SpringWebAppInitializer

/\*\*

\* This is main method for web application

\* This is configuration for front controller

\*

\* this is equivalent to web.xml ->>>SpringWebAppInitializer

\*/

**public** **class** SpringWebAppInitializer **extends** AbstractAnnotationConfigDispatcherServletInitializer {

/\*\*

\* This is responsible for service layer , dao layer ,jdbctemplate and datasource

\*/

@Override

**protected** Class<?>[] getRootConfigClasses() {

//this is integration between controller and service layer

//this files are used for spring root container where your service layer , dao layer etc bean will present

**return** **new** Class[] { DBConfig.**class**,ServiceDaoConfig.**class**};

}

@Override

**protected** Class<?>[] getServletConfigClasses() {

//ApplicationContextConfig.class is used in spring mvc web configuration

**return** **new** Class[] { ApplicationContextConfig.**class**};

}

@Override

**protected** String[] getServletMappings() {

**return** **new** String[] { "/" };

}

}

**DBConfig**

@Configuration

@EnableTransactionManagement // @Transactional ->>>>>> transactionManager

@PropertySource("classpath:/dbconfig.properties")

**public** **class** DBConfig {

@Autowired

**private** Environment env;

@Bean

**public** DriverManagerDataSource getDriverManageDataSource() {

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

dataSource.setDriverClassName(env.getProperty("db.driver"));

dataSource.setUrl(env.getProperty("db.url"));

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

dataSource.setPassword(env.getProperty("db.password"));

System.***out***.println("datasource created");

**return** dataSource;

}

@Bean

**public** LocalSessionFactoryBean getSessionFactory() {

LocalSessionFactoryBean factoryBean = **new** LocalSessionFactoryBean();

factoryBean.setDataSource(getDriverManageDataSource());

Properties props=**new** Properties();

props.put("hibernate.show\_sql", env.getProperty("hibernate.show\_sql"));

props.put("hibernate.hbm2ddl.auto", env.getProperty("hibernate.hbm2ddl.auto"));

factoryBean.setHibernateProperties(props);

factoryBean.setAnnotatedClasses(StudentEntity.**class**);

System.***out***.println("session factory created");

**return** factoryBean;

}

@Bean(name = "transactionManager")

**public** HibernateTransactionManager getTransactionManager() {

HibernateTransactionManager transactionManager = **new** HibernateTransactionManager();

transactionManager.setSessionFactory(getSessionFactory().getObject());

System.***out***.println("transactin manager created");

**return** transactionManager;

}

}

**ServiceDaoConfig**

@Configuration

@ComponentScan("com.synergisticit.dao")

@ComponentScan("com.synergisticit.service")

**public** **class** ServiceDaoConfig {

}

**ApplicationContextConfig**

@Configuration

@EnableWebMvc // this will enable spring latest features

@ComponentScan("com.synergisticit.web.controller")

**public** **class** ApplicationContextConfig **implements** WebMvcConfigurer {

ApplicationContextConfig(){

System.***out***.println("constructor from ApplicationContextConfig ");

}

@Override

**public** **void** configureViewResolvers(ViewResolverRegistry registry) {

registry.jsp("/", ".jsp");

System.***out***.println("configuring view resolver from ApplicationContextConfig ");

}

@Override

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

System.***out***.println(" addResourceHandlers() from ApplicationContextConfig");

// Register resource handler for images,by default img,css,js etc are blocked

registry.addResourceHandler("/img/\*\*").addResourceLocations("/img/")

.setCacheControl(CacheControl.*maxAge*(2, TimeUnit.***HOURS***).cachePublic());

registry.addResourceHandler("/css/\*\*").addResourceLocations("/css/")

.setCacheControl(CacheControl.*maxAge*(2, TimeUnit.***HOURS***).cachePublic());

registry.addResourceHandler("/js/\*\*").addResourceLocations("/js/")

.setCacheControl(CacheControl.*maxAge*(2, TimeUnit.***HOURS***).cachePublic());

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

}

}

**StudentDataServiceImpl**

@Service("studentDataService")

**public** **class** StudentDataServiceImpl **implements** IStudentDataService {

StudentDataServiceImpl(){

System.***out***.println("creating StudentDataServiceImpl const");

}

// dao layer we are auto wiring in service

@Autowired

**private** IStudentDataDao studentDataDao;

**public** IStudentDataDao getStudentDataDao() {

**return** studentDataDao;

}

**public** **void** setStudentDataDao(IStudentDataDao studentDataDao) {

**this**.studentDataDao = studentDataDao;

}

@Override

**public** List<Student> findStudents() {

// **TODO** Auto-generated method stub

List<Student> students = **new** ArrayList<>();

List<StudentEntity> entityList = studentDataDao.findStudents();

// /List<StudentEntity> ...we have to convert into List<Student>

**for** (StudentEntity entity : entityList) {

Student student = **new** Student();

BeanUtils.*copyProperties*(entity, student);

students.add(student);

}

**return** students;

}

}

**StudentDaoHibernateImpl**

@Repository

@Transactional

**public** **class** StudentDaoHibernateImpl **implements** IStudentDataDao {

@Autowired

**private** SessionFactory sessionFactory;

**private** Session getSession() {

**return** sessionFactory.getCurrentSession();

}

@Override

**public** List<StudentEntity> findStudents() {

TypedQuery<StudentEntity> query = getSession().createQuery("from StudentEntity"); // HQL

**return** query.getResultList();

}

}

****

#This is comment

db.username=root

db.password=root

db.driver=com.mysql.jdbc.Driver

db.url=jdbc:mysql://localhost:3306/asha\_db

# Hibernate properties

hibernate.show\_sql=true

hibernate.hbm2ddl.auto=update



person.name=Nagendra Kumar

person.education=M. Tech

email=nagen@synergisticit.com

**public** **class** Student {

**private** String name;

**private** String email;

**private** String mobile;

**private** **int** age;

**private** String gender;

**private** String college;

**private** MultipartFile bphoto;

@JsonIgnore

**private** **byte**[] photo;

//@JsonIgnore

**private** String encodeImage;

**public** String getEncodeImage() {

**return** encodeImage;

}

**public** **void** setEncodeImage(String encodeImage) {

**this**.encodeImage = encodeImage;

}

……………..

@Entity

@Table(name = "students\_tbl")

**public** **class** StudentEntity {

**private** String name;

**private** String email;

**private** String mobile;

**private** **int** age;

**private** String gender;

**private** String college;

**public** StudentEntity() {

}