@ConfigurationProperties`?

```
logging:
  level:
    org.springframework: DEBUG
app:
  name: In28Minutes
  description: ${app.name} is your first Spring Boot applicat
```

```
import
    org.springframework.boot.context.properties.ConfigurationProp

@Component @ConfigurationProperties("basic")
public class BasicConfiguration {
    private boolean value;
    private String message;
    private int number;
```

@Autowired

private BasicConfiguration configuration;

@RequestMapping("/dynamic-configuration")

public Map dynamicConfiguration() {

// Not the best practice to use a map to store different ty

Map map = new HashMap();

map.put("message", configuration.getMessage());

map.put("number", configuration.getNumber());

map.put("key", configuration.isValue());

return map;

}

## application.properties

```
basic.value= true  
basic.message= Dynamic Message basic.number=  
100
```

## application.yaml

```
basic:  
  value: true  
  message: Dynamic Message YAML  
  number: 100
```

# Advantage

- Type Safety

\*\*\*\*\*

APPLICATION FAILED TO START \*\*\*\*\*

Description:

Binding to target

com.in28minutes.springboot.configuration.BasicConfiguration @ 39  
failed:

Property: basic.number

Value: ABC

Reason: Failed to convert property value of type [java.lang.String] to required type [int] for property 'number'; nested exception org.springframework.core.convert.converter.ConverterNotFoundException:



# Good Practice

- Design all your application configuration using `ConfigurationProperties`

# Questions

- What is a profile?
- How do you define beans for a specific profile?
- How do you create application configuration for a specific profile?

# Profile

How do you have different  
configuration for different  
environments?

# Profile

- application-dev.properties
- application-qa.properties
- application-stage.properties
- application-prod.properties
- application.properties

# Profile

- Based on the active profile, appropriate configuration is picked up.
- Used to Configure Resources - Databases, Queues, External Services

# Setting a profile

- Using `-Dspring.profiles.active=prod` in VM Arguments
- In `application.properties`,  
`spring.profiles.active=prod`

# Profiles in code

@Profile("dev") on a bean

```
@Profile("dev")
@Bean
public String devBean() {
    return "I will be available in profile dev";
}

@Profile("prod")
@Bean
public String prodBean() {
    return "I will be available in profile prod";
}
```

# Questions

- What is Spring Boot Actuator?
- How do you monitor web services using Spring Boot Actuator?
- How do you find more information about your application environment using Spring Boot?



# Spring Boot Actuator

- Monitoring
  - /env, /metrics, /trace, /dump
  - /beans, / autoconfig, /configprops, /mappings
- URL has changed in Spring Boot 2.0 - /application

```
<dependency> <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>
```

# Questions

- What is a CommandLineRunner?

# CommandLineRunner

Spring Documentation - interface used  
to indicate that a bean should run  
when it is contained within a  
SpringApplication

```
public interface CommandLineRunner {  
    void run(String... args) throws Exception;  
}
```

# **Database Connectivity - JDBC, Spring JDBC & JPA**

# Questions

- What is Spring JDBC? How is different from JDBC?
- What is a JdbcTemplate?
- What is a RowMapper?

# JDBC - Update Todo

```
Connection connection = datasource.getConnection();

PreparedStatement st = connection.prepareStatement(
    "Update todo set user=?, desc=?, target_date=?, is_done=? wh

st.setString(1, todo.getUser()); st.setString(2,
todo.getDesc());
st.setTimestamp(3, new Timestamp(todo.getTargetDate().getTime(
st.setBoolean(4, todo.isDone()); st.setInt(5, todo.getId());

st.execute();
st.close();
connection.close();
```

# Spring JDBC

```
jdbcTemplate
.update
("Update todo set user=?, desc=?, target_date=?, is_done=? wh
    todo.getUser(), todo.getDesc(),
    new Timestamp(todo.getTargetDate().getTime()),
    todo.isDone(), todo.getId());

jdbcTemplate.update("delete from todo where id=?",
    id);
```

# Spring JDBC - RowMapper

```
new BeanPropertyRowMapper(Todo.class)
```

```
class TodoMapper implements RowMapper<Todo> {  
    @Override  
    public Todo mapRow(ResultSet rs, int rowNum)  
        throws SQLException {  
        Todo todo = new Todo();  
  
        todo.setId(rs.getInt("id"));  
        todo.setUser(rs.getString("user"));  
        todo.setDesc(rs.getString("desc"));  
        todo.setTargetDate(  
            rs.getTimestamp("target_date"));  
        todo.setDone(rs.getBoolean("is_done"));  
        return todo;  
    }  
}
```



# Spring JDBC - RowMapper

```
return jdbcTemplate.query(  
    "SELECT * FROM TODO where user = ?",  
    new Object[] { user }, new TodoMapper());  
  
return jdbcTemplate.queryForObject(  
    "SELECT * FROM TODO where id=?",  
    new Object[] { id }, new TodoMapper())
```

# Questions

- What is JPA?
- What is Hibernate?
- How do you define an entity in JPA?
- What is an Entity Manager?
- What is a Persistence Context?

# JPA - Update Todo

## Defining an entity

```
@Entity
@Table(name = "Todo")
public class Todo {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String user;

    private String desc;

    private Date targetDate;

    private boolean isDone;
```

# JPA - Update Todo

```
public class TodoJPAService implements TodoDataService {  
  
    @PersistenceContext  
    private EntityManager entityManager;  
  
    @Override  
    public void updateTodo(Todo todo) {  
        entityManager.merge(todo);  
    }  
}
```

# Questions

- How do you map relationships in JPA?
- What are the different types of relationships in JPA?
- How do you define One to One Mapping in JPA?
- How do you define One to Many Mapping in JPA?
- How do you define Many to Many Mapping in JPA?

# One to One Relationship

```
@Entity
public class Passport {

    ....
    // Inverse Relationship
    // bi-directional OneToOne relationship
    // Column will not be created in the table
    // Try removing mappedBy = "passport" => You will see that s
    // will be created in passport
    @OneToOne(fetch = FetchType.LAZY, mappedBy = "passport")
    private Student student;
```

```
@Entity
@Table(name = "Student")
public class Student {

    @OneToOne
    private Passport passport;
```

# One to Many Relationship

```
@Entity  
public class Project {  
    @OneToMany(mappedBy = "project")  
    private List<Task> tasks;
```

```
@Entity  
public class Task {  
    @ManyToOne @JoinColumn(name="PROJECT_ID")  
    private Project project;
```

# Many to Many Relationship

@Entity

```
public class Project {
```

```
    @ManyToMany
```

```
    // @JoinTable(name="STUDENT_PROJ",
```

```
    // joinColumns=@JoinColumn(name="STUDENT_ID"),
```

```
    // inverseJoinColumns=@JoinColumn(name="PROJECT_ID"))
```

```
    private List<Student> students;
```

```
public class Student {
```

```
    @ManyToMany(mappedBy = "students")
```

```
    private List<Project> projects;
```



# Questions

- How do you define a datasource in a Spring Context?
- What is the use of persistence.xml
- How do you configure Entity Manager Factory and Transaction Manager?
- How do you define transaction management for Spring – Hibernate integration?

