THE SPRING FRAMEWORK

Julia Damerow (ASU), DHTech Virtual Workshop, Feb. 25, 2019

AGENDA

- The Spring Framework
- Dependency Injection
- A Webapp with Spring
- Spring Security
- Spring Data
- Spring AOP
- Spring in Action

WHAT IS SPRING?

- https://spring.io/
- Multitude of projects centered around the Spring Framework
- At its core, a dependency injection framework
- Different modules provide support for data access, authentication/ authorization, aspect-oriented programming, etc.
- Spring Boot to “just run” your application.
- Open Source developed by Pivotal Software

WHAT ARE THE STEPS?

- Add dependencies (e.g. spring-context, spring-web, or spring-webmvc)
- Configure application (either XML or Java annotations)
- Write services, components, etc.
- Tell Spring where to inject what

DEPENDENCY INJECTION

This comic was created at www.MakeBeliefsComix.com. Go there and make one now!

HOW TO DO IT IN SPRING?

Tell Spring what we have:

- @Service, @Component
- @Bean

Tell Spring what we want:

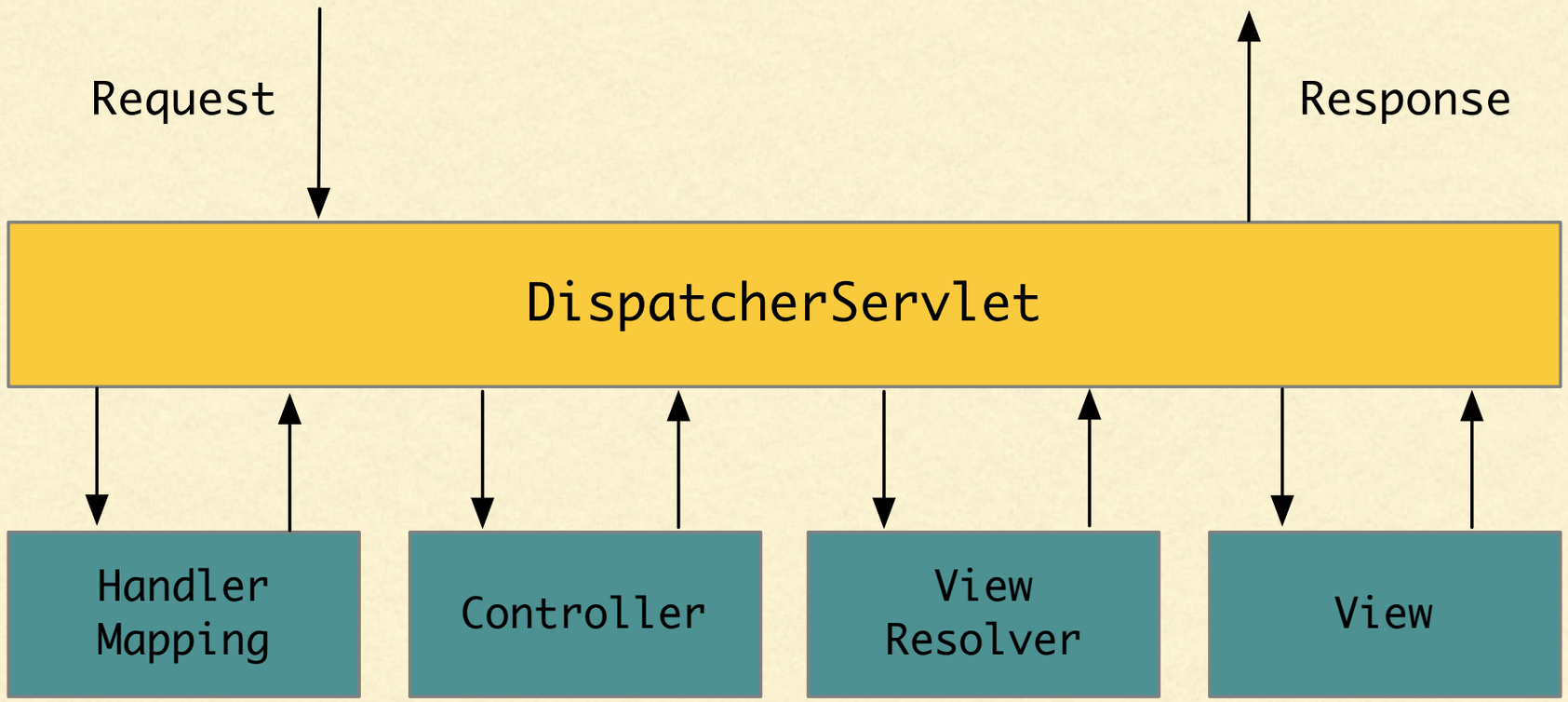
- @Autowired

EXAMPLE

```
@Service
class AutoFactory {

    @Autowired
    private EngineFactory engineFactory;

    public Car build() {
      Car car = new Car();
      Car.setEngine(engineFactory.build());

      return car;
    }
}
```

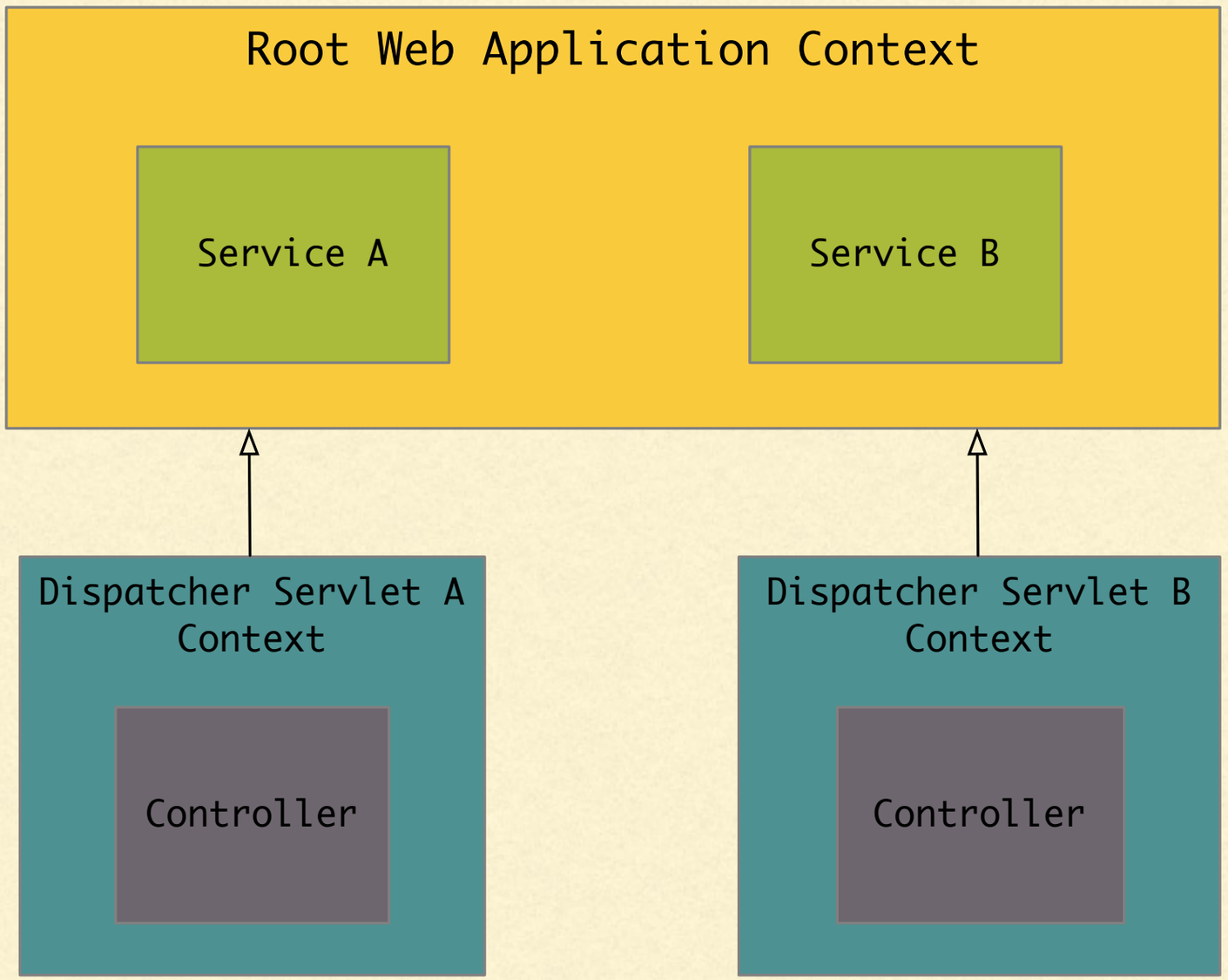
```
@Service
class EngineFactory {

    public Engine build() {
      Engine engine = new Engine();
      return engine;
    }
}
```

