This post presents a web project sample with a full java configuration using Spring MVC 4 and Servlet 3 api. The code source is available in this [link](http://project4example8.googlecode.com/svn/trunk/MvcProject/).

This project implements client side for the persistence sample presented in [Spring-Data-JPA article](https://samerabdelkafi.wordpress.com/2014/05/29/spring-data-jpa/)

**1 – Add project dependencies**imlmplements

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <dependency>      <groupId>org.springframework</groupId>      <artifactId>spring-webmvc</artifactId>      <version>4.0.1.RELEASE</version>  </dependency>  <dependency>      <groupId>javax.servlet</groupId>      <artifactId>servlet-api</artifactId>      <version>3.0.1</version>      <scope>provided</scope>  </dependency>  <dependency>      <groupId>org.codehaus.jackson</groupId>      <artifactId>jackson-mapper-asl</artifactId>      <version>1.9.13</version>  </dependency> |

**2 – Spring MVC config class**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | import org.springframework.context.annotation.\*;  import org.springframework.context.support.ReloadableResourceBundleMessageSource;  import org.springframework.web.multipart.commons.CommonsMultipartResolver;  import org.springframework.web.servlet.config.annotation.\*;  import org.springframework.web.servlet.view.InternalResourceViewResolver;    @EnableWebMvc  @Configuration  @ComponentScan(basePackages = { "com.mycompany.myproject.web.controller" })  public class MvcConfig extends WebMvcConfigurerAdapter {        @Override      public void addResourceHandlers(ResourceHandlerRegistry registry) {          registry.addResourceHandler("/resources/\*\*").addResourceLocations("/resources/");      }        @Override      public void configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {          configurer.enable();      }        @Bean      public InternalResourceViewResolver jspViewResolver() {          InternalResourceViewResolver bean = new InternalResourceViewResolver();          bean.setPrefix("/WEB-INF/views/");          bean.setSuffix(".jsp");          return bean;      }        @Bean(name = "multipartResolver")      public CommonsMultipartResolver getMultipartResolver() {          return new CommonsMultipartResolver();      }        @Bean(name = "messageSource")      public ReloadableResourceBundleMessageSource getMessageSource() {          ReloadableResourceBundleMessageSource resource = new ReloadableResourceBundleMessageSource();          resource.setBasename("classpath:messages");          resource.setDefaultEncoding("UTF-8");          return resource;      }    } |

**3 – Webapp initializer**

With servlet 3 API you can use a class to configure your web project instead of the classic web.xml file.  
You have to check if your application server or your servlet container supports servlet 3 spec. For Tomcat you can have this information in this link:   
[tomcat.apache.org/whichversion](http://tomcat.apache.org/whichversion.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | import javax.servlet.ServletContext;  import javax.servlet.ServletRegistration;    import org.springframework.web.WebApplicationInitializer;  import org.springframework.web.context.ContextLoaderListener;  import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;  import org.springframework.web.filter.DelegatingFilterProxy;  import org.springframework.web.servlet.DispatcherServlet;    public class WebAppInitializer implements WebApplicationInitializer {        @Override      public void onStartup(ServletContext container) {          // Create the 'root' Spring application context          AnnotationConfigWebApplicationContext rootContext = new AnnotationConfigWebApplicationContext();          rootContext.register(ServiceConfig.class, JPAConfig.class, SecurityConfig.class);            // Manage the lifecycle of the root application context          container.addListener(new ContextLoaderListener(rootContext));            // Create the dispatcher servlet's Spring application context          AnnotationConfigWebApplicationContext dispatcherServlet = new AnnotationConfigWebApplicationContext();          dispatcherServlet.register(MvcConfig.class);            // Register and map the dispatcher servlet          ServletRegistration.Dynamic dispatcher = container.addServlet("dispatcher", new DispatcherServlet(dispatcherServlet));          dispatcher.setLoadOnStartup(1);          dispatcher.addMapping("/");        }     } |

