

Başlangıç seviyesinde Maven, JPA ve Spring Dökümanı

Hazırlayan: Ufuk KARAAĞAÇ

Tarih: 12.05.2012

Versiyon: 1.1

blog: www.ufukkaraagac.com

İçindekiler

Giriş.....	3
1- Maven Projesi Nedir?	3
2- JPA ve Hibernate Nedir?	4
3- Spring Nedir?	5
4- Adım Adım Basit Bir Projenin Geliştirilmesi.....	6

Giriş

Bu döküman başlangıç seviyesinde aşağıdaki bilgileri içermektedir.

- Maven 'i kullanarak yeni bir proje oluşturma
- JPA'nın yapısı, kullanım amacı ve Hibernate ile arasındaki farklar
- Spring 'in yapısını ve kullanım amacı

Dökümanda giriş seviyesinde bilgiler verildikten sonra Maven, JPA ve Spring kullanarak basit bir örneğin adım adım nasıl geliştirileceği, dökümanın ilerleyen sayfalarında anlatılmaktadır. Örneği uygulamaya başlanmadan önce döküman içerisinde verilen kaynaklardan ilgili konuya ait belgelerin incelenmesi örneğin anlaşılması açısından yararlı olacaktır.

1- Maven Projesi Nedir?

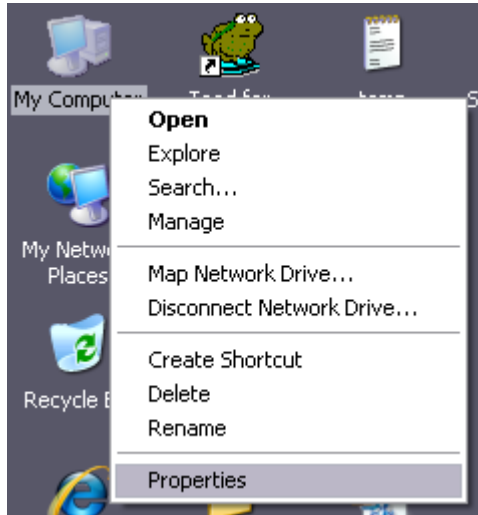
Maven, Jakarta Turbine projesinin geliştirilme sürecinde, proje aşamalarını basitleştirmek için kullanılmıştır. Apache vakfı tarafından geliştirilmiştir ve geliştirilmeye devam etmektedir. Maven ile standart bir yol ile projeler geliştirilebilir. Geliştirilen projelerin yönetilmesi ve anlaşılması kolaydır. Maven'in proje geliştirme sürecindeki faydalarına daha detaylı bakmak için <http://maven.apache.org/benefits-of-using-maven.html> adresini kullanabilirsiniz.

Maven projesinin resmi sitesi; <http://maven.apache.org/>

Türkçe kaynak için; <http://belgeler.cs.hacettepe.edu.tr/yayinlar/eski/Maven2.pdf>

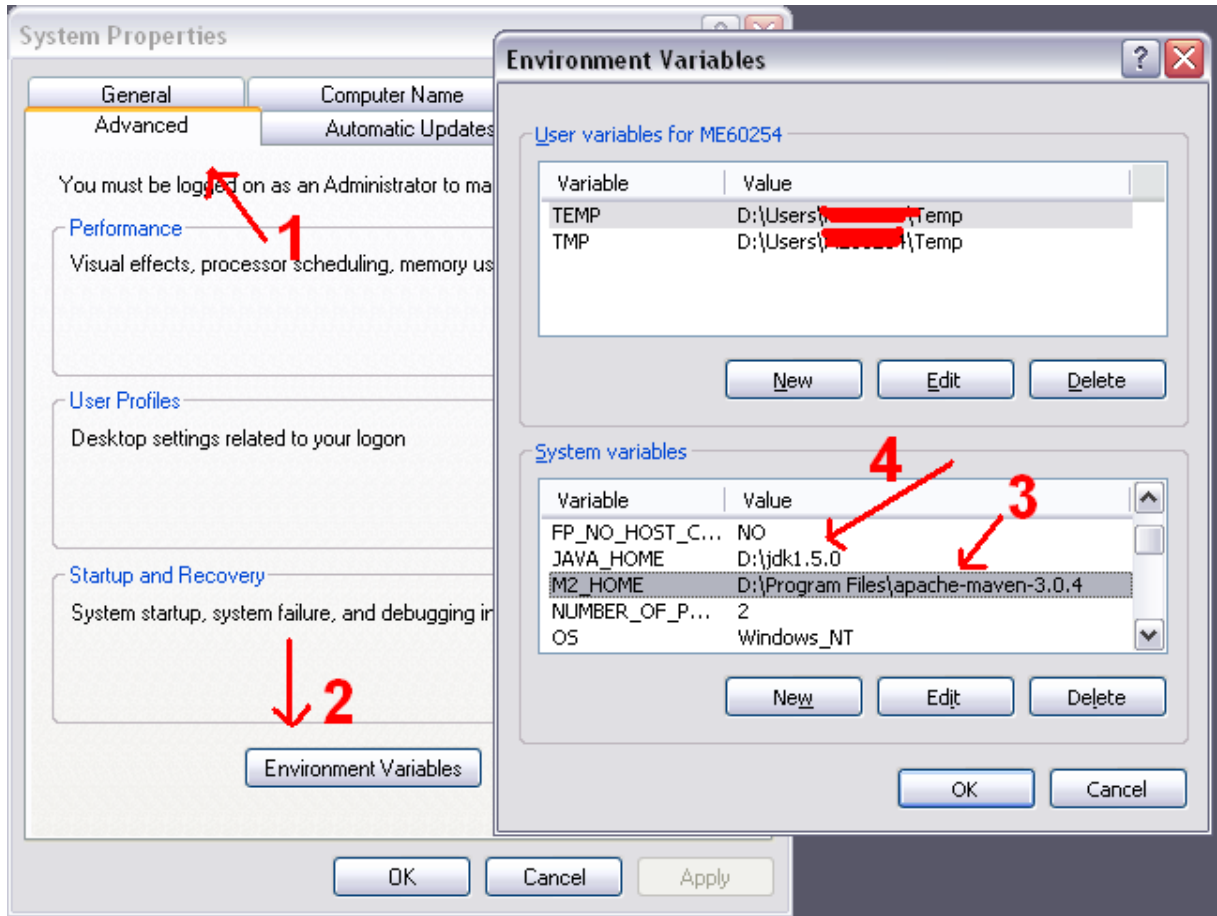
Maven projesini indirmek için; <http://godel.cs.bilgi.edu.tr/apache/maven/binaries/apache-maven-3.0.4-bin.zip> adresi kullanılabilir.