DEVELOPING A WEBAPP



Adapted from: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

SPRING CONTEXTS



ROOT CONTEXT

- Register the root context shared by all servlets and filters in web.xml

```
<context-param>
    <param-name>contextClass</param-name>
    <param-value>
        org.springframework.web.context.support.AnnotationConfigWebApplicationContext
    </param-value>
</context-param>

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>my.mvc.app.RootConfig</param-value>
</context-param>

<listener>
    <listener-class>
         org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

DISPATCHER CONTEXT

- Register a dispatcher servlet in web.xml with its own context.

```
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
        <param-name>contextClass</param-name>
        <param-value>
            org.springframework.web.context.support.AnnotationConfigWebApplicationContext
        </param-value>
    </init-param>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>my.mvc.app.MvcConfig</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

JAVA CONFIG

```
@Configuration
@EnableWebMvc
@ComponentScan(“my.mvc.app”)
public class RootConfig {



}
```

```
@Configuration
public class MvcConfig {

}
```

CONTROLLERS

- Handle incoming requests at specific endpoints

```
@Controller
public class HomeController {

    @RequestMapping("/")
    public String home(Model model) {
        return "home";
    }

    @RequestMapping("/date")
    public String date(Model model) {
        model.addAttribute("date", OffsetDateTime.now());
        return "home";
    }
}
```

REQUEST MAPPINGS

- @RequestMapping(value="end/point", method="GET", …)
  - Shortcuts:
    - GetMapping
    - PostMapping
    - PutMapping
    - DeleteMapping
    - PatchMapping

VIEW RESOLVERS

- What kind of views do we have and where are they coming from?

```
@Configuration
public class MvcConfig {

    @Bean
    public ViewResolver internalResourceViewResolver() {
        InternalResourceViewResolver bean = new InternalResourceViewResolver();
        bean.setViewClass(JstlView.class);
        bean.setPrefix("/WEB-INF/views/");
        bean.setSuffix(".jsp");
        return bean;
    }
}
```

http://localhost:8080/my-webapp/date

request to '/date'

```
//@Controller

@RequestMapping("/date")
public String date(Model model) {
    model.addAttribute("date", OffsetDateTime.now());
    return "home";
}
```

```
//@Configuration - ViewResolver

bean.setPrefix("/WEB-INF/views/");
bean.setSuffix(".jsp");
```

use

src/main/webapp/WEB-INF/views/home.jsp

VIEW

- home.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<body>
<h1>Hello World!</h1>

<c:if test="${not empty date}">
${date}
</c:if>
<c:if test="${empty date}">
See the <a href="<c:url value="/date" />">date</a>.
</c:if>
</body>
</html>
```

VIEW TECHNOLOGIES

- JSP
- Thymeleaf
- Groovy
- FreeMaker
- And others…

SPRING SECURITY

- https://spring.io/projects/spring-security
- Securing Spring applications
- Authenticating and authorizing users
- XML-based or annotation-based configuration
- Modules for LDAP, CAS, OAuth, OpenID, … integration

CONFIGURATION

