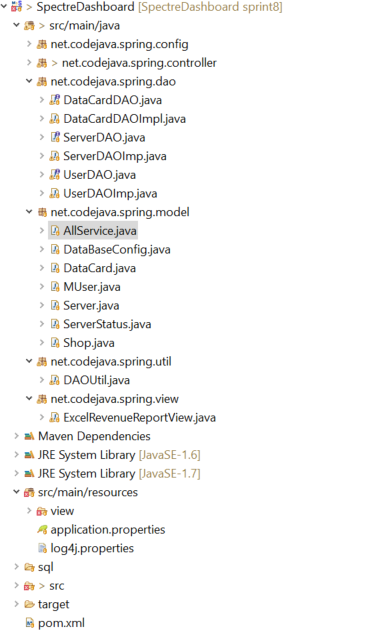
Maven Project + External project integration + Global login integration + Spring MVC with Annotation + MySQL (JDBC Template).



<properties>

<java.version>1.7</java.version>

<spring.version>4.0.3.RELEASE</spring.version>

<cglib.version>2.2.2</cglib.version>

<log4j.version>1.2.17</log4j.version>

<jackson.version>1.9.10</jackson.version>

<tiles.version>3.0.7</tiles.version>

</properties>

<dependencies>

<!-- Spring core & mvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

<type>jar</type>

<scope>compile</scope>

</dependency>

<!-- CGLib for @Configuration -->

<dependency>

<groupId>cglib</groupId>

<artifactId>cglib-nodep</artifactId>

<version>${cglib.version}</version>

<scope>runtime</scope>

</dependency>

<!-- Servlet Spec -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>javax.servlet.jsp-api</artifactId>

<version>2.3.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.9</version>

</dependency>

<dependency>

<groupId>com.att</groupId>

<artifactId>Spectre2-DatabaseAPI</artifactId>

<version>1.0.8</version>

</dependency>

Application Properties:

database.ip=135.89.52.223

database.pass=Read1!only

database.username=spectre

database.port=3306

ExternalURL.engineeringGUI=http://m5vas001.gcsc.att.com:8080/spectre2/

ExternalURL.cannedreport=http://m5vas001.gcsc.att.com:8080/spectreportal/

# Adding the external project (How to include custom library into maven local repository?)

$ mvn install:install-file -Dfile=c:\Spectre2-DatabaseAPI.jar -DgroupId= com.att

-DartifactId= Spectre2-DatabaseAPI -Dversion=1.0.8 -Dpackaging=jar

**Application Configuration:**

@Configuration

@ComponentScan(basePackages="net.codejava.spring")

/\*@EnableWebMvc\*/

@PropertySource("classpath:application.properties")

**public** **class** MvcConfiguration **extends** WebMvcConfigurerAdapter{

@Autowired

**private** Environment env;

@Bean

**public** ViewResolver getViewResolver(){

/\*InternalResourceViewResolver resolver = new InternalResourceViewResolver();

resolver.setPrefix("/WEB-INF/views/");

resolver.setSuffix(".jsp");

return resolver;\*/

UrlBasedViewResolver viewResolver=**new** UrlBasedViewResolver();

viewResolver.setViewClass(TilesView.**class**);

**return** viewResolver;

}

@Bean

**public** TilesConfigurer tilesConfigurer(){

TilesConfigurer tilesConfigurer=**new** TilesConfigurer();

tilesConfigurer.setDefinitions(**new** String[]{"/WEB-INF/tiles.xml"});

tilesConfigurer.setCheckRefresh(**true**);

**return** tilesConfigurer;

}

@Bean

**public** XmlViewResolver xmlViewResolver() {

XmlViewResolver resolver = **new** XmlViewResolver();

Resource resource = **new** ClassPathResource("view/views.xml");

resolver.setLocation(resource);

resolver.setOrder(0);

**return** resolver;

}

@Override

**public** **void** addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/resources/\*\*").addResourceLocations("/resources/");