Proje indirildikten sonra; Şekil 1'deki gibi My Computer → Properties seçeneği seçilmelidir.



Şekil 1

Daha sonra Şekil 2 ‘deki adımlar takip edilmelidir.



Şekil 2

JAVA_HOME değişkeni Java SDK’ nın kurulduğu yeri, M2_HOME değişkeni Maven projesinin kurulu olduğu yeri göstermektedir. Yine aynı sayfada PATH değişkeninin sonuna Maven projesinin “bin” klasörünün tam adresi kopyalanmalıdır.

Örneğin; (path değişkenleri);D:\Program Files\apache-maven-3.0.4\bin

Not: Path değişkeninin sonuna kayıt eklenmeden önce noktalı virgül unutulmamalıdır.

2- JPA ve Hibernate Nedir?

JPA (Java Persistence API) veritabanından bağımsız kodlama yapabilmenizi ve sorgular hazırlayabilmenizi olanaklı kılar. Java geliştiricilerine veritabanı tabloları ile sınıflar arasında mapping imkanı sunarak, ilişkisel veritabanlarının nesne tabanlı bir geliştirme iskeleti ile kolaylıkla yönetilmesini sağlar. The Java Persistence API, Java Persistence Query Language ve nesne/tablo ilişkisini gösteren mapping metadataları gibi üç kola ayrılmış bir geliştirim mantığı vardır. JPQL (Java Persistence Query Language) ile yazacağınız sorgular, veritabanınızdaki ilişkisel yapıya göre hazırladığınız, birbiriyle ilişkili POJO(plain old java object) nesnelerini kullandığınız sorgular olacaktır. JPA 'nın sunduğu imkanlar ile çoğu basit işlemi (kayıt silme, ekleme ve birleştirme) de, JPQL kullanmadan, kolaylıkla hallededilme olanağı bulunmaktadır.

Hibernate, JPA' nın özelliklerini taşımakla birlikte kendine ait yararlı ek özellikleri de bulunmaktadır. JPA bir standart olduğu için bu döküman JPA 'ya ağırlık verilmektedir..

Türkçe kaynak;

1 - <http://www.kodcu.com/2011/10/java-persistence-giris/> ve pdf döküman

http://www.google.com.tr/url?sa=t&rct=j&q=jpa%20nedir&source=web&cd=2&ved=0CF4QFjAB&url=http%3A%2F%2Fbelgeler.cs.hacettepe.edu.tr%2Fyayinlar%2Feski%2FJPA.pdf&ei=i_6mT4a5BYbb4QSY38zGCQ&usg=AFQjCNH4JhfaRCmvC6WfoGZ6gB9_a9j2pQ&cad=rja

İngilizce Kaynak;

<http://docs.oracle.com/javaee/5/tutorial/doc/bnbpz.html>

3- Spring Nedir?

Spring bir framework' tür. Aspect Oriented Programming metodolojisini kullanarak web tabanlı yazılımlar geliştirilmesini sağlar. Bu işlevini dependency injection ve inversion of control mekanizmaları ile sağlar. Öğrenmesi zaman almakla beraber kullanıldıktan sonra gayet faydalı işlere imza atar. Ayrıca .net uygulamaları için de bir sürümü mevcuttur.

Türkçe kaynak;

http://belgeler.cs.hacettepe.edu.tr/yayinlar/eski/spring_ve_springaop.pdf

http://belgeler.cs.hacettepe.edu.tr/yayinlar/eski/Spring_grup3.pdf

http://belgeler.cs.hacettepe.edu.tr/yayinlar/eski/AK_SpringIoC_20221829.pdf

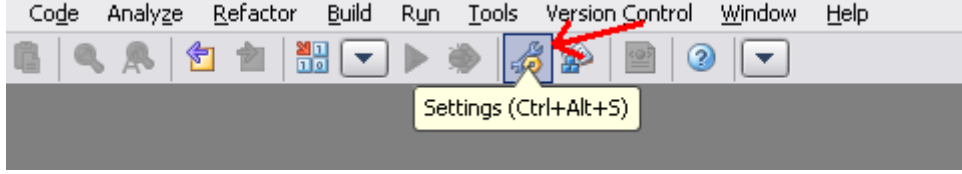
İngilizce kaynak;