```
@EnableWebSecurity
public class WebSecurityConfig implements WebMvcConfigurer {

    protected void configure(HttpSecurity http) throws Exception {
        http.formLogin().and()
                .authorizeRequests()
                .antMatchers("/", "/resources/**",
                        "/date").permitAll()
                .antMatchers("/users/**", "/admin/**").hasRole("ADMIN")
                .anyRequest().hasRole("USER");
    }

    @Bean
    public UserDetailsService userDetailsService() throws Exception {
        InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();
        manager.createUser(User.withDefaultPasswordEncoder().username(“user")
                .password("password").roles("USER").build());
        return manager;
    }
}
```

SPRING DATA

- https://spring.io/projects/spring-data

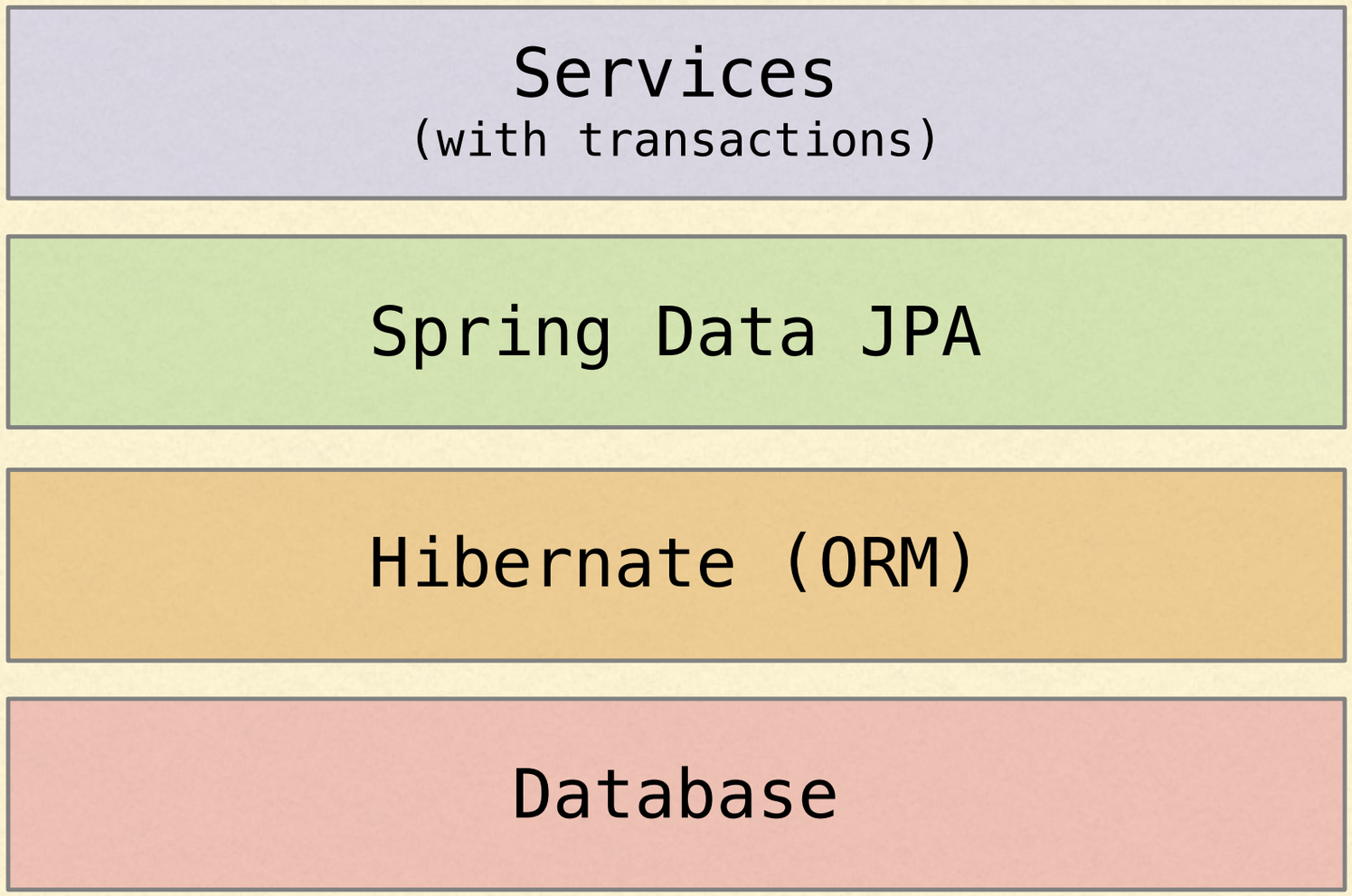
- "It makes it easy to use data access technologies, relational and non-relational databases, map-reduce frameworks, and cloud-based data services." (https://spring.io/projects/spring-data)

- Subprojects that provide support for JDBC, JPA, MongoDB, SolR, and many more

THE REPOSITORY INTERFACE