}

@Bean

**public** DataSource getDataSource() {

DriverManagerDataSource dataSource = **new** DriverManagerDataSource();

dataSource.setDriverClassName("com.mysql.jdbc.Driver");

dataSource.setUrl("jdbc:mysql://" +env.getProperty("database.ip") + ":" + env.getProperty("database.port") +"/spectre2");

dataSource.setUsername(env.getProperty("database.username"));

dataSource.setPassword(env.getProperty("database.pass"));

**return** dataSource;

}

@Bean

**public** DataCardDAO getDataCardDAO() {

**return** **new** DataCardDAOImpl(getDataSource(),getDBConfig() );

}

@Bean

**public** UserDAO getUserDAO() {

**return** **new** UserDAOImp(getDataSource(),getDBConfig() );

}

@Bean

**public** ServerDAO getServerDAO(){

**return** **new** ServerDAOImp(getDBConfig());

}

@Bean

**public** DataBaseConfig getDBConfig(){

DataBaseConfig db = **new** DataBaseConfig();

String username =env.getProperty("database.username");

String pass=env.getProperty("database.pass");

String ip = env.getProperty("database.ip");

String port =env.getProperty("database.port");

db.setIp(ip);

db.setPass(pass);

db.setUsername(username);

db.setPort(port);

**return** db;

}

}

**Global Login with cookies and session handling**

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletContext;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

@Controller

@Configuration

@ComponentScan(basePackages = "net.codejava.spring.controller")

@EnableWebMvc

**public** **class** LoginController **extends** HttpServlet **implements** Serializable {

@Autowired

ServletContext context;

// This URL encoding step only becomes necessary if you need parameters

// passed to your application as

// illustrated here with "fakeparameter". Without parameters you could skip

// this step.

**static** String *encReturnURL* = URLEncoder.~~encode~~("http://m5vas001.gcsc.att.com:8080/SpectreDashboard/");

// CSP "PROD" environment

// Do not mix environments for the noCookieURL and tooWeakURL.

// Use both from (webtest) below or both from (Production) above.

**static** String *noCookieURL* = "https://www.e-access.att.com/empsvcs/hrpinmgt/pagLogin/?retURL=" + *encReturnURL* + "&sysName=CEPPlus ";

**static** String *tooWeakURL* = "https://www.e-access.att.com/empsvcs/hr/pagMenu\_chgpin/?opt=12";

// CSP "DEVL" environment

// static String noCookieURL = "https://webtest.csp.att.com/empsvcs/hrpinmgt/pagLogin/?retURL=" +encReturnURL;

// static String tooWeakURL = "https://webtest.csp.att.com/empsvcs/hr/pagMenu\_chgpin/?opt=12";

**final** **int** TOKENSINSECCOOKIE = 8;

**final** **int** TOKENSINHRCOOKIEEMP = 12;

**final** **int** TOKENSINHRCOOKIENPW = 11;

/\*\*

\* Gets executed EACH TIME LoginController is called. Simply calls performTask()

\*/

@RequestMapping(value = "/", method = RequestMethod.***GET***)

**public** **void** service(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// Originally, the check for the "user" Object, which indicated a logged

// in

// user was located here. This was useful when LDAP (IBM Intranet

// UID/PW) was

// used, but not if LDAP logins are disabled (via web.xml)

// TD Bank Modification:

// There used to be a check here for the user Object, but as it is not

// used when

// Intranet Login is disabled, that check is now in the performTask

// method

String strMsg = "";

**boolean** mErr = **false**;

RequestDispatcher dispatcher = **null**;

HttpSession session = request.getSession(**true**);

// ATT Global Logon changes, for the local environment uncomment attESSec and attESHr

//String attESSec = "006cm0ceqvOTsr6rZ42Sa6gHV2DxGwZda1LpTmz6AM6gJteBx1Iqb1aPdNkGxtqjzaLA5RKXeyd5gvoXWeyNxANVy2rVZcEhUyuQkFxUXwJzJnU.iKzdJ8bU9TkM\_g6aU1MDjkpxbtlnFlfOCEJj55kqXA8N48f9Qq3dcxYXD7tNvjhHoEvygb\_JYXKL\_Ms2LHHIakSOd5wSy\_09UZt\_sPyzHpGTEJR8lX7haGsV77qUEG9Y";