<http://static.springsource.org/spring/docs/2.5.5/reference/beans.html>

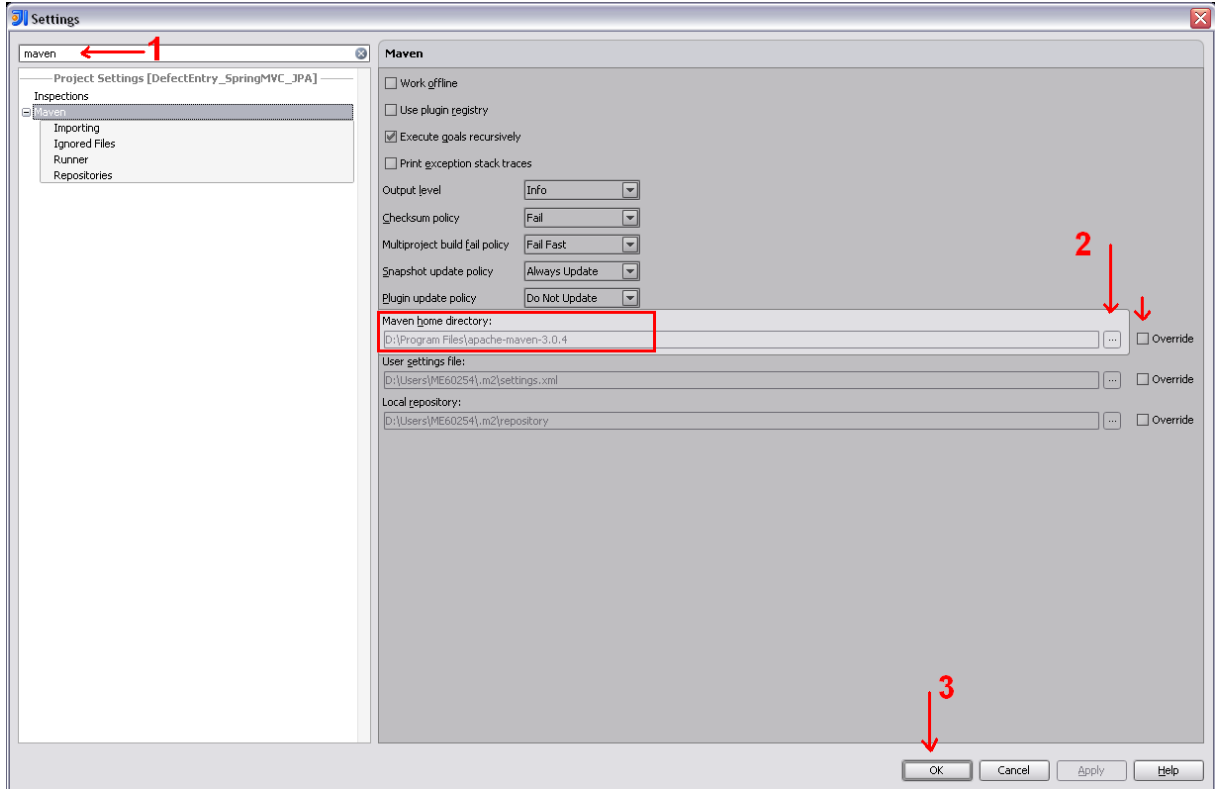
İlerleyen kısımda IntelliJ IDEA v.10.5 ile adım adım bir proje oluşturulmaktadır. Bu proje bir Maven projesi olacağı için farklı bir geliştirme ortamı da kullanılabilir. (Örneğin: Eclipse, Netbeanse vb.) Bu aşamada dikkat edilmesi gereken nokta Maven proje yapısına uygun şekilde dizinlerin doğru oluşturulmasıdır. Ayrıca pom.xml dosyasında projeye ait bilgilerin düzgün olarak girilmesi gerekir.

4- Adım Adım Basit Bir Projenin Geliştirilmesi

Herşeyden önce Maven uygulamasının IntelliJ IDEA da tanımlı olması gerekmektedir. Bunun için aşağıdaki gibi “Settings” alanına girilir.



Birinci adımdaki gibi arama kısmına “maven” yazıldıktan sonra eğer tanımlı bir “Maven home directory” yoksa ikinci adımdaki gibi tanımlanmalıdır. Eğer tanımlı ise bu adım atlanabilir.



Proxy Ayarları:

Bulunduğunuz ortamda bir proxy varsa aşağıdaki adımlar uygulanmalıdır.

- IntelliJ IDEA ‘da proxy ayarlarının hızlıca değiştirilmesi istenirse yukarıdaki gibi “maven” yerine “proxy“ yazılıp gerekli değişiklikler yapılabilir.

- Maven uygulamasının proxy ayarları değiştirilmek istenirse, maven uygulamasının bulunduğu dizinde (Örn: D:\Program Files\apache-maven-3.0.4\conf ") conf klasörü altında bulunan settings.xml içerisinde aşağıdaki satırlar eklenmelidir.