# Defining a Data Source

```
#HSQL in-memory db db.driver=org.hsqldb.jdbcDriver
db.url=jdbc:hsqldb:mem:firstdb
db.username=sa
db.password=
```

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDa
    destroy-method="close">
    <property name="driverClass" value="\${db.driver}" />
    <property name="jdbcUrl" value="\${db.url}" />
    <property name="user" value="\${db.username}" />
    <property name="password" value="\${db.password}" />
</bean>
```

# Configuring Hibernate

src/main/resources/config/hibernate.properties

```
hibernate.dialect=org.hibernate.dialect.HSQLDialect  
hibernate.show_sql=false  
hibernate.format_sql=false hibernate.use_sql_comments=true
```

# persistence.xml

src/main/resources/META-INF/persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  version="2.0">
  <persistence-unit name="hsqldb" transaction-type="RESOURCE_
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</
    <properties>
      <property name="hibernate.dialect" value="org.hibernate.
      <property name="hibernate.connection.url" value="jdbc:hs
      <property name="hibernate.connection.driver_class" value
      <property name="hibernate.connection.username" value="sa
      <property name="hibernate.connection.password" value=""
    </properties>
  </persistence-unit>
</persistence>
```

# Configure Entity Manager Factory and Transaction Manager

```
<bean
  class="org.springframework.orm.jpa.LocalContainerEntityMan
  id="entityManagerFactory">
  <property name="persistenceUnitName" value="hsql_pu" />
  <property name="dataSource" ref="dataSource" />
</bean>

<bean id="transactionManager" class="org.springframework.orm
  <property name="entityManagerFactory" ref="entityManagerFa
  <property name="dataSource" ref="dataSource" /> </bean>

<tx:annotation-driven transaction-manager="transactionManage
```

# Making Service Transactional

```
@Service
public class StudentService {

    @Autowired
    StudentRepository service;

    @Transactional
    public Student insertStudent(Student student) {
        return service.insertStudent(student);
    }
}
```

```
@Service
@Transactional
public class StudentService {

    @Autowired
    StudentRepository service;
```

# Spring Data

# Questions

- What is Spring Data?
- What is the need for Spring Data?
- What is Spring Data JPA?



# Duplication in JPA Repositories

## Passport Repository

```
@PersistenceContext
```

```
private EntityManager entityManager;
```

```
public Passport getPassport(final long id) {  
    Passport passport = entityManager  
        .find(Passport.class, id);  
    return passport;  
}
```

```
public Passport createPassport(Passport passport) {  
    return entityManager.merge(passport);  
}
```

# Duplication in JPA Repositories

## Student Repository

```
@PersistenceContext
```

```
private EntityManager entityManager;
```

```
public Student retrieveStudent(final long id) {  
    return entityManager.find(Student.class, id);  
}
```

```
public Student createStudent(Student student) {  
    return entityManager.merge(student);  
}
```

# **Explosion of Data Stores**

- Variety of Big Data Stores

# Spring Data

Common Abstractions to store and retrieve data from data stores

- Independent of type of data store

# Spring Data JPA

Extends Spring Data for connecting to  
JPA

# Using Spring Data JPA

```
public interface StudentRepository extends  
CrudRepository<Stud }
```

```
public interface PassportRepository extends  
CrudRepository<Pas }
```

# Questions

- What is a CrudRepository?
- What is a PagingAndSortingRepository?

# CrudRepository

```
public interface CrudRepository<T, ID> extends Repository<T, ID> {
    <S extends T> S save(S entity);
    Optional<T> findById(ID id);
    boolean existsById(ID id);
    Iterable<T> findAll();
    void deleteById(ID id);
    long count();
    //Other Methods
}
```



# PagingAndSortingRepository

## Pagination and Sorting

```
public interface PagingAndSortingRepository<T, ID>  
    extends CrudRepository<T, ID> {  
    Iterable<T> findAll(Sort sort);  
    Page<T> findAll(Pageable pageable);  
}
```

# Using PagingAndSortingRepository

```
Sort sort = new Sort(Sort.Direction.DESC, "field_name");  
passportRepository.findAll(sort);
```

```
//Page Size - 10
```

```
PageRequest pageable = new PageRequest(0,10);  
Page<Passport> page = passportRepository.findAll(pageable);  
System.out.println(userPage.getTotalPages());  
System.out.println(userPage.nextPageable());
```

# Unit Testing

# Questions

- How does Spring Framework Make Unit Testing Easy?  
What is Mockito?
- What is your favorite mocking framework?
- How do you do mock data with Mockito?
- What are the different mocking annotations that you
- worked with?

```
public class SomeBusinessImpl {  
    private DataService dataService;  
    //Constructor - public SomeBusinessImpl(DataService dataServ  
    int findTheGreatestFromAllData() {  
        int[] data = dataService.retrieveAllData();  
        int greatest = Integer.MIN_VALUE;  
  
        for (int value : data) {  
            if (value > greatest) {  
                greatest = value;  
            }  
        }  
        return greatest;  
    }  
}
```

# Basic Mocking

@Test

```
public void testFindTheGreatestFromAllData() {  
  
    DataService dataServiceMock = mock(DataService.class);  
    when(dataServiceMock.retrieveAllData())  
        .thenReturn(new int[] { 24, 15, 3 });  
  
    SomeBusinessImpl businessImpl =  
        new SomeBusinessImpl(dataServiceMock);  
  
    int result = businessImpl.findTheGreatestFromAllData();  
    assertEquals(24, result);  
}
```

# Using Annotations

```
@RunWith(MockitoJUnitRunner.class)
public class SomeBusinessMockAnnotationsTest {

    @Mock
    DataService dataServiceMock;

    @InjectMocks
    SomeBusinessImpl businessImpl;

    @Test
    public void testFindTheGreatestFromAllData() {
        when(dataServiceMock.retrieveAllData())
            .thenReturn(new int[] { 24, 15, 3 });
        assertEquals(24, businessImpl.findTheGreatestFromAllData())
    }
}
```

# Questions

- What is MockMvc?
- What is @WebMvcTest?
- What is @MockBean?
- How do you write a unit test with MockMVC?
- What is JSONAssert?