```
public interface CrudRepository<T, ID extends Serializable>
  extends Repository<T, ID> {

  <S extends T> S save(S entity);

  Optional<T> findById(ID primaryKey);

  Iterable<T> findAll();

  long count();

  void delete(T entity);

  boolean existsById(ID primaryKey);

  // … more functionality omitted.
}
```

SPRING DATA JPA EXAMPLE



SPRING DATA JPA EXAMPLE

```java
@Entity
public class Cat {

    @Id
    @GeneratedValue
    private Long id;
    private String name;
    private String breed;

    // getter and setters
}
```

```java
public interface CatRepository extends CrudRepository<Cat, Long> {

}
```

```java
@Controller
public class HomeController {

    @Autowired
    private CatRepository repo;

    @RequestMapping("/")
    public String home(Model model) {
        model.addAttribute("cats", repo.findAll());
        return “home";
    }

   @RequestMapping("/cat")
    public String cat(Model model) {
        Cat cat = new Cat();
        cat.setName("Kitty "
              + OffsetDateTime.now().getSecond());
        cat.setBreed("DSH");
        repo.save(cat);
        return "redirect:/";
    }
}
```

ASPECT-ORIENTED PROGRAMMING

“In computing, aspect-oriented programming (AOP) is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns. It does so by adding additional behavior to existing code (an advice) without modifying the code itself, instead separately specifying which code is modified via a "pointcut" specification, such as "log all function calls when the function's name begins with ‘set’”.”

Source: https://en.wikipedia.org/wiki/Aspect-oriented_programming

WHAT CAN YOU DO?

- Logging
- Performance measurements
- Authorization checks
- …

SOME AOP CONCEPTS

- Aspect: A modularization of a concern that cuts across multiple classes.
- Join Point: A point during the execution of a program, such as the execution of a method or the handling of an exception.
- Advice: Action taken by an aspect at a particular join point.
- Pointcut: A predicate that matches join points.

```java
@Configuration
@EnableWebMvc
@EnableAspectJAutoProxy
@ComponentScan("my.mvc.app")
public class RootConfig {

}
```

```java
@Component
@Aspect
public class FirstAspect {

    private final Logger logger = LoggerFactory.getLogger(getClass());

    @Before("within(my.mvc.app..*)")
    public void always(JoinPoint jp) throws Throwable {
        logger.info("Always " + jp.getSignature());
    }
}
```

```
22 Feb 2019 10:45:42,685 INFO :
edu.asu.diging.spring5.example.aspects.FirstAspect - Always String
edu.asu.diging.spring5.example.web.HomeController.home(Model)
```

SPRING IN ACTION

HTTPS://GITHUB.COM/JDAMEROW/SPRING5-JAVA-CONFIG-EXAMPLE