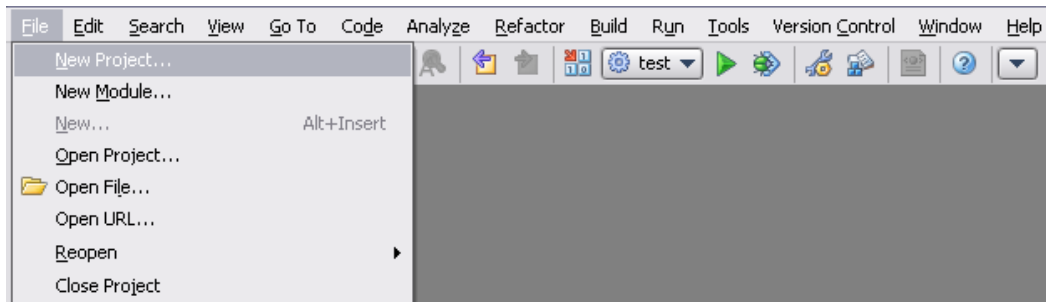
<proxies>

```
<proxy>
  <id>optional</id>
  <active>true</active>
  <protocol>http</protocol>
  <host>ip-adresi</host>
  <port>port</port>
  <nonProxyHosts>localhost</nonProxyHosts>
</proxy>
```

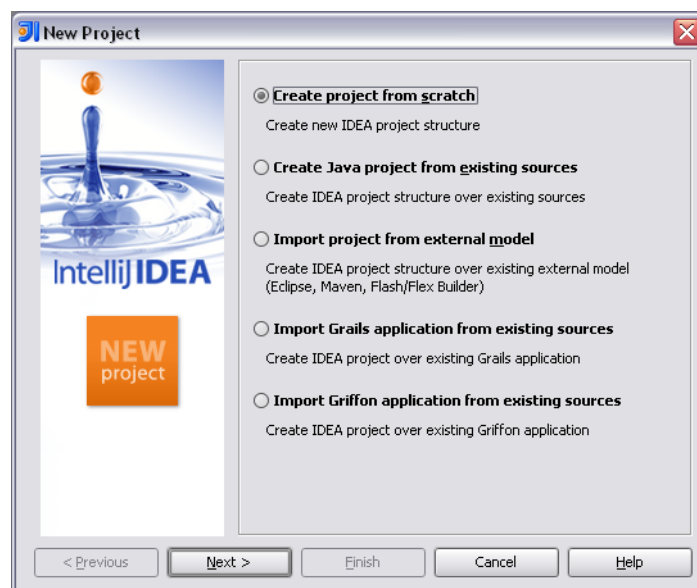
</proxies>

İşlem Adımları;

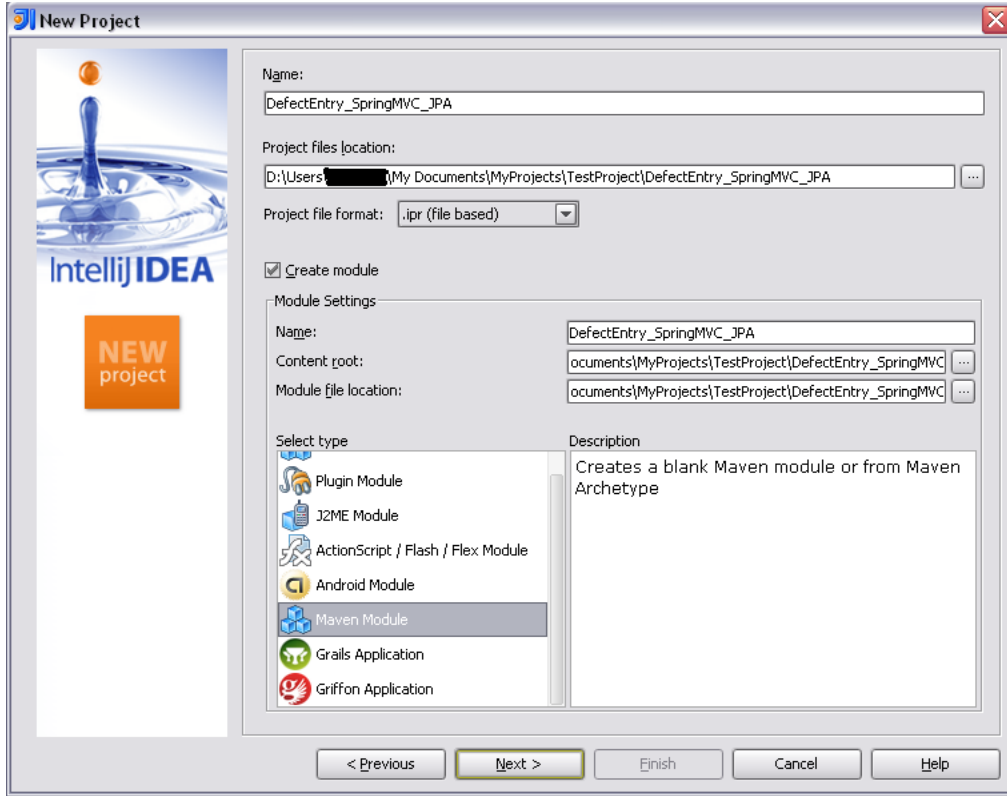
“File” menüsünden “New Project” seçeneği seçilir.