# Mock MVC Test with Spring Boot

```
public Question retrieveDetailsForQuestion(  
    @PathVariable String surveyId,  
    @PathVariable String questionId) {  
    return  
        surveyService.retrieveQuestion(surveyId, questionId);  
}
```

# Mock MVC Test with Spring Boot

```
@RunWith(SpringRunner.class)
@WebMvcTest(value = SurveyController.class, secure = false)
public class SurveyControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private SurveyService surveyService;
```

# Mock MVC Test - Continued

@Test

```
public void retrieveDetailsForQuestion() throws Exception {  
    Question mockQuestion = new Question("Question1",  
        "Largest Country in the World", "Russia", Arrays.asList(  
            "India", "Russia", "United States", "China"));  
  
    Mockito.when(  
        surveyService.retrieveQuestion(Mockito.anyString(), Mockito.  
            .anyString())).thenReturn(mockQuestion);  
  
    RequestBuilder requestBuilder =  
        MockMvcRequestBuilders.get( "/surveys/Survey1/questions  
            /Question1").accept( MediaType.APPLICATION_JSON);
```

# Mock MVC Test - Continued

```
MvcResult result = mockMvc.perform(requestBuilder)
                             .andReturn();
```

```
String expected =
    "{id:Question1,description:Largest Country in the World,co
```

```
JSONAssert.assertEquals(expected, result.getResponse()
    .getContentAsString(), false);
```

```
// Assert
}
```

# Questions

- How do you write an integration test with Spring Boot?
- What is @SpringBootTest?
- What is @LocalServerPort?
- What is TestRestTemplate?

# Integration Test with Spring Boot

```
@RunWith(SpringRunner.class)
@SpringBootTest(classes = Application.class,
    webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT
public class SurveyControllerIT {

    @LocalServerPort
    private int port;

}
```

# Integration Test with Spring Boot

@Test

```
public void testRetrieveSurveyQuestion() {  
  
    String url = "http://localhost:" + port  
        + "/surveys/Survey1/questions/Question1";  
  
    HttpHeaders headers = new HttpHeaders();  
    headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON))  
    HttpEntity<String> entity =  
        new HttpEntity<String>(null, headers);  
  
    TestRestTemplate restTemplate = new TestRestTemplate();  
    ResponseEntity<String> response = restTemplate.exchange(url,  
        HttpMethod.GET, entity, String.class);  
}
```

# Integration Test with Spring Boot

```
String expected =
```

```
"{id:Question1,description:Largest Country in the World,corr
```

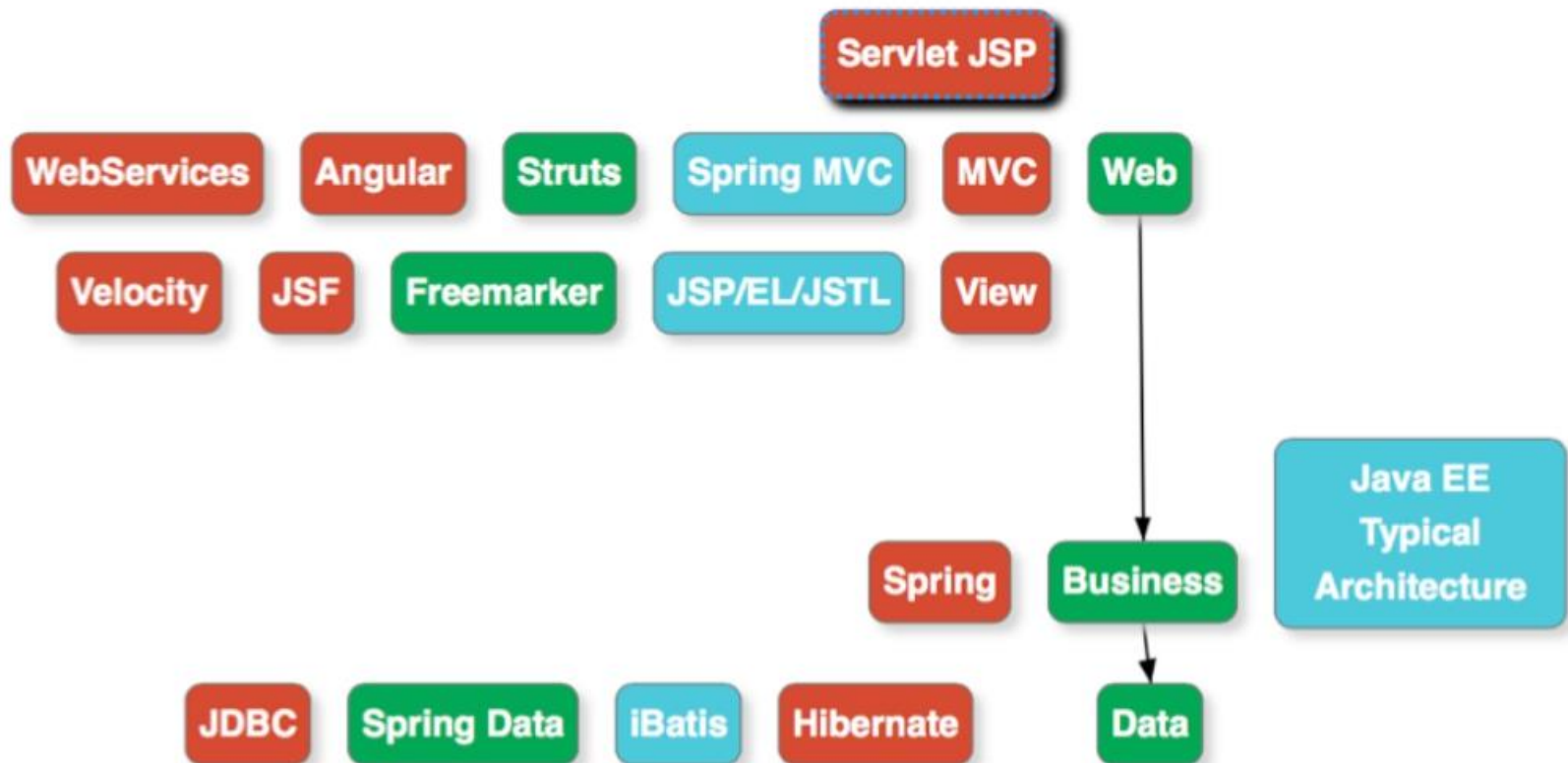
```
JSONAssert.assertEquals(expected, response.getBody(), false)  
}
```



**AOP**

# Questions

- What are cross cutting concerns?
- How do you implement cross cutting concerns in a web application?
- If you would want to log every request to a web application, what are the options you can think of?
- If you would want to track performance of every request, what options can you think of?



# Questions

- What is an Aspect and Pointcut in AOP? What
- are the different types of AOP advices? What
- is weaving?