//String attESHr = "SACHIN%7cCHIPPALAKATTI%7csc849m%40intl%2eatt%2ecom%7c918067605067%7c%7crn220r%7c%7csc849m%2cRHZQMDP%2cHTS3H35%2c2322386%7cYNNNNNNNNNNNNNYNYYNNNNNN%7cSACHIN%7cAT1P5G200%7c";

// create an empty 'attESSec' string and create an empty 'attESHr' string for dev and production

String attESSec = **null**;

String attESHr = **null**;

// get all cookies

Cookie[] cookies = request.getCookies();

**if** ((cookies == **null**) || (cookies.length == 0)) {

response.sendRedirect(*noCookieURL*);

**return**;

}

**if** (request.getRequestedSessionId() == **null** && request.isRequestedSessionIdValid()) {

**for** (**int** i = 0; i < cookies.length; i++) {

cookies[i].setMaxAge(0);

}

response.sendRedirect(*noCookieURL*);

**return**;

}

// look for the 'attESSec' cookie

**for** (**int** i = 0; i < cookies.length; i++) {

Cookie thisCookie = cookies[i];

String cookieName = thisCookie.getName();

**if** (cookieName == **null**) {

mErr = **true**;

strMsg = "BasLogin: cookieName is null";

// out.println("GateServlet: cookieName is null");

response.sendRedirect(*noCookieURL*);

**return**;

}

**if** (cookieName.equals("attESSec")) {

attESSec = thisCookie.getValue();

**break**;

}

}

// if we can't find the 'attESSec' cookie, send to the browser to

// 'noCookieURL' and return

**if** ((attESSec == **null**) || (attESSec.length() == 0)) {

response.sendRedirect(*noCookieURL*);

**return**;

}

// out.println( "attESSec=" + attESSec ); //Debugging

// look for the 'attESHr' cookie

**for** (**int** i = 0; i < cookies.length; i++) {

Cookie thisCookie = cookies[i];

String cookieName = thisCookie.getName();

**if** (cookieName == **null**) {

mErr = **true**;

strMsg = "BasLogin: cookieName is null";

response.sendRedirect(*noCookieURL*);

**return**;

// out.println("GateServlet: cookieName is null");

}

**if** (cookieName.equals("attESHr")) {

attESHr = thisCookie.getValue();

**break**;

}

}

// if we can't find the 'attESHr' cookie, send to the browser to

// 'noCookieURL' and return

**if** ((attESHr == **null**) || (attESHr.length() == 0)) {

response.sendRedirect(*noCookieURL*);

**return**;

}

**if** (!mErr) {

performGloballogon(request, response, attESHr);

} **else** {

request.setAttribute("dispatchTarget", "presentation.jsp");

dispatcher = context

.getRequestDispatcher("/presentation?PID=bas/bas\_error.jsp&ERROR=" + strMsg);

dispatcher.forward(request, response);

}

} // end of service

**public** **void** performGloballogon(HttpServletRequest request, HttpServletResponse response,

String attESHr) **throws** ServletException, IOException {

**try** {

RequestDispatcher dispatcher = **null**;

HttpSession session = request.getSession(**true**);

**boolean** mErr = **false**;

String HRFirstName = "", HRLastName = "", HREMAIL = "", HRPhone = "", HRMIddleName = "", HRWorkRoom = "",

HRNameSuffix = "", HRMgrID = "", HRAttID = "";

// Tokenize the attESHr string

String attEMP[] = attESHr.split("%7c");

StringTokenizer tokens = **new** StringTokenizer(attESHr, "%7c");

**if** ((tokens.countTokens() != TOKENSINHRCOOKIEEMP) && (tokens.countTokens() != TOKENSINHRCOOKIENPW)) {

// out.println("GateServlet: Incorrect parsing of attESHr

// cookie");

}

**if** (attEMP.length >= 6) {

HRFirstName = attEMP[0];

HRLastName = attEMP[1];

HREMAIL = attEMP[2];

HREMAIL = HREMAIL.replaceAll("%40", "@").replaceAll("%2e", ".");

System.***out***.println(HREMAIL);

HRPhone = attEMP[3];

HRMIddleName = attEMP[4];

// HRWorkRoom= tokens.nextToken();