**4 – Model class.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public class User {        private Long id;        private String firstName;        private String familyName;        private String email;        // add extra attributes        // add getters and setters    } |

**5 – Add view as jsp page**  
We create a usersView.jsp page with code source bellow to display users list.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ taglib prefix="spring" uri="<http://www.springframework.org/tags>"%>  <%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c"%>    <div class="col-sm-offset-1 col-sm-10">        <legend>          <spring:message code="table.user.title" />      </legend>        <div>          <table id="dataTable" class="table table-striped table-bordered">              <thead>                  <tr>                      <th><spring:message code="table.user.id" /></th>                      <th><spring:message code="table.user.firstName" /></th>                      <th><spring:message code="table.user.falilyName" /></th>                      <th><spring:message code="table.user.email" /></th>                  </tr>              <thead>              <tbody>                  <c:forEach var="u" items="${usersModel}">                      <tr>                          <td>${u.id}</td>                          <td>${u.firstName}</td>                          <td>${u.familyName}</td>                          <td>${u.email}</td>                      <tr>                  </c:forEach>              </tbody>          </table>      </div>  </div> |

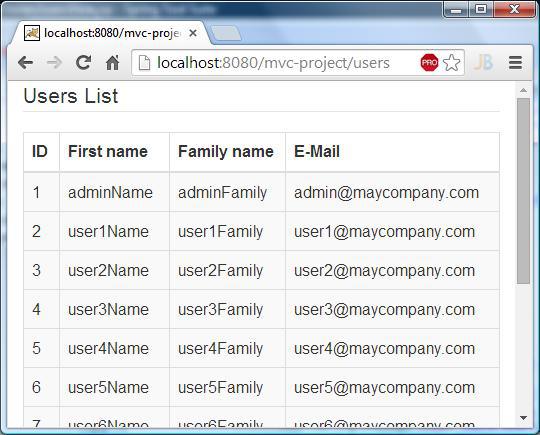
**6 – Develop your controller**

The @Controller annotation indicates that a particular class serves the role of a controller.  
The @RequestMapping annotation is used to map a URL to either an entire class or a particular handler method.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.context.MessageSource;  import org.springframework.context.annotation.Scope;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.\*;    @Controller  public class UserController {        @Autowired      private UserService userService;        /\*\*      \* Request mapping for user      \*/      @RequestMapping(value = "users", method = RequestMethod.GET)      public ModelAndView getUsersView() {          ModelAndView mv= new ModelAndView("usersView");          mv.addObject("usersModel", userService.findAll());          return mv;      }        /\*\*      \* Rest web service      \*/      @RequestMapping(value = "/usersList", method = RequestMethod.GET)      public @ResponseBody List<UserDto> getUsersRest() {          return userService.findAll();      }  } |

**7 – Test web project**  
When building your project you may have maven error telling you that the web.xml file is missing.  
You can fix it by configuring the maven war plugin as presented bellow.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <plugin>      <groupId>org.apache.maven.plugins</groupId>      <artifactId>maven-war-plugin</artifactId>      <configuration>          <failOnMissingWebXml>false</failOnMissingWebXml>      </configuration>  </plugin> |

* Test display users  
  <http://localhost:8080/myproject/users>  
  
* Test rest web service  
  <http://localhost:8080/myproject/usersList>  
  You will get json reponse.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | [      {          "id": 1,          "firstName": "adminName",          "familyName": "adminFamily",          "email": "admin@maycompany.com",          "phone": "0033 1 23 45 67 89",          "language": "en",          "login": "admin",          "password": "admin",          "burthDate": null,          "authorities": [              {"id": 3, "name": "user"},              {"id": 2, "name": "technical user"},              {"id": 1, "name": "admin"}          ],          "enabled": true,          "pictureId": null,          "authoritiesAsString": "user, technical user, admin"      },      ..........  ] |