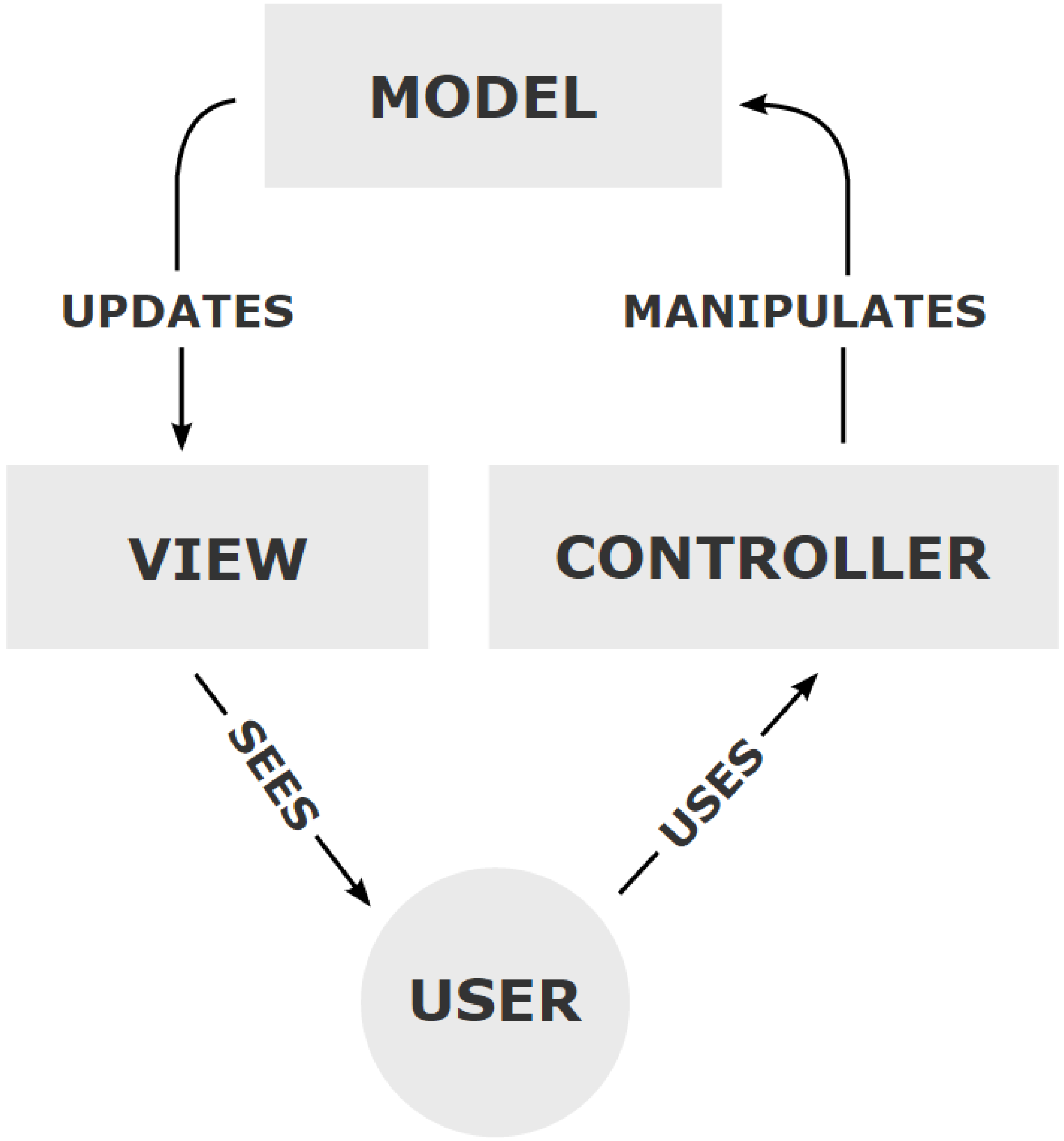
Module 3
Java Web Applications
Spring MVC

Web Layer Integration

- Spring provides support in the Web Layer
  - Spring MVC
- You can use Spring with any Java Web framework
  - Java Server Faces (JSF)
  - Integration might be provided by Spring or by the web framework

MVC Design Pattern



Spring Application Context in Webapps

- Spring can be initialized within a web application
- Uses a standard servlet listener
  - Initialization occurs before any servlet execute
  - Application is ready to process client requests
  - Spring context is closed when the application is stopped

Configuration in web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/librarymaster-application-config.xml</param-value>
  </context-param>

