Spring MVC

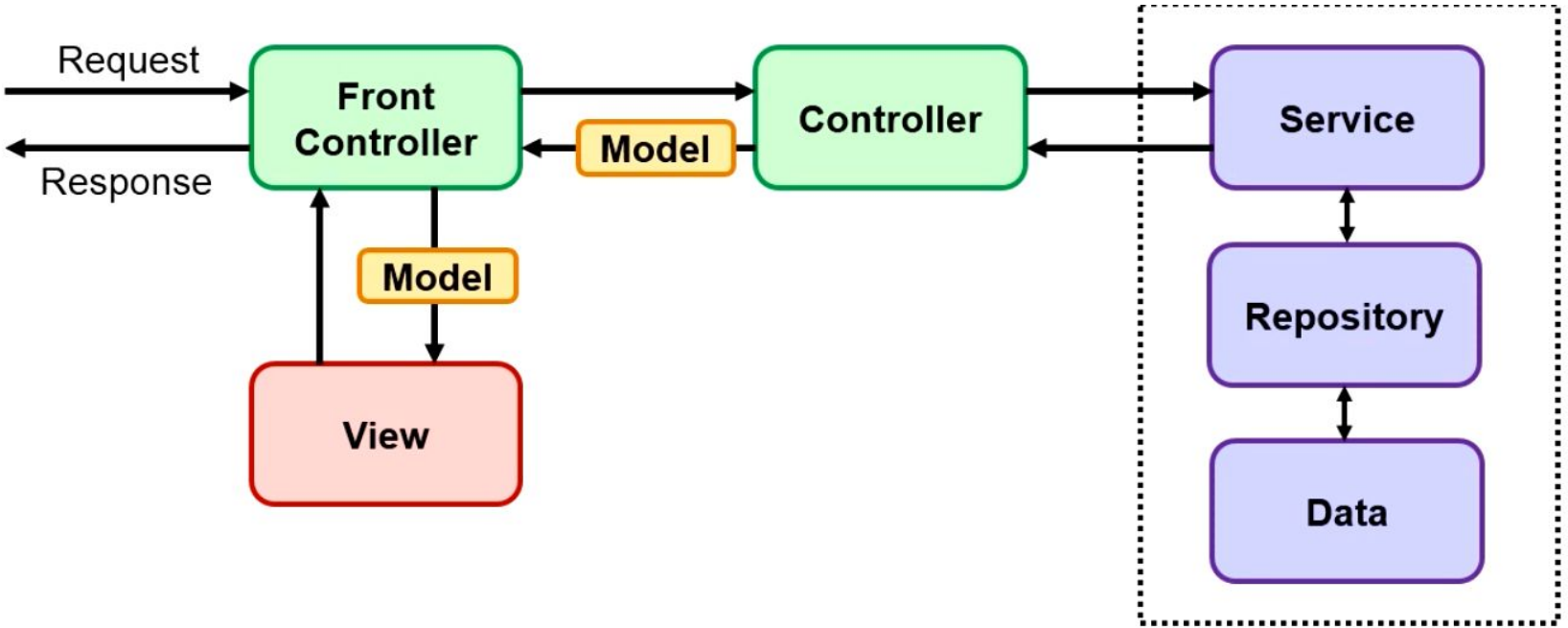
Web Frameworks

- Component - Based Framework
- Request (Action) Based Framework

Spring MVC

- Action based web framework that uses MVC pattern to build application
- Based on Spring IoC Container
- Offers a powerful suite of components for processing request and response

Spring MVC Processing



Spring MVC Features

- Data Binding
- Error Handling
- Validation
- File Upload
- Tag Library
- View Resolution
- REST Support

