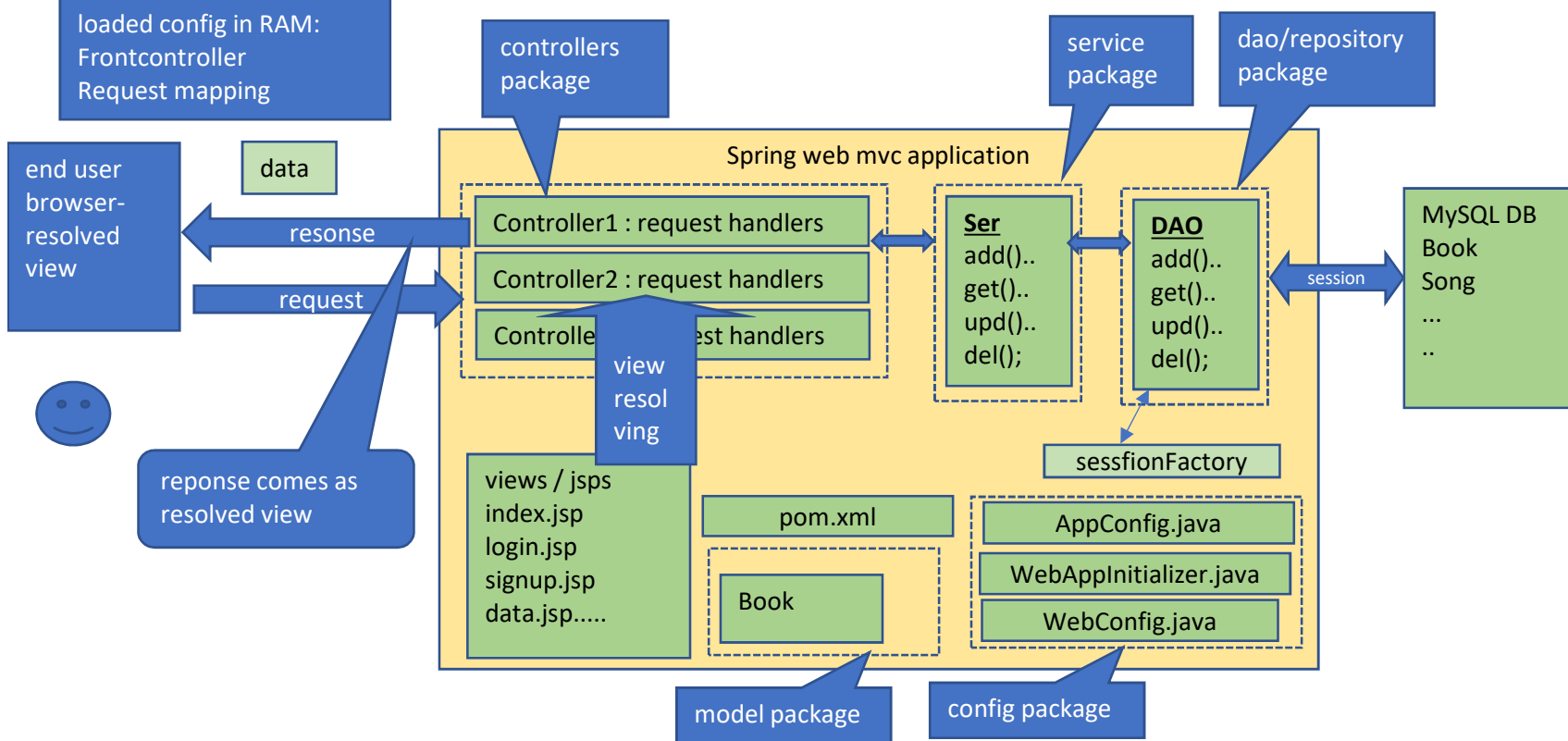
