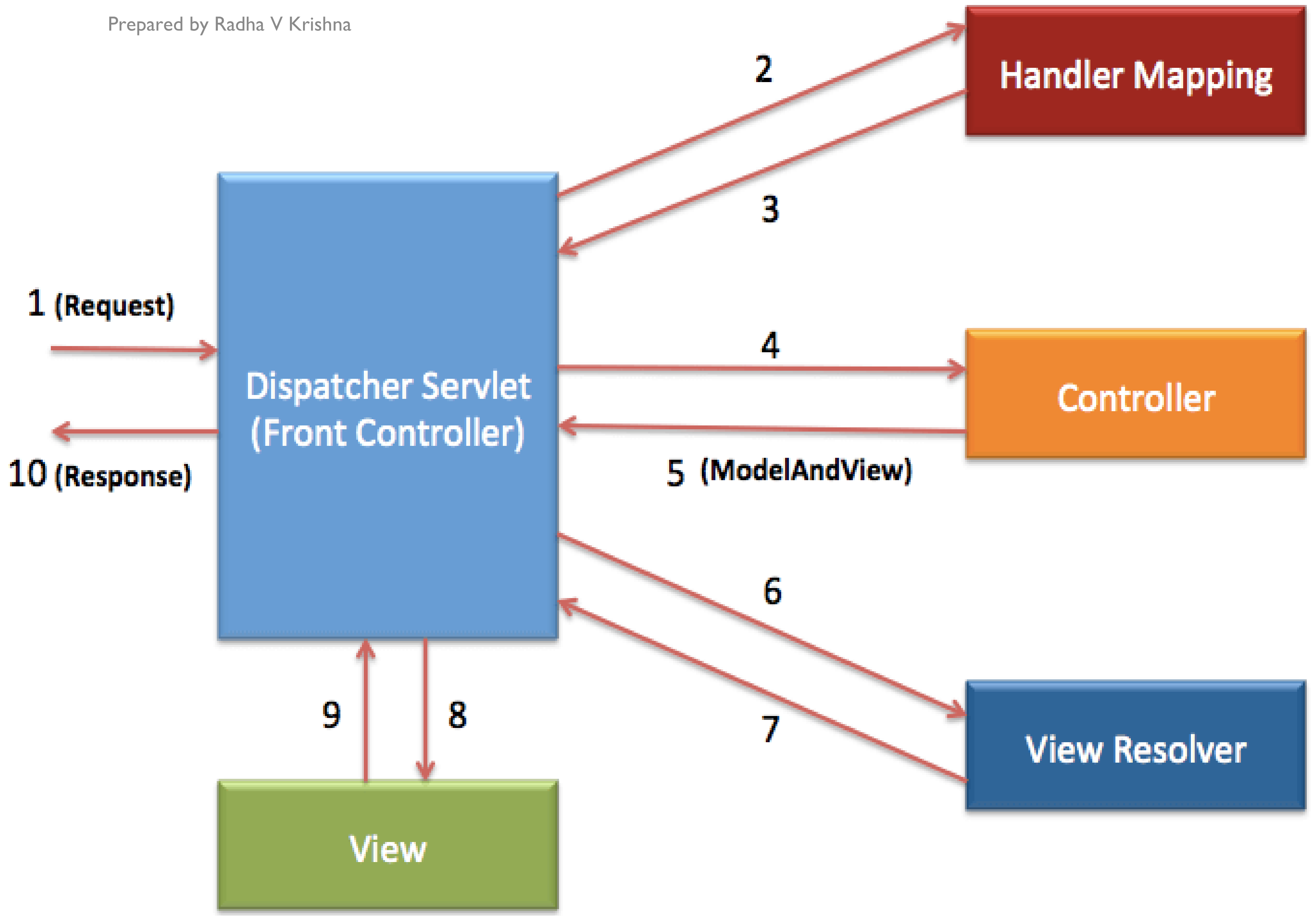
Prepared by Radha V Krishna

SPRING MVC

INTRODUCTION TO SPRING WEB MVC FRAMEWORK

- The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale and theme resolution as well as support for uploading files.