@Component

```
class HiByeService {  
    public void sayHi(String name) {  
        System.out.println("Hi " + name);  
    }  
  
    public void sayBye() {  
        System.out.println("Bye");  
    }  
  
    public String returnSomething() {  
        return "Hi Bye";  
    }  
}
```

```
@Aspect
@Component
class MyAspect {
    @Before("execution(* HiByeService.*(..))")
    public void before(JoinPoint joinPoint)
    { System.out.print("Before ");
      System.out.print(joinPoint.getSignature().getName());
      System.out.println(Arrays.toString(joinPoint.getArgs()));
    }
}
```

```
@AfterReturning(pointcut = "execution(*  
    HiByeService.*(..))", returning =  
    "result")  
public void after(JoinPoint joinPoint, Object result)  
{ System.out.print("After ");  
  System.out.print(joinPoint.getSignature().getName());  
  System.out.println(" result is " + result);  
}
```

```
@Around(value = "execution(* HiByeService.*(..))")
public void around(ProceedingJoinPoint joinPoint)
                                                    throws Throwable {
    long startTime = new Date().getTime();
    Object result = joinPoint.proceed();
    long endTime = new Date().getTime();
    System.out.print("Execution Time - "
                    + (endTime - startTime));
}
```



# AOP concepts

- Joinpoint
- Advice
- Pointcut
- Aspect
- Weaving
- Weaver

# Advice Types

- Before advice
- After returning advice
- After throwing advice
- After (finally) advice - Always executed Around
- advice - Most powerful - Performance Logging

# AspectJ vs Spring AOP

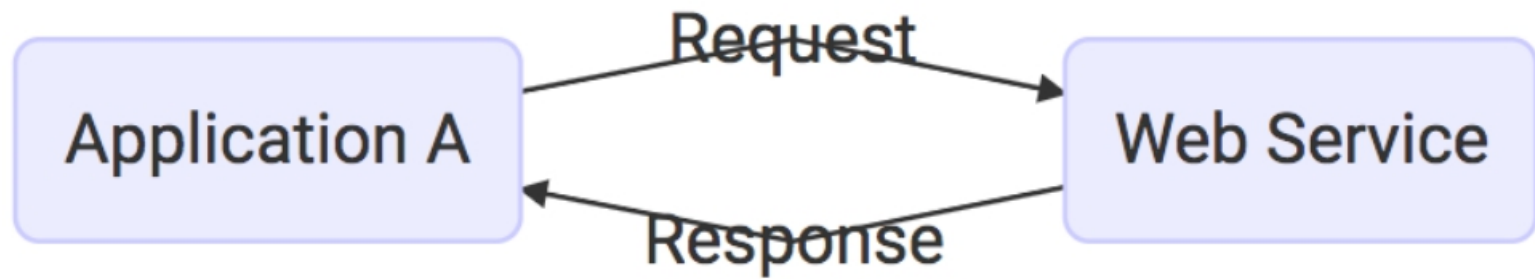
AspectJ is a full fledged AOP framework

- Advise objects not managed by the Spring container
- Advise join points other than simple method executions
  - (for example, field get or set join points)

# SOAP Web Services

# Questions

- What is a Web Service?



# 3 Keys

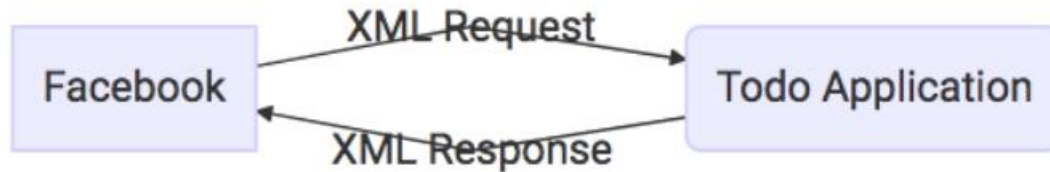
- Designed for machine-to-machine (or application-to-application) interaction
- Should be interoperable - Not platform dependent
- Should allow communication over a network

# Questions

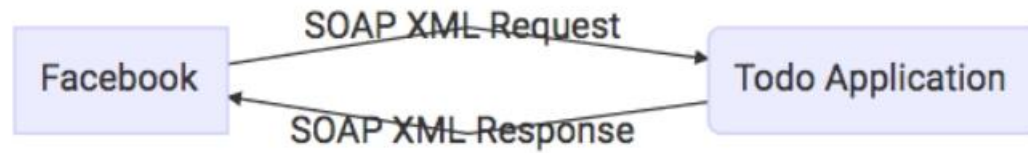
- What is SOAP Web Service?
- What is SOAP?
- What is a SOAP Envelope?
- What is SOAP Header and SOAP Body?



# This is Not SOAP Web Service



# This is SOAP Web Service



# SOAP Envelope

SOAP-ENV: Envelope

A diagram showing a SOAP-ENV: Envelope structure. It consists of a light blue rounded rectangle containing two yellow rounded rectangles. The top yellow rectangle is labeled 'SOAP-ENV: Header' and the bottom yellow rectangle is labeled 'SOAP-ENV: Body'.

SOAP-ENV: Header

SOAP-ENV: Body

# SOAP

- Format
  - SOAP XML Request
  - SOAP XML Response
- Transport
  - SOAP over MQ SOAP
  - over HTTP
- Service Definition
  - WSDL

# Questions

- Can you give an example of SOAP Request and SOAP Response?
- What is a SOAP Header? What kind of information is sent in a SOAP Header?
- Can you give an example of a SOAP Header with Authentication information?

# SOAP Request

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:GetCourseDetailsRequest xmlns:ns2="http://in28min
      <ns2:course>
        <ns2:id>1</ns2:id>
      </ns2:GetCourseDetailsRequest>
    </ns2:getCourseDetailsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:getCourseDetailsResponse xmlns:ns2="http://in28mi
      <ns2:course>
        <ns2:id>1</ns2:id>
        <ns2:name>Spring</ns2:name>
        <ns2:description>10 Steps</ns2:description>
      </ns2:course>
    </ns2:getCourseDetailsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



# SOAP Header

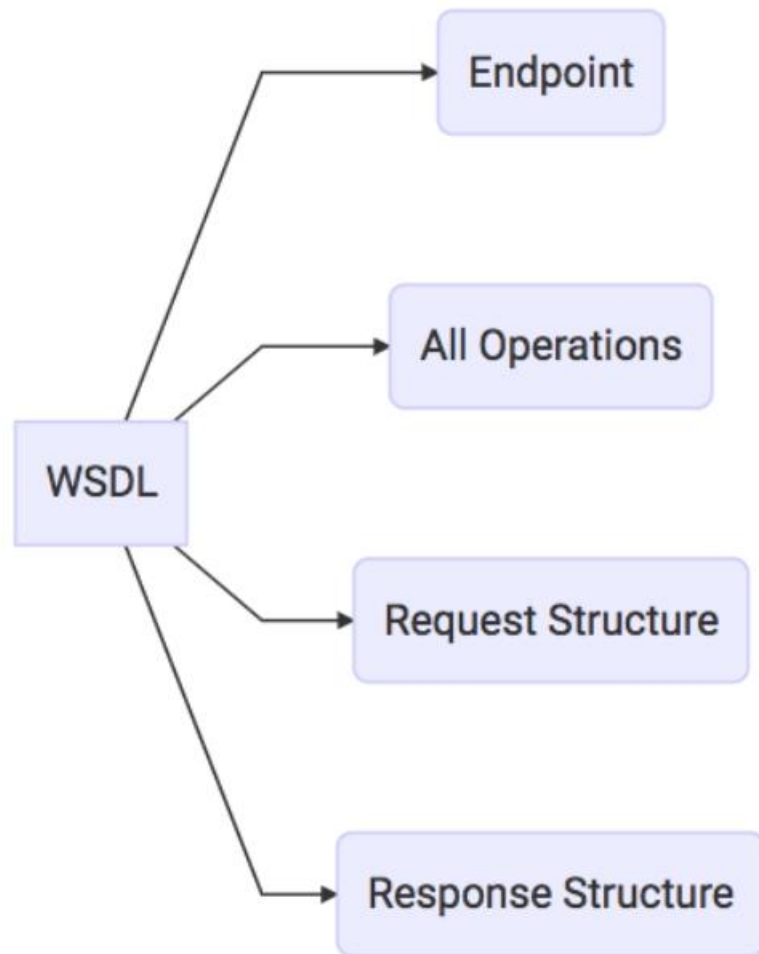
```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-
      open.org/wss/2004/01/oasis mustUnderstand="1">
      <wsse:UsernameToken> <wsse:Username>user</wsse:Username>
        <wsse:Password>password</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </Header>
  <Body>
    <GetCourseDetailsRequest xmlns="http://in28minutes.com/cou
      <id>1</id>
    </GetCourseDetailsRequest>
```