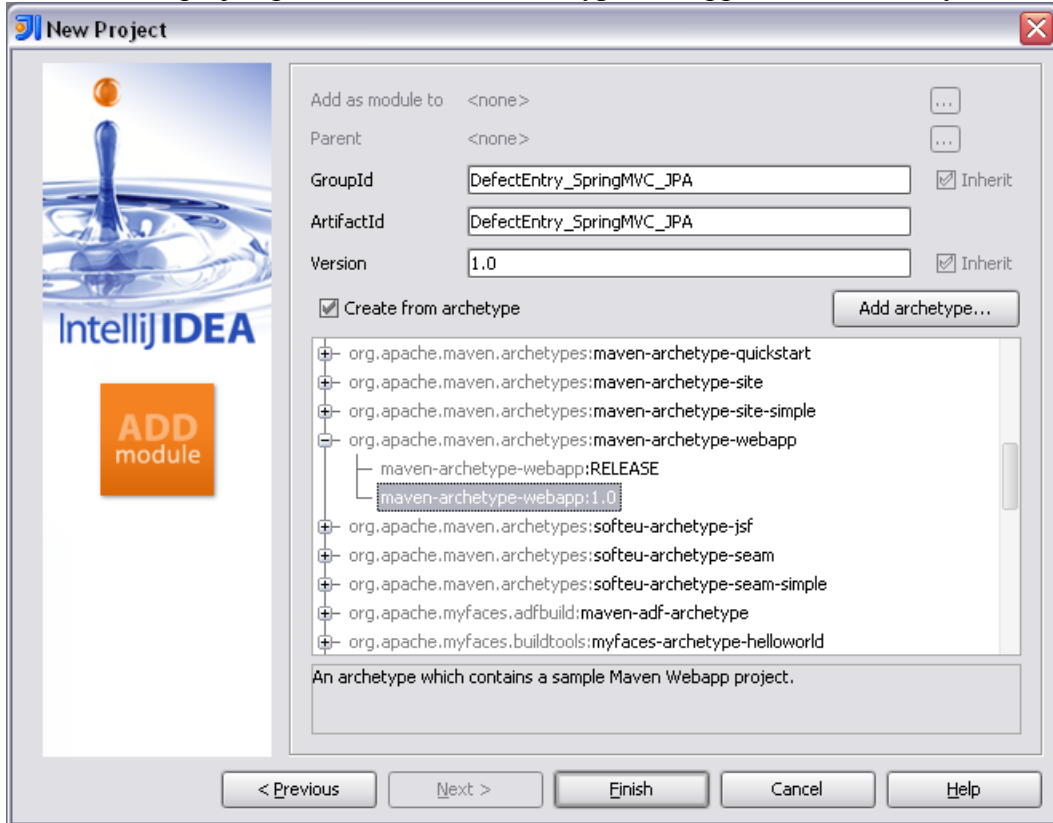
Daha sonraki ekranda aşağıda gösterilen seçenek uygulanır.



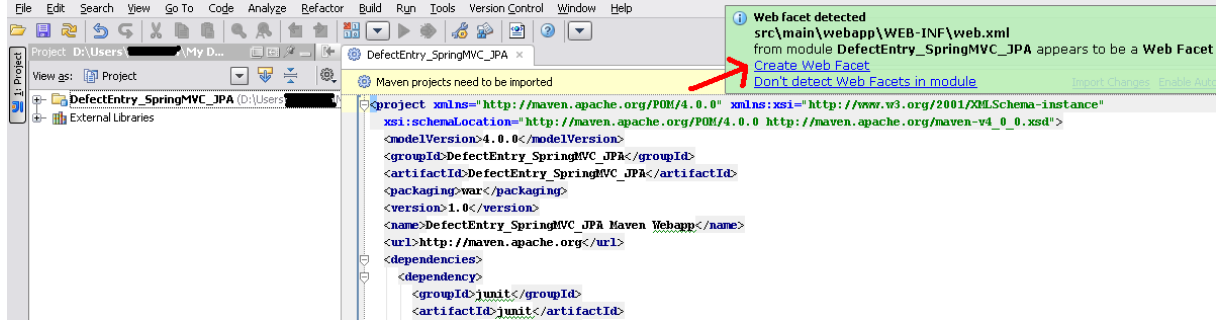
Sonraki ekranda “Maven Module” seçeneği işaretlenir ve projeye bir isim verilir. (DefectEntry_SpringMVC_JPA) Ardından bir sonraki ekrana geçilir.



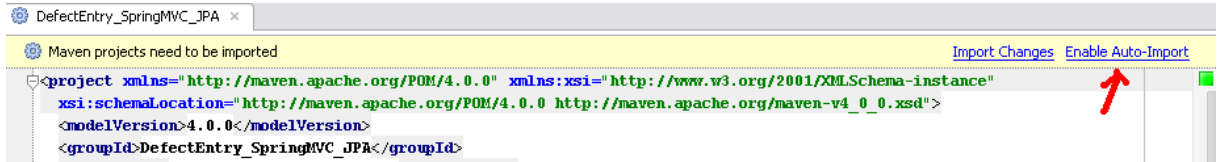
Bu ekranda Maven proje tipi olarak “maven-archetype-webapp” ardından versiyon 1.0 seçilir.