Prepared by Radha V Krishna



Spring MVC Dependencies

```
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.0.3.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>4.0.3.RELEASE</version>
</dependency>
```

DISPATCHERSERVLET

```
<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_2_5.xsd"
version="2.5">

<display-name>SpringSampleProject</display-name>

<servlet>
<servlet-name>spring</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
<servlet-name>spring</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>

</web-app>
```

Configuration file
<servlet-name>-servlet.xml

In this case :
spring-servlet.xml
(location : WEB-INF)

```
beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

 xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

        http://www.springframework.org/schema/beans

        http://www.springframework.org/schema/beans/spring-beans.xsd

        http://www.springframework.org/schema/context

        http://www.springframework.org/schema/context/spring-context.xsd

        http://www.springframework.org/schema/mvc

        http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

 <mvc:annotation-driven/>

<context:component-scan base-package="com.controllers.*" />
```

Enables Annotaion for autowiring

To scan the packages

VIEW RESOLVER CONFIGURATION

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

Prepared by Radha V Krishna

CONTROLLER

```
<bean
class="org.springframework.web.servlet.v
iew.InternalResourceViewResolver">
<property name="prefix">
<value>/WEB-INF/pages/</value>
</property>
<property name="suffix">
<value>.jsp</value>
</property>
</bean>
```

```
@Controller
public class HelloController {

@RequestMapping("/")
public String sayHello()
{
return "hello";
}
```

hello.jsp
In /WEB-INF/pages/

http://localhost:8080/BookStore/
Where BookStore is the Project Name

Web App using Java Configuration

```
// In place of web.xml
public class WebServletConfiguration implements WebApplicationInitializer{
    public void onStartup(ServletContext ctx) throws ServletException {
        AnnotationConfigWebApplicationContext webCtx = new
AnnotationConfigWebApplicationContext();
        webCtx.register(MyConfiguration.class);
        webCtx.setServletContext(ctx);
        ServletRegistration.Dynamic servlet = ctx.addServlet("dispatcher", new
DispatcherServlet(webCtx));
        servlet.setLoadOnStartup(1);
        servlet.addMapping("/");
    }
}
```

```
// In place of spring-servlet.xml
@Configuration
@EnableWebMvc
@ComponentScan("com.training.bookstore")
public class MyConfiguration extends WebMvcConfigurerAdapter{


        @Override public void
configureDefaultServletHandling(DefaultServletHandlerConfigurer
configurer) {
        configurer.enable(); }


}
```

STEPS FOR SIMPLE MVC IN SPRING

- Add dependencies in Maven/Dynamic web project
- Configure DispatcherServlet in web.xml
- Create configuration xml file
- add the required elements
- Create controller
- Add Request mapping
- Run the server
- Type the url in browser

```
//reading parameters from Http Get request

@RequestMapping("/getbook")
public String getbook(@RequestParam("isbn")
String isbn,Model model)
{
Book book = bookService.getBook(isbn);
model.addAttribute("book", book);
System.out.println(book);
return "book";

}
```

```
//Creating request attribute

@RequestMapping("/home")
public String start(Model model)
{
model.addAttribute("book",new Book("21","C"));
return "bookform";
}
```

```
//bookform.jsp

<%@ page isELIgnored="false" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
${book.isbn}<br>
${book.title}
```

Prints 21 C

FOR DATABASE CONFIGURATION

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"

destroy-method="close">

<property name="driverClassName" value="com.mysql.jdbc.Driver" />

<property name="url"

value="jdbc:mysql://localhost:3306/training" />

<property name="username" value="root" />

<property name="password" value="root" />

</bean>
```

```
<dependency>
<groupId>commons-dbcp</groupId>
<artifactId>commons-dbcp</artifactId>
<version>1.4</version>
</dependency>
```

MAPPING HIBERNATE PROPERTIES

```
<bean id="hibernate4AnnotatedSessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
<property name="dataSource" ref="dataSource" />
<property name="annotatedClasses">
<list>
<value>com.classes.Book</value>
</list>
</property>
<property name="hibernateProperties">
<props>
<prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect
</prop>
<prop key="hibernate.show_sql">true</prop>

<prop key="hbm2ddl.auto">create</prop>
</props>
</property>
</bean>
```

Creates a table

```
<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.mysql.jdbc.Driver" />
      <property name="url" value="jdbc:mysql://localhost:3306/jpmc1" />
      <property name="username" value="root" />
      <property name="password" value="root" />
  </bean>
```

To integrate with JPA

```
 <bean id="myEmf"

class="org.springframework.orm.jpa.LocalContainerEntityMana
gerFactoryBean">
       <property name="dataSource" ref="dataSource" />