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

Dependency Injection

- Example using Spring MVC

```
@Controller
public class BookController {

    private LibraryService libraryService;

    @Autowired
    public BookController(LibraryService libraryService) {
        this.libraryService = libraryService;
    }

    @RequestMapping("/allBooks")
    public void allBooks(Model model) {
        model.addAttribute("books", libraryService.getAllBooks());
    }
}
```

Spring Web MVC

- Spring’s Web framework
  - Uses Spring for its own configuration
  - Controllers are Spring beans
  - Annotations based model
  - Testability support
- Builds on the Java Servlet API
- The core platform for developing Web applications in Spring

Spring Web Flow

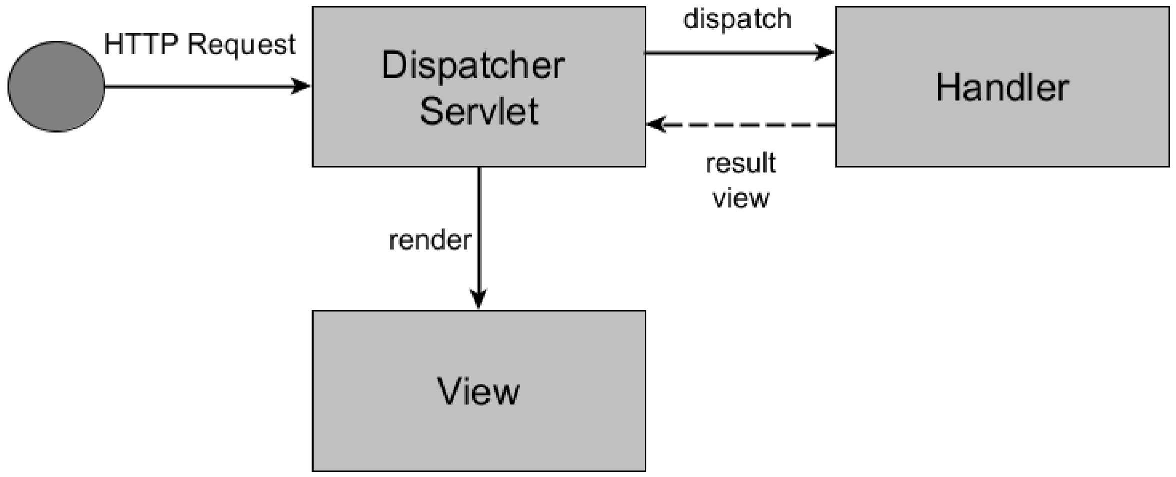
- Plugs into Spring MVC as a Controller for implementing stateful view flows
  - Checks that user follows the right navigation path
  - Handles back browser button and multiple window issues
  - Provides scopes beyond session and request scopes
    - Conversation scope
- View flows are declared in xml

Spring and JSF Integration

- Spring centric integration
  - Provides by Spring faces
- JSF centric integration
  - Spring plugs in as a JSF Managed Beans provider