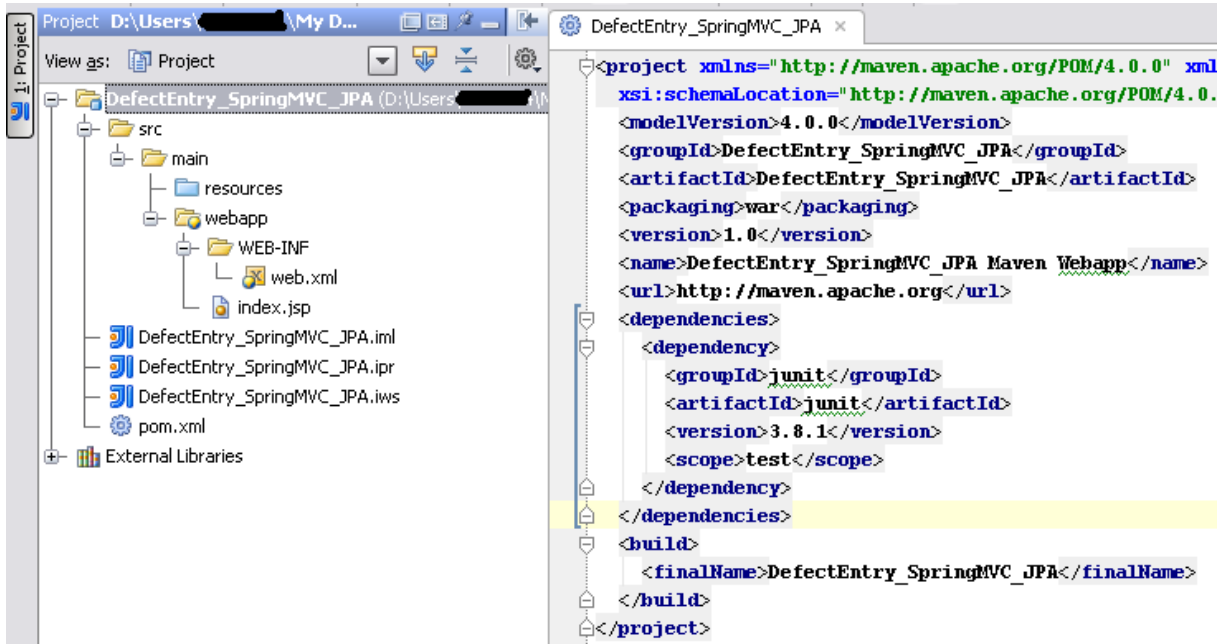
Proje oluşturulduğunda aşağıdaki gibi web görünümü özelliği aktif hale getirilebilir. (Bu geliştirme ortamına ait programcıya yardımcı bir seçenektir. Yapılmasada olur.)



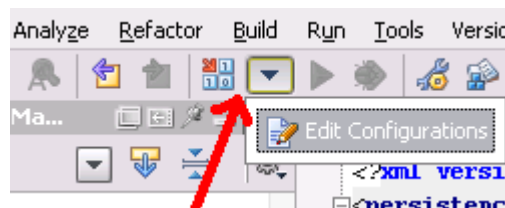
Aşağıdaki ekranda görülen “Enable Auto-Import” özelliği “pom.xml” dosyasına yazılan ayarların otomatik olarak projeye dahil edilmesini sağlar. (Bu işlem de geliştirme ortamına ait programcıya yardımcı bir seçenektir. Yapılmasada olur.)



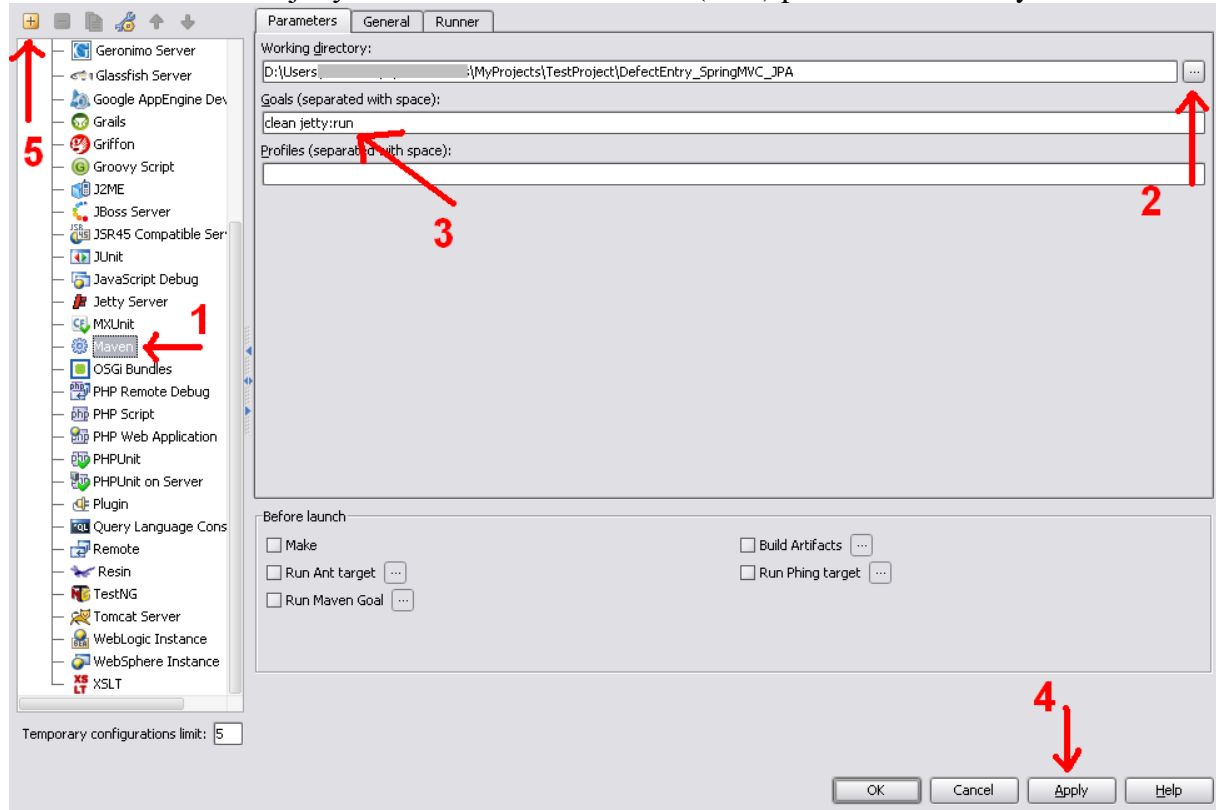
İlk görünüm itibari ile proje ekranı aşağıdaki gibidir.