String HRAttIDarray[] = attEMP[7].replaceAll("%2", ",").split(",");

HRAttID = HRAttIDarray[0];

HRNameSuffix = attEMP[6];

HRMgrID = attEMP[5];

}

//Session handling

session.removeAttribute("userProfile");

session = request.getSession(**true**);

session.setAttribute("DSN", context.getInitParameter("DATASOURCE")); // enoscr1d

session.setAttribute("EMPID", HRAttID);

session.setAttribute("FIRSTNAME", HRFirstName);

session.setAttribute("LASTNAME", HRLastName);

**if** ((session.getAttribute("USERID") != **null**) && (!session.getAttribute("USERID").toString().equals(""))

&& (!session.getAttribute("USERID").toString().equalsIgnoreCase(HREMAIL))) {

dispatcher = context.getRequestDispatcher("/error?ERROR=User Authentication Error. User ID Object was blank or null.");

dispatcher.forward(request, response);

} **else** {

session.setAttribute("USERID", HREMAIL);

}

**if** (!mErr) {

**if** (session.getAttribute("EMPID") != **null**) {

dispatcher = context.getRequestDispatcher("/start?attid=" + HRAttID);

}

}

dispatcher.forward(request, response);

} **catch** (Throwable theException) {

// handleError(response, theException);

theException.printStackTrace();

}

}

}

**Controller:**

@Controller

@Configuration

@ComponentScan(basePackages="net.codejava.spring.controller")

@RequestMapping("server")

@PropertySource("classpath:application.properties")

**public** **class** ServerController {

@Autowired

**private** ServerDAO serverdao;

@Autowired

**private** DataCardDAO datacardDAO;

@Autowired

**private** UserDAO userDAO;

@Autowired

**private** Environment env;

**private** **static** **final** Logger ***logger*** = Logger.*getLogger*(ServerController.**class**);

@RequestMapping(value="/serverstatus", method = RequestMethod.***GET***)

**public** ModelAndView getServerStatus(ModelAndView model,HttpServletRequest req) **throws** IOException{

***logger***.info("Entring in to ServerController::getServerStatus");

String customerid = req.getParameter("custid");

Map<String, Integer> serverstatus = serverdao.getServerStatus(customerid);

model.addObject("fmserver", serverstatus);

model.setViewName("datacard");

***logger***.info("Exting from the ServerController::getServerStatus with the values :" + serverstatus );

**return** model;

}

}

DAO:

**public** **interface** ServerDAO {

**public** Map<String, Integer> getServerStatus(String customerid);

**public** List<Spectre2NewReportServerFM> getFaultMangamentServer(String customerid);

**public** List<Spectre2NewReportServerCM> getConfigurationManagementServer(String customerid);

**public** List<Spectre2NewReportServerPM> getPerformanceManagement(String customerid);

**public** List<Spectre2NewReportServerSM> getSecurityManagement(String customerid);

**public** AllService getAllService(String customerid);

}

Implementation:

**public** **class** DataCardDAOImpl **implements** DataCardDAO{

**private** JdbcTemplate jdbcTemplate;

**private** DataBaseConfig db;

**public** Spectre2DatabaseConnectionhandler sdbc;

**private** **static** **final** Logger ***logger*** = Logger.*getLogger*(DataCardDAOImpl.**class**);

**public** DataCardDAOImpl(DataSource dataSource, DataBaseConfig dataBaseConfig) {

jdbcTemplate = **new** JdbcTemplate(dataSource);

db = dataBaseConfig;

**try** {

sdbc = **new** Spectre2DatabaseConnectionhandler(db.getUsername(), db.getPass(), db.getIp(), db.getPort());

sdbc.*setDebug*(**true**);

}**catch** (SQLException e) {

// **TODO** Auto-generated catch block

***logger***.error("Error occured : " + e.getMessage());

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

***logger***.error("Error occured : " + e.getMessage());

} **catch** (InstantiationException e) {

// **TODO** Auto-generated catch block

***logger***.error("Error occured : " + e.getMessage());

} **catch** (IllegalAccessException e) {

// **TODO** Auto-generated catch block

***logger***.error("Error occured : " + e.getMessage());

}

}

}