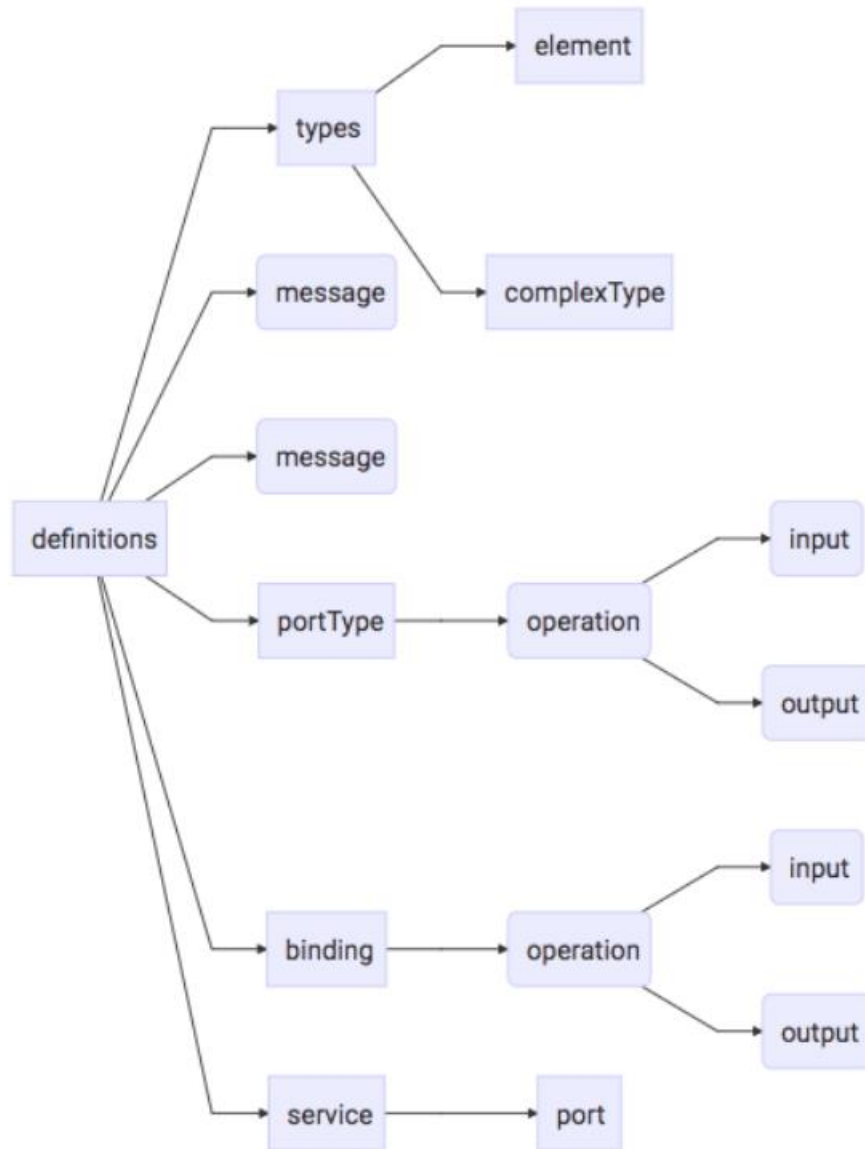
# Questions

- What is WSDL (Web Service Definition Language)?
- What are the different parts of a WSDL?

# WSDL

- Web Service Definition Language





# Questions

- What is Contract First Approach?
- What is an XSD?
- Can you give an example of an XSD?

# Contract

Service Definition specifying

- Format of Request
- Format of Response
- Request/Response Structure
- Transport used
- Endpoint details

# Contract First

We define a contract first!

- With Spring Web Services, we define an XSD first



# XSD

- XML Specification Document!
- How does a valid XML Look like?

# Request XML

```
<GetCourseDetailsRequest xmlns="http://in28minutes.com/courses"
    <id>1</id>
</GetCourseDetailsRequest>
```

# XSD

```
<xs:element name="GetCourseDetailsRequest">  
  <xs:complexType> <xs:sequence>  
    <xs:element name="id" type="xs:int" />  
  </xs:sequence>  
</xs:complexType> </xs:element>
```

# Response XML

```
<ns2:GetCourseDetailsResponse xmlns:ns2="http://in28minutes.co
  <ns2:CourseDetails> <ns2:id>1</ns2:id>
    <ns2:name>Spring</ns2:name>
    <ns2:description>10 Steps</ns2:description>
  </ns2:CourseDetails>
</ns2:GetCourseDetailsResponse>
```

# XSD

```
<xs:element name="GetCourseDetailsResponse"> <xs:complexType>  
  <xs:sequence>  
    <xs:element name="CourseDetails" type="tns:CourseDetails"  
  </xs:sequence>  
</xs:complexType>  
</xs:element>
```

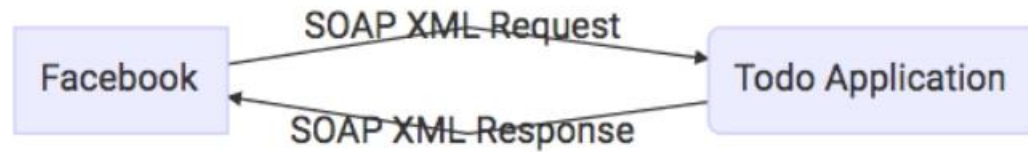
# XSD

```
<xs:complexType name="CourseDetails">  
  <xs:sequence>  
    <xs:element name="id" type="xs:int" /> <xs:element  
      name="name" type="xs:string" />  
    <xs:element name="description" type="xs:string" />  
  </xs:sequence>  
</xs:complexType>
```

# Questions

- What is JAXB?
- How do you configure a JAXB Plugin?