Oluşturulan projeyi çalıştırabilmek için aşağıdaki gibi yeni bir konfigürasyon ayarlanması gerekmektedir.



Aşağıdaki resimde görüldüğü sıra ile gerekli adımların bir bir uygulanması gerekir. Bu kısımda Goals “clean jetty:run” ile Maven komutuna (mvn) parametre verebiliyoruz.



Not: Bu işlem Maven projesini çalıştırırken hangi parametrelerle çağrılacağını belirler. Alternatif olarak komut satırından ilgili proje dosyasının içerisine girilerek (pom.xml dizininin bulunduğu dizin) bu işlem manuel olarak da gerçekleştirilebilir. (Örnek kod: mvn clean jetty:run)

```
D:\Users\... - \My Documents\MyProjects\TestProject\DefectEntry_SpringMVC_JPA>dir
Volume in drive D has no label.
Volume Serial Number is 58BD-EC61

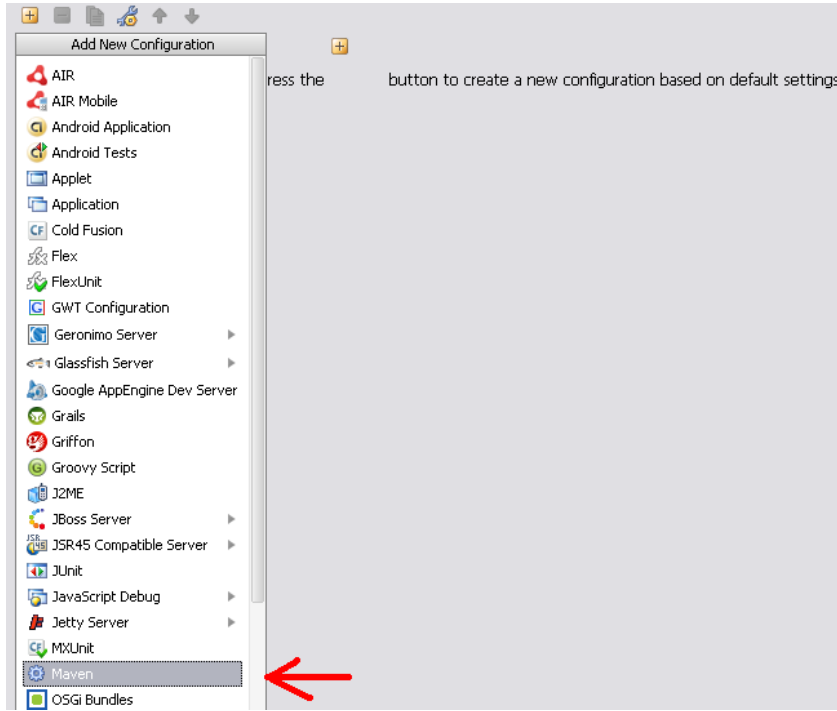
Directory of D:\Users\... \My Documents\MyProjects\TestProject\DefectEntry_SpringMVC_JPA

07/05/2012  09:29    <DIR>          .
07/05/2012  09:29    <DIR>          ..
06/05/2012  21:11             6,150 DefectEntry_SpringMVC_JPA.iml
06/05/2012  21:11             41,049 DefectEntry_SpringMVC_JPA.ipr
07/05/2012  09:29             60,428 DefectEntry_SpringMVC_JPA.iws
06/05/2012  23:30             4,870 pom.xml
03/05/2012  15:39    <DIR>          src
07/05/2012  08:12    <DIR>          target
               4 File(s)          112,497 bytes
               4 Dir(s)  107,671,318,528 bytes free

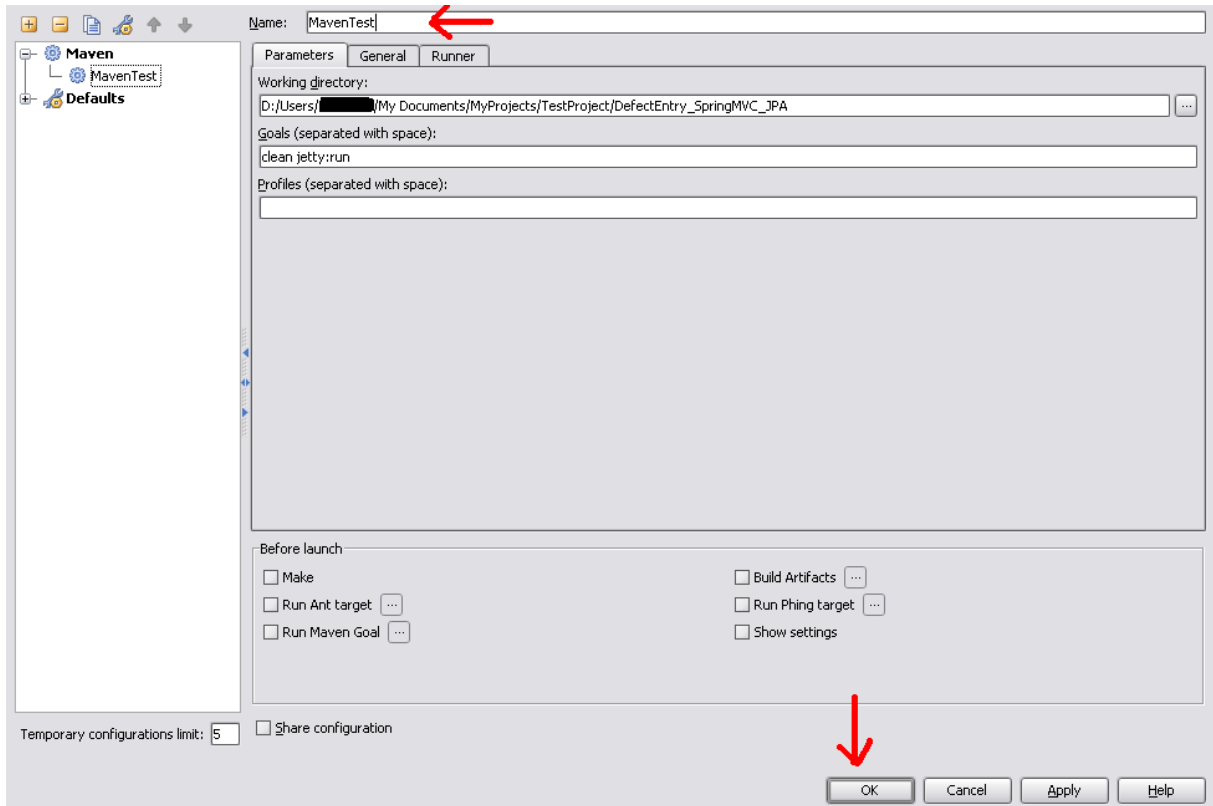
D:\Users\... \My Documents\MyProjects\TestProject\DefectEntry_SpringMVC_JPA>mvn clean jetty:run
```