Web Request Handling

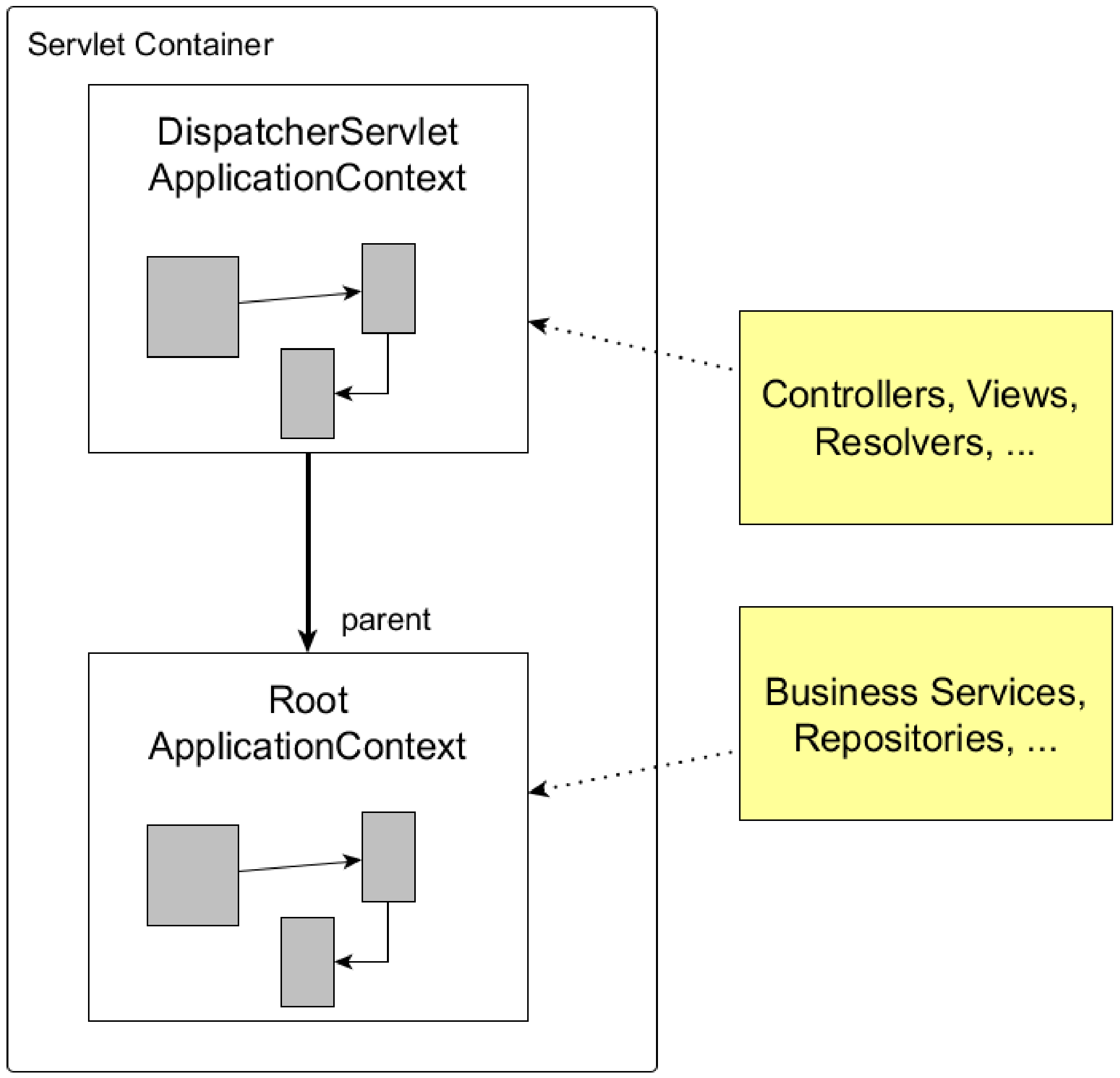
- Web request handling is rather simple
  - Based on incoming URL
  - You need to call method
  - After which the return value
  - Needs to be rendered using a view



DispatcherServlet

- Acts as front controller
  - Coordinates all requests handling activities
- Delegates to web infrastructure beans
- Invokes user web components
- Fully customizable
- Defined in web.xml
- Creates separate application context
  - Full access to parent application context

Servlet Application Context


Servlet Container
DispatcherServlet ApplicationContext
parent
Root ApplicationContext
Controllers, Views, Resolvers, ...
Business Services, Repositories, ...

Controllers as Request Handlers

- Handlers you define are typically called Controllers and and annotated by @Controller
- @RequestMapping tells Spring what method to execute when processing a particular request

```
@Controller
public class BookController {

    @RequestMapping("/allBooks")
    public void allBooks(Model model) {
        model.addAttribute("books", libraryService.getAllBooks());
    }
}
```

Extracting Request Parameters

- Use @RequestParam annotation
  - Extracts parameter from the request
  - Performs type conversion

```
@Controller
public class BookController {

    @RequestMapping("/book")
    public void showBook(@RequestParam("bookId") long bookId) {
        ...
    }
}
```

URI Templates

- Values can be extracted from request URLs
- Allows clean URLs without request parameters

```
@Controller
public class BookController {

    @RequestMapping("/book/{bookId}")
    public void showBook(@PathVariable(“bookId”) long bookId) {
        ...
    }
}
```

Selecting a View

- Controllers typically return a view name
  - By default view name is interpreted as path to JSP page
- Controllers may return null (void)
  - DispatcherServlet selects a default view based on the request URL
- Controllers may return concrete view
  - Path to JSP file

View Resolvers

- DispatcherServlet delegates to a ViewResolvers to map returned view names to View implementations
- The default ViewResolver treats the view name as a Web application relative file path
- You can override this behavior by registering ViewResolver bean with the DispatcherServlet

Useful Stuff

- Spring MVC template engines comparison
  - https://github.com/jreijn/spring-comparing-template-engines