# Convert from Java to SOAP XML





# JAXB Plugin

```
<plugin> <groupId>org.codehaus.mojo</groupId>  
  <artifactId>jaxb2-maven-plugin</artifactId>  
  <version>1.6</version>  
  <executions>  
    <execution>  
      <id>xjc</id>  
      <goals> <goal>xjc</goal>  
      </goals>  
    </execution>  
  </executions>  
  <configuration>  
    <schemaDirectory>${project.basedir}/src/main/resources</sc  
    <outputDirectory>${project.basedir}/src/main/java</outputD
```

# Questions

- What is an Endpoint?
- Can you show an example endpoint written with Spring Web Services?

# Endpoint

```
@PayloadRoot(namespace = "http://in28minutes.com/courses",
               localPart = "GetCourseDetailsRequest")
@ResponsePayload
public GetCourseDetailsResponse processCourseDetailsRequest
    (@RequestPayload GetCourseDetailsRequest request) {

    Course course = service.findById(request.getId());

    if (course == null)
        throw new CourseNotFoundException("Invalid Course Id " + r

    return mapCourseDetails(course);
}
```

# Questions

- What is a `MessageDispatcherServlet`?
- How do you configure a `MessageDispatcherServlet`?

# MessageDispatcherServlet

@Bean

```
public ServletRegistrationBean messageDispatcherServlet
    (ApplicationContext context) {
    MessageDispatcherServlet messageDispatcherServlet
        = new MessageDispatcherServlet();
    messageDispatcherServlet.setApplicationContext(context);
    messageDispatcherServlet.setTransformWsdlLocations(true);
    return new ServletRegistrationBean(messageDispatcherServlet,
        "/ws/*");
}
```

# Questions

- How do you generate a WSDL using Spring Web Services?

```
@Bean(name = "courses")
public DefaultWsd111Definition defaultWsd111Definition(XsdSchema
    DefaultWsd111Definition definition = new DefaultWsd111Defini
    definition.setPortTypeName("CoursePort");
    definition.setTargetNamespace("http://in28minutes.com/course
    definition.setLocationUri("/ws");
    definition.setSchema(coursesSchema);
    return definition;
}
```

```
@Bean
public XsdSchema coursesSchema() {
    return new SimpleXsdSchema(
        new ClassPathResource("course-details.xsd"));
}
```

# Questions

- How do you implement error handling for SOAP Web Services?
- What is a SOAP Fault?



# Endpoint

```
@PayloadRoot(namespace = "http://in28minutes.com/courses",
               localPart = "GetCourseDetailsRequest")
@ResponsePayload
public GetCourseDetailsResponse processCourseDetailsRequest
    (@RequestPayload GetCourseDetailsRequest request) {

    Course course = service.findById(request.getId());

    if (course == null)
        throw new CourseNotFoundException("Invalid Course Id " + r

    return mapCourseDetails(course);
}
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault> <faultcode>SOAP-
      ENV:Server</faultcode>
      <faultstring xml:lang="en">Invalid Course Id 1000<
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
@SoapFault(  
    faultCode = FaultCode.CUSTOM,  
    customFaultCode =  
        "{http://in28minutes.com/courses}001_COURSE_NOT_FOUND"  
)  
public class CourseNotFoundException extends  
    RuntimeException }
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
  <SOAP-ENV:Header />
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode xmlns:ns0="http://in28minutes.com/courses">
        ns0:001_COURSE_NOT_FOUND</faultcode>
      <faultstring xml:lang="en">
        Invalid Course Id 1234</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# RESTful Web Services

# Questions

- What is REST?

# REpresentational State Transfer

REST is a style of software architecture  
for distributed hypermedia systems



# Make best use of HTTP

REST(REpresentational State Transfer)	
HTTP	
HTTP Methods (GET, PUT, POST..)	HTTP Status Codes (200, 404..)

# Key abstraction - Resource

- A resource has an URI (Uniform Resource Identifier)
  - /users/Ranga/todos/1
  - /users/Ranga/todos
  - /users/Ranga

# Example

- Create a User - POST /users
- Delete a User - DELETE /users/1
- Get all Users - GET /users
- Get one Users - GET /users/1

# REST

- Data Exchange Format
  - No Restriction. JSON is popular
- Transport
  - Only HTTP
- Service Definition
  - No Standard. WADL/Swagger/...

# Questions

- What are the Best Practices of RESTful Services?

# **Best Practices in RESTful Design**

**Consumer First**

**Make best use of  
HTTP**



# Request Methods

- GET
- POST
- PUT
- DELETE

# Response Status

- 200 - SUCCESS
- 404 - RESOURCE NOT FOUND
- 400 - BAD REQUEST
- 201 - CREATED
- 401 - UNAUTHORIZED
- 500 - SERVER ERROR

**No Secure Info in URI**

# Use Plurals

- Prefer `/users` to `/user`
- Prefer `/users/1` to `/user/1`

# Questions

- Can you show the code for an example Get Resource method with Spring REST?
- What happens when we return a bean from a Request Mapping Method?
- What is GetMapping and what are the related methods available in Spring MVC?

```
@GetMapping("/users")  
public List<User> retrieveAllUsers() {  
    return service.findAll();  
}
```

# Questions

- Can you show the code for an example Post Resource method with Spring REST?
- Why do we use ResponseEntity in a RESTful Service?

```
@PostMapping("/users")
public ResponseEntity<Object> createUser(
    @Valid @RequestBody User user) {
    User savedUser = service.save(user);

    URI location = ServletUriComponentsBuilder
        .fromCurrentRequest()
        .path("/{id}").buildAndExpand(savedUser.getId()).toUri();

    return ResponseEntity.created(location).build();
}
```



# Questions

- What is HATEOAS?
- Can you give an Example Response for HATEOAS?
- How do we implement it using Spring?

# HATEOAS

- Hypermedia as The Engine of Application State
- Example
  - When requested for details of a facebook post, we return
    - Link for actions to like, unlike or comment on the post

```
{  
  "id": 1,  
  "name": "Adam",  
  "birthDate": "2017-07-19T09:26:18.337+0000",  
  "_links": {  
    "all-users": {  
      "href": "http://localhost:8080/users"  
    }  
  }  
}
```

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-hateoas</artifactId>  
</dependency>
```

```
@GetMapping("/users/{id}")
public Resource<User> retrieveUser
    (@PathVariable int id) {
    User user = service.findOne(id);

    Resource<User> resource = new Resource<User>(user);

    ControllerLinkBuilder linkTo =
        linkTo(methodOn(this.getClass()).retrieveAllUsers());

    resource.add(linkTo.withRel("all-users"));

    return resource;
}
```

# Questions

- How do you document RESTful web services?
- Can you give a brief idea about Swagger Documentation?
- How do you automate generation of Swagger Documentation from RESTful Web Services? How
- do you add custom information to Swagger Documentation generated from RESTful Web Services?
- What is Swagger-UI?