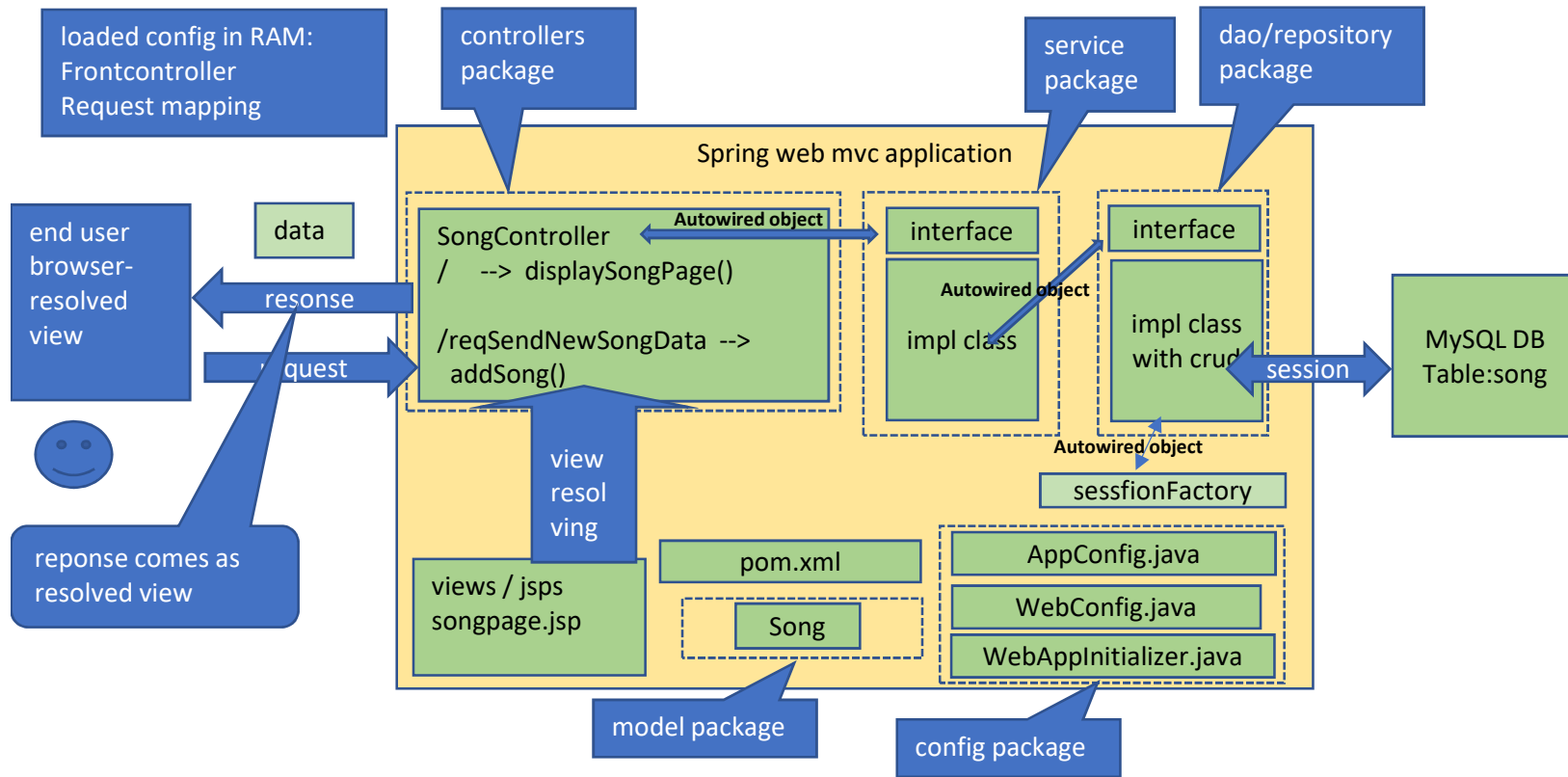