       <property name="jpaProperties">
         <props>
             <prop key="hibernate.hbm2ddl.auto">update</prop>
            <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</
prop>
            <prop key="hibernate.show_sql">true</prop>
         </props>
       </property>
   </bean>
```

ContextLoaderListener – Root application context

Prepared by Radha V Krishna

ContextLoaderListener creates the root application context and will be shared with child contexts created by all DispatcherServlet contexts. You can have only one entry of this in web.xml.

web.xml

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

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/applicationContext.xml</param-value>
</context-param>
```

This will be the configuration file now

SAMPLE DAO IMPLEMENTATIONS

```
@Autowired
private SessionFactory sf;

public List<Book> getAllBooks() {

Session s= sf.openSession();
return s.createQuery("from Book").list();
}

public boolean addBook(Book book) {
Session s=sf.openSession();
s.getTransaction().begin();
s.save(book);
s.getTransaction().commit();
return true;
}
```

Dao Implementation
To retrieve records

Adds Records to DB

Follow the same steps for other operations

```java
public interface BookDaoI {
            public boolean addBook(Book book);
            public Book getBook(String isbn);
            public Book updateBook(Book book);
            public List<Book> getAllBooks();
            public Book deleteBook(Book book);
      }
}
```

```xml
<bean id="transactionManager"

      class="org.springframework.orm.hibernate4.Hibe
rnateTransactionManager">
            <property name="sessionFactory"
ref="sessionFactory" />
      </bean>
<tx:annotation-driven transaction-
manager="transactionManager" proxy-target-
class="true"/>
```

```java
//Using hibernate

@Repository
public class BookDao implements BookDaoI {

      @Autowired
      private SessionFactory sessionFactory;
      public boolean addBook(Book book) {

            Session session = sessionFactory.openSession();
            //Transaction tx=session.beginTransaction();
            session.persist(book);
//          tx.commit();
            return false;
      }
```

```
//Using JPA
```

```
@Repository
public class BookDao implements BookDaoI {

	@PersistenceContext
	private EntityManager em;

	@Transactional
	@Override
	public boolean addBook(Book book) {
		System.out.println("In Book Dao");
		//em.getTransaction().begin();
		em.persist(book);
		//em.getTransaction().commit();
		return true;
	}
```

```
<bean id="transactionManager"

class="org.springframework.orm.jpa.JpaTransactionManager">
	<property name="entityManagerFactory" ref="myEmf" />
   </bean>
   <tx:annotation-driven transaction-
manager="transactionManager" proxy-target-class="true"/>
```

Other Operations using JPA

```
@Override
	public Book getBook(String isbn) {
		// TODO Auto-generated method stub
		return em.find(Book.class, isbn);
	}

	@Override
	public Book updateBook(Book bookUp) {
		// TODO Auto-generated method stub
		em.remove(em.find(Book.class,bookUp.getIsbn()));
		em.persist(bookUp);
		return bookUp;
	}
```

```
@Override
public List<Book> getAllBooks() {

return em.createQuery("from Book b").getResultList();

}

@Override
public void deleteBook(String isbn) {

	em.remove(em.find(Book.class, isbn));
}
```

```
// Sample Spring UI form
<html>
<head>
<title>
</title>
<body>
<a href="getallbooks">Go to Book Store</a>
<br><br>
   <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
       <form:form name="f1" method="post" action="bookdetails" modelAttribute="book">
        <table >
         <tr>
          <td>Isbn : </td>
          <td><form:input path="isbn"  />
          <form:errors path="isbn" cssStyle="color: #ff0000;"/>
          </td>
         </tr>
        </table>
        <input type="submit" value="add"/>
       </form:form>

  </body>
</html>
```

Spring UI Forms and Data Binding

```
//bookform.jsp
      <form:form name="f1" method="get" action="addbook"
modelAttribute="book" >
       <table >
        <tr>
         <td>Isbn : </td>
         <td><form:input path="isbn"  />
          <form:errors path="isbn" cssStyle="color:
#ff0000;"/></td>
```

Isbn is an attribute in bean Book

```
@RequestMapping("/addbook")
      public String addBook(@Valid @ModelAttribute("book")  Book
book,
                   BindingResult result,Model model)
      {
            if(result.hasErrors()) return "bookform";
            boolean status = bookService.addBook(book);
            if(status)
                  model.addAttribute("message","Book Added..");
            else
                  model.addAttribute("message","Error in adding Book");

            return "success";
      }
```