# OpenAPI Specification (formerly called the Swagger Specification).

The specification creates the RESTful contract for your API, detailing all of its resources and operations in a human and machine readable format for easy development, discovery, and integration.

```
<dependency>  
  <groupId>io.springfox</groupId>  
  <artifactId>springfox-swagger2</artifactId>  
  <version>2.4.0</version> </dependency>
```

```
<dependency> <groupId>io.springfox</groupId>  
  <artifactId>springfox-swagger-ui</artifactId>  
  <version>2.4.0</version>  
</dependency>
```



```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2);
    }
}
```

```
public static final Contact DEFAULT_CONTACT = new Contact(
    "Ranga Karanam", "http://www.in28minutes.com", "in28minute

public static final ApiInfo DEFAULT_API_INFO = new
    ApiInfo( "Awesome API Title", "Awesome API Description",
    "1.0", "urn:tos", DEFAULT_CONTACT,
    "Apache 2.0", "http://www.apache.org/licenses/LICENSE-2.0"

private static final Set<String> DEFAULT_PRODUCES_AND_CONSUMES
    new HashSet<String>(Arrays.asList("application/json",
        "application/xml"));
```

@Bean

```
public Docket api() {  
    return new Docket(DocumentationType.SWAGGER_2)  
        .apiInfo(DEFAULT_API_INFO)  
        .produces(DEFAULT_PRODUCES_AND_CONSUMES)  
        .consumes(DEFAULT_PRODUCES_AND_CONSUMES);  
}
```

# Customizing

```
@ApiModelProperty(description="All details about the user.") @Entity  
public class User {  
  
    @Size(min=2, message="Name should have at least 2 characters")  
    @ApiModelProperty(notes="Name should have at least 2 characters")  
    private String name;  
}
```

# Questions

- What is "Representation" of a Resource?
- What is Content Negotiation?
- Which HTTP Header is used for Content Negotiation?
- How do we implement it using Spring Boot?
- How do you add XML support to your RESTful Services built with Spring Boot?

- A resource can have different representations
  - XML
  - HTML
  - JSON

# GET <http://localhost:8080/users>

```
[  
  {  
    "id": 2,  
    "name": "Eve",  
    "birthDate": "2017-07-19T04:40:20.796+0000"  
  },  
  {  
    "id": 3,  
    "name": "Jack",  
    "birthDate": "2017-07-19T04:40:20.796+0000"  
  }  
]
```

**GET** <http://localhost:8080/users>

- Accept application/xml

```
<List>
  <item>
    <id>2</id>
    <name>Eve</name>
    <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
  </item>
  <item>
    <id>3</id>
    <name>Jack</name> <birthDate>2017-07-
    19T10:25:20.450+0000</birthDate>
  </item>
</List>
```



```
<dependency>  
  <groupId>com.fasterxml.jackson.dataformat</groupId>  
  <artifactId>jackson-dataformat-xml</artifactId>  
</dependency>
```

# Questions

- How do you implement Exception Handling for RESTful Web Services?
- What are the different error status that you would return in RESTful Web Services?
- How would you implement them using Spring Boot?
- How do you handle Validation Errors with RESTful Web Services?

# Response Status

- 200 - SUCCESS
- 201 - CREATED
- 404 - RESOURCE NOT FOUND
- 400 - BAD REQUEST
- 401 - UNAUTHORIZED
- 500 - SERVER ERROR

# GET <http://localhost:8080/users/1000>

- Get request to a non existing resource.

```
{  
  "timestamp": "2017-07-19T05:28:37.534+000"  
  "status": 500,  
  "error": "Not Found",  
  "message": "id-500",  
  "path": "/users/500"  
}
```

```
@ResponseStatus(HttpStatus.NOT_FOUND)  
public class UserNotFoundException extends RuntimeException {
```

# GET <http://localhost:8080/users/1000>

- Get request to a non existing resource.
- Default Spring Boot Structure.

```
{  
  "timestamp": "2017-07-19T05:28:37.534+000"  
  "status": 404,  
  "error": "Not Found",  
  "message": "id-500",  
  "path": "/users/500"  
}
```

# GET <http://localhost:8080/users/1000>

- Get request to a non existing resource.
- The response shows a Customized Message Structure

```
{  
  "timestamp": "2017-07-19T05:31:01.961+0000",  
  "message": "id-500",  
  "details": "Any details you would want to add"  
}
```

```
public class ExceptionResponse {  
    private Date timestamp;  
    private String message;  
    private String details;
```



```
@ControllerAdvice  
@RestController  
public class CustomizedResponseEntityExceptionHandler  
        extends ResponseEntityExceptionHandler {
```

[illegible]

```
@ExceptionHandler(UserNotFoundException.class)
public final ResponseEntity<Object> handleUserNotFoundExcept
    UserNotFoundException ex, WebRequest request) {
    ExceptionResponse exceptionResponse =
        new ExceptionResponse(new Date(), ex.getMessage(),
                               request.getDescription(false));

    return new ResponseEntity(exceptionResponse,
                              HttpStatus.NOT_FOUND);
}
```

# POST <http://localhost:8080/users> with Validation Errors

```
{  
  "name": "R",  
  "birthDate": "2000-07-19T04:29:24.054+0000"  
}
```

## Response - 400 Bad Request

```
{  
  "timestamp": "2017-07-19T09:00:27.912+0000", "message":  
  "Validation Failed",  
  "details": "org.springframework.validation.BeanPropertyBind  
1 errors\nField error in object 'user' on field 'name': re  
default message [Name should have at least 2 characters]"  
}
```

```
@Entity
public class User {

    @Id
    @GeneratedValue
    private Integer id;

    @Size(min=2, message="Name should have atleast 2 characters")
    @ApiModelProperty(notes="Name should have atleast 2 characters")
    private String name;

    @Past
    @ApiModelProperty(notes="Birth date should be in the past")
    private Date birthDate;
```

```
@PostMapping("/users")  
public ResponseEntity<Object>  
    createUser(@Valid @RequestBody User user) {
```

@Override

```
protected ResponseEntity<Object> handleMethodArgumentNotValid(  
    MethodArgumentNotValidException ex,  
    HttpHeaders headers, HttpStatus status, WebRequest req  
    ExceptionResponse exceptionResponse = new ExceptionResponse(  
        new Date(), "Validation Failed", ex.getBindingResult().  
    return new ResponseEntity(exceptionResponse, HttpStatus.BAD_  
}
```



# Questions

- Why do we need Versioning for RESTful Web Services?
- What are the versioning options that are available?
- How do you implement Versioning for RESTful Web Services?

```
public class PersonV1 {  
    private String name;
```

```
public class PersonV2 {  
    private Name name;
```

```
public class Name {  
    private String firstName;  
    private String lastName;
```

# Versioning Options

- URI Versioning <http://localhost:8080/v1/person>
  - <http://localhost:8080/v2/person>

## Request Param Versioning

- <http://localhost:8080/person/param?version=1>
  - <http://localhost:8080/person/param?version=2>
  -

# Versioning Options

- Header Versioning
  - <http://localhost:8080/person/header>
    - headers=[X-API-VERSION=1]
  - <http://localhost:8080/person/header>
    - headers=[X-API-VERSION=2]