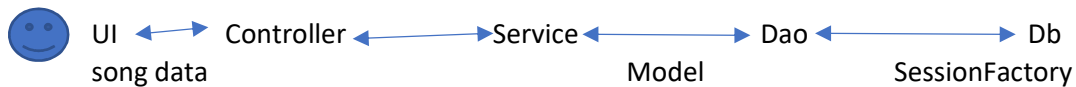
Target



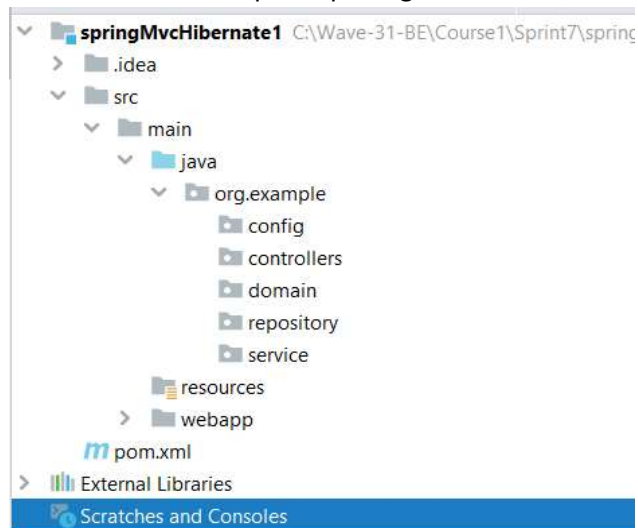
Achieved



Spring WebMvc application with Hibernate



Step 1 Create maven webapp
create required packages



Step 2 Add required dependencies in pom

spring-webmvc
jstl
javax.servlet-api

spring-orm
hibernate-core
mysql

```
18 <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
19 <dependency>
20 <groupId>org.springframework</groupId>
21 <artifactId>spring-webmvc</artifactId>
22 <version>5.3.18</version>
23 </dependency>
24
25 <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
26 <dependency>
27 <groupId>javax.servlet</groupId>
28 <artifactId>jstl</artifactId>
29 <version>1.2</version>
30 </dependency>
31
32 <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
33 <dependency>
34 <groupId>javax.servlet</groupId>
35 <artifactId>javax.servlet-api</artifactId>
36 <version>4.0.1</version>
37 <scope>provided</scope>
38 </dependency>
```

```
40 <!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
41 <dependency>
42 <groupId>org.springframework</groupId>
43 <artifactId>spring-orm</artifactId>
44 <version>5.3.18</version>
45 </dependency>
46
47 <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
48 <dependency>
49 <groupId>org.hibernate</groupId>
50 <artifactId>hibernate-core</artifactId>
51 <version>5.6.12.Final</version>
52 </dependency>
53
54 <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
55 <dependency>
56 <groupId>mysql</groupId>
57 <artifactId>mysql-connector-java</artifactId>
58 <version>8.0.30</version>
59 </dependency>
```

Step 3 Make configuration files under config package

WebAppInitializer

WebConfig

AppConfig

```
5 public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
6
7     protected Class<?>[] getRootConfigClasses() {
8         return new Class[]{AppConfig.class};
9     }
10
11     protected Class<?>[] getServletConfigClasses() {
12         return new Class[]{WebConfig.class};
13     }
14
15     protected String[] getServletMappings() {
16         return new String[]{"/"};
17     }
18 }
```

```
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.ComponentScan;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.web.servlet.ViewResolver;
7 import org.springframework.web.servlet.config.annotation.EnableWebMvc;
8 import org.springframework.web.servlet.view.InternalResourceViewResolver;
9
```

1 usage

15 @Configuration

16 @EnableTransactionManagement

17 public class AppConfig {

y/", suffix: ".jsp");

18 }

```

25     @Bean
26     public DataSource getDataSource(){
27         DriverManagerDataSource ds = new DriverManagerDataSource();
28         ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
29         ds.setUrl("jdbc:mysql://localhost:3306/wave31db");
30         ds.setUsername("root");
31         ds.setPassword("password");
32         return ds;
33     }
34
35     @Bean
36     public LocalSessionFactoryBean getSessionFactory(DataSource dataSource){
37         LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
38         sessionFactory.setDataSource(dataSource);
39         sessionFactory.setPackagesToScan("org.example.domain");
40         Properties properties = new Properties();
41         properties.put("hibernate.dialect", "org.hibernate.dialect.MySQL8Dialect");
42         properties.put("hibernate.show_sql", "true");
43         properties.put("hibernate.hbm2ddl.auto", "update");
44         sessionFactory.setHibernateProperties(properties);
45         return sessionFactory;
46     }
47
48     @Bean
49     public HibernateTransactionManager getTransactionManager(SessionFactory sessionFactory){
50         HibernateTransactionManager transactionManager = new HibernateTransactionManager();
51         transactionManager.setSessionFactory(sessionFactory);
52         return transactionManager;
53     }
54 }

```

1 usage

```

15 @Configuration
16 @EnableTransactionManagement
17 public class AppConfig {
18