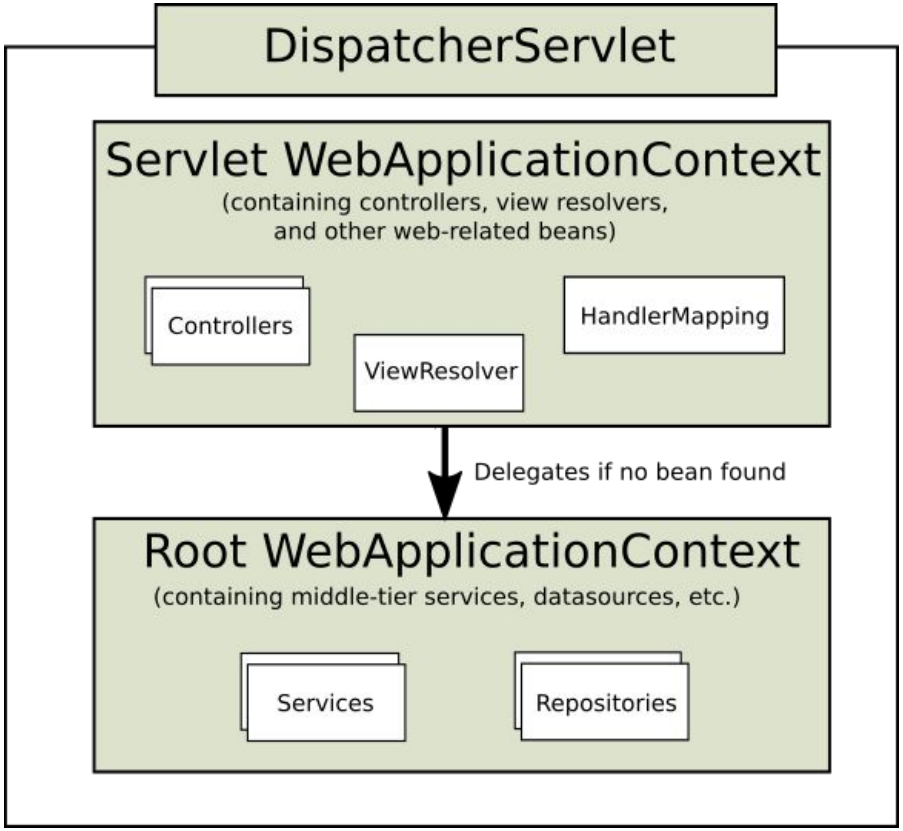
Spring Hello World Configuration

```
public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
   @Override
   protected Class<?>[] getRootConfigClasses() {
       return new Class<?>[]{
               GameConfig.class
       };
   }

   @Override
   protected Class<?>[] getServletConfigClasses() {
       return new Class<?>[]{
               WebConfig.class
       };
   }

   @Override
   protected String[] getServletMappings() {
       return new String[]{"/"};
   }
}
```

Context Hierarchies



Basic Web Context Configuration

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = {"rs.saga.web"})
public class WebConfig implements WebMvcConfigurer {

    // Serves up cached and compressed static content at /resources/* from the webapp root and classpath
    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/").setViewName("form");
    }

    @Bean
    InternalResourceViewResolver getViewResolver(){
        InternalResourceViewResolver resolver = new InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/");
        resolver.setSuffix(".jsp" );
        return resolver;
    }
}
```

Controller

- @EnableWebMvc - Adding this annotation to an @Configuration class imports the Spring MVC configuration

```
@EnableWebMvc
@ComponentScan(basePackages = {"rs.saga.web"})
public class WebConfig implements WebMvcConfigurer
```

- @Controller - Indicates that an annotated class is a "Controller" (e.g. a web controller). This annotation serves as a specialization of @Component, allowing for implementation classes to be autodetected through classpath scanning. It is typically used in combination with annotated handler methods based on the RequestMapping annotation

- @RequestMapping - Annotation for mapping web requests onto methods in request-handling classes with flexible method signatures

```
Request URL: http://localhost:8080/player/list


@Controller @RequestMapping("/player")
public class PlayerController {

   @RequestMapping(value = "/list", method = RequestMethod.GET)
   public String list() {
   }
}
```

Model

- defines a holder for model attributes

```
@RequestMapping(value = "", method = RequestMethod.GET)
public String list(@RequestParam Long id, Model model) {
   model.addAttribute("player", playerService.findPlayer(id));
   return "player/hello";
}
```

- access model from view

```
<h1>Hello ${requestScope.player.firstName}</h1>
```

View

- view rendering technology e.g. JSP, Thymeleaf, Freemarker, Velocity, Tapestry

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
   <title>Player detail</title>
</head>
<body>
<h1>Hello ${requestScope.player.firstName}</h1>
</body>
</html>
```