# Versioning Options

- MIME Type or Accept Header Versioning
  - <http://localhost:8080/person/produces>
    - produces=[application/vnd.company.app-v1+json]
  - <http://localhost:8080/person/produces>
    - produces=[application/vnd.company.app-v2+json]

```
@GetMapping("v1/person")  
public PersonV1 personV1() {  
    return new PersonV1("Bob Charlie");  
}
```

```
@GetMapping("v2/person")  
public PersonV2 personV2() {  
    return new PersonV2(new Name("Bob", "Charlie"));  
}
```

```
@GetMapping(value = "/person/param",  
              params = "version=1")  
public PersonV1 paramV1() {  
    return new PersonV1("Bob Charlie");  
}
```

```
@GetMapping(value = "/person/param",  
              params = "version=2")  
public PersonV2 paramV2() {  
    return new PersonV2(new Name("Bob", "Charlie"));  
}
```

```
@GetMapping(value = "/person/header",
              headers = "X-API-VERSION=1")
public PersonV1 headerV1() {
    return new PersonV1("Bob Charlie");
}

@GetMapping(value = "/person/header",
              headers = "X-API-VERSION=2")
public PersonV2 headerV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
}
```



```
@GetMapping(value = "/person/produces",  
    produces = "application/vnd.company.app-v1+json")  
public PersonV1 producesV1() {  
    return new PersonV1("Bob Charlie");  
}
```

```
@GetMapping(value = "/person/produces",  
    produces = "application/vnd.company.app-v2+json")  
public PersonV2 producesV2() {  
    return new PersonV2(new Name("Bob", "Charlie"));  
}
```

# Versioning

- Media type versioning (a.k.a “content negotiation” or “accept header”)
  - GitHub
- (Custom) headers versioning
  - Microso
- URI Versioning
  - Twitter
- Request Parameter versioning
  - Amazon

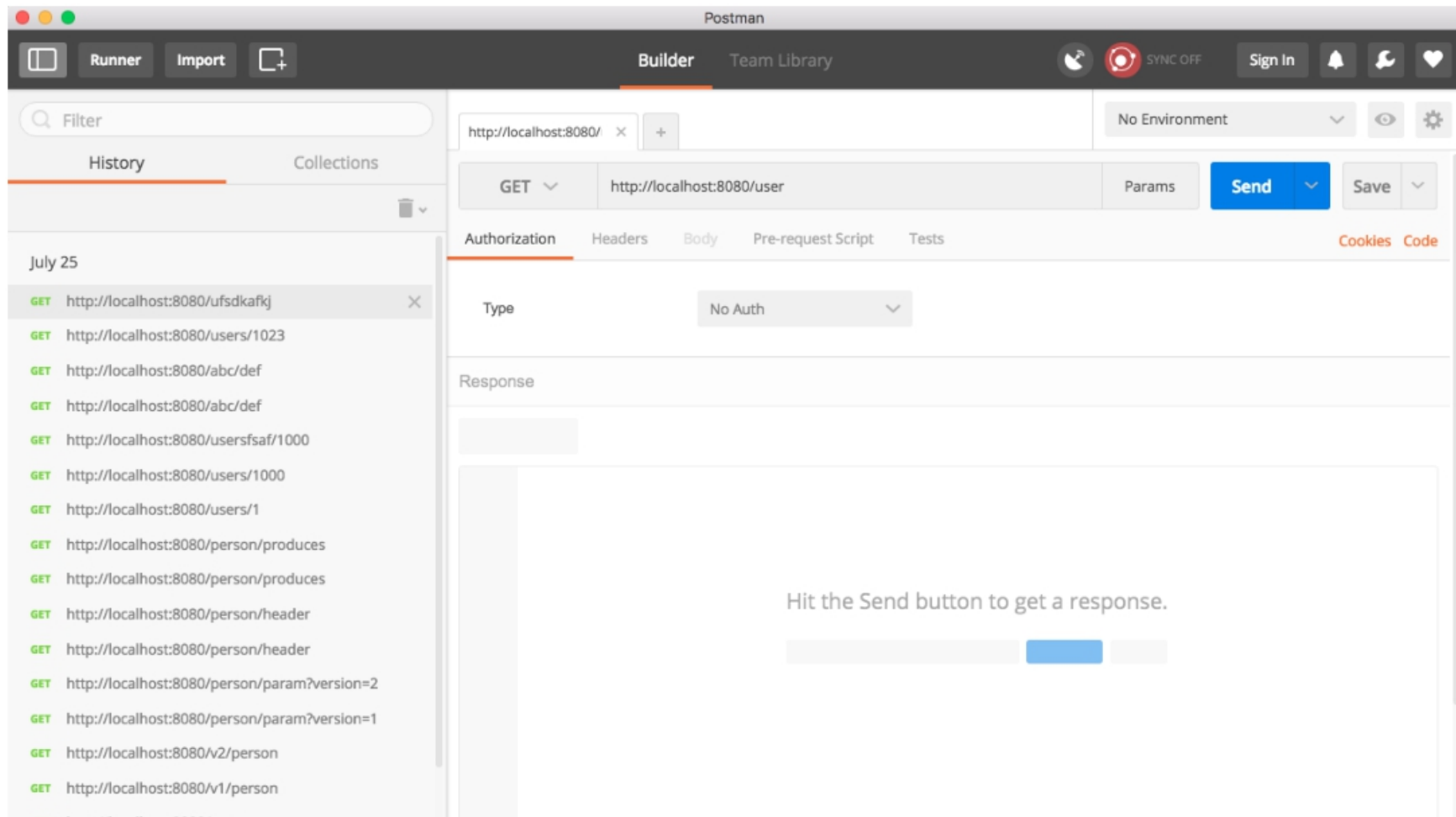
# Versioning

- Factors
  - URI Pollution
  - Misuse of HTTP Headers
  - Caching
  - Can we execute the request on the browser?
  - API Documentation
- No Perfect Solution

# Questions

- Which is the client you use to test RESTful Web Services?
- How do you use Postman to execute RESTful Service Requests?
- How can you send Request Headers using Postman?

**Postman**



http://localhost:8080/ x +

No Environment v

eye

gear

GET v

http://localhost:8080/user

Params

Send v

Save v

Authorization •

Headers

Body

Pre-request Script

Tests

Cookies

Code

Type

Basic Auth v

Clear

Update Request

Username

postman

The authorization header will be generated and added as a custom header

Password

.....

☒ Save helper data to request

☐ Show Password

http://localhost:8080/

×

+

No Environment

⌵

👁

⚙

GET

⌵

http://localhost:8080/user

Params

Send

⌵

Save

⌵

Authorization

Headers

Body

Pre-request Script

Tests

Cookies

Code

Type



No Auth

⌵

Response



http://localhost:8080/ x +

No Environment v  

GET v

http://localhost:8080/user

Params

Send v

Save v

Authorization ●

Headers (2)





Body

Pre-request Script

Tests

Cookies

Code

<input checked="" type="checkbox"/>	Accept	application/json	 	Bulk Edit	Presets v
<input checked="" type="checkbox"/>	X-API-VERSION	1	 		
	key	value			

**Thank You**