Not: Bu işlem geliştirme ortamından bağımsız olarak proje çalıştırılmak istendiğinde alternatif olarak izlenebilir.

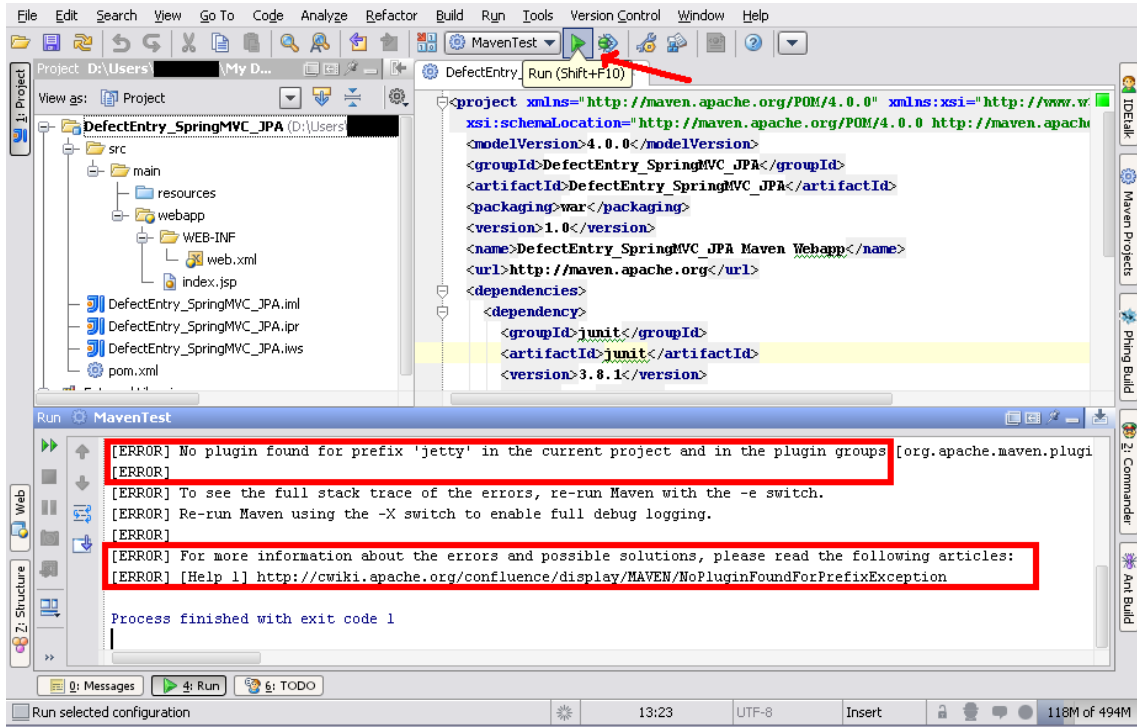
Daha sonra oluşturulan Maven konfigürasyonu sisteme eklenir.



Bu konfigürasyona bir isim verilir ve “Ok” seçeneği ile işlemi tamamlanır.