```

Step 4 Define model classes

Song class

```

6     @Entity
7     public class Song {
8         4 usages
9         @Id
10        private String songId;
11        4 usages
12        private String songName, artistName, duration;
13
14        public Song() {}
15
16        public Song(String songId, String songName, String artistName, String duration) {...}
17

```

Step 5 Define dao/repository layer

Define interface

```
7 public interface SongRepository {
8     public abstract boolean addSong(Song song);
9     public abstract List<Song> getAllSongs();
10 }
```

Define impl class

```
13 @Repository
14 public class SongRepositoryImpl implements SongRepository{
15     // inject sessionFactory
16     // 2 usages
17     @Autowired
18     private SessionFactory sessionFactory;
19
20     public boolean addSong(Song song) {
21         Session ses = sessionFactory.openSession();
22         Transaction tr = ses.beginTransaction();
23         ses.save(song);
24         tr.commit();
25         ses.close();
26         return true;
27     }
28
29     public List<Song> getAllSongs() {
30         Session ses = sessionFactory.openSession();
31         Query q = ses.createQuery("from Song");
32         List<Song> songs = q.list();
33         ses.close();
34         return songs;
35     }
36 }
```

Step 6 Define service layer

Define service interface

```
7 public interface SongService {
8     public abstract boolean addSong(Song song);
9     public abstract List<Song> getSongs();
10 }
```

Define service impl class

```
10  @Service
11  public class SongServiceImpl implements SongService {
12      2 usages
13      @Autowired
14      private SongRepository songRepository;
15
16      public boolean addSong(Song song) {
17          return songRepository.addSong(song);
18      }
19
20      public List<Song> getSongs() {
21          return songRepository.getAllSongs();
22      }
23  }
```

Step 7 Define controller

define request handler to get songs and attach to view

```
13  @Controller
14  public class SongController {
15
16      1 usage
17      @Autowired
18      private SongService songService;
19
20      @RequestMapping("/")
21      @GetMapping("/")
22      public String displaySongPage(Model m){
23          List<Song> songs = songService.getSongs();
24          m.addAttribute("songs", songs);
25          return "songpage";
26      }
27  }
```

Step 8 Create jsp in proper location, with proper name
remove existng index.jsp








```
1 <%@ page isELIgnored="false" %>
2 <html>
3     <head>
4         <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
5     </head>
6     <body>
7
8         <hr/>
9         <c:forEach items="${songs}" var="s">
10             ${s}
11             <br/>
12         </c:forEach>
13
14     </body>
15 </html>
```

Make sure table created in db

Insert few records manual in db table

refresh page, make sure records are visible in page

```
16 • insert into song (songId, songName, duration, artistName) values
17 ('S0001', 'abcd', '0:20:15', 'artist1'),
18 ('S0002', 'xtz', '1:12:11', 'artist3'),
19 ('S0003', 'mnop', '0:45:22', 'artist2');
20
21 • select * from song;
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:    				
Export/Import:  				
Wrap Cell Content: 				
songId	artistName	duration	songName	
S0001	artist1	0:20:15	abcd	
S0002	artist3	1:12:11	xtz	
S0003	artist2	0:45:22	mnop	
NULL	NULL	NULL	NULL	

localhost:8080/springMvcHibernate1/

```
Song {songId='S0001', songName='abcd', artistName='artist1', duration='0:20:15'}
Song {songId='S0002', songName='xtz', artistName='artist3', duration='1:12:11'}
Song {songId='S0003', songName='mnop', artistName='artist2', duration='0:45:22'}
```


Additional

add new song

Create form in page

```
8      <h1>Song Form</h1>
9      <form action="/reqSendNewSongData" method="POST">
10         Song ID <input type="text" name="sid"/>
11         <br/>Song name<input type="text" name="sn"/>
12         <br/>Artist name<input type="text" name="an"/>
13         <br/>Song duration<input type="text" name="sd"/>
14         <br/><input type="submit" value="Add Song"/>
15     </form>
```

Define request handler in controller

receive all param data

```
27     @PostMapping("/reqSendNewSongData")
28     public String addSong(@RequestParam("sid")String a, @RequestParam("sn")String b,
29                           @RequestParam("an")String c, @RequestParam("sd")String d){
30         Song song = new Song(a,b,c,d);
31         songService.addSong(song);
32         return "redirect:/";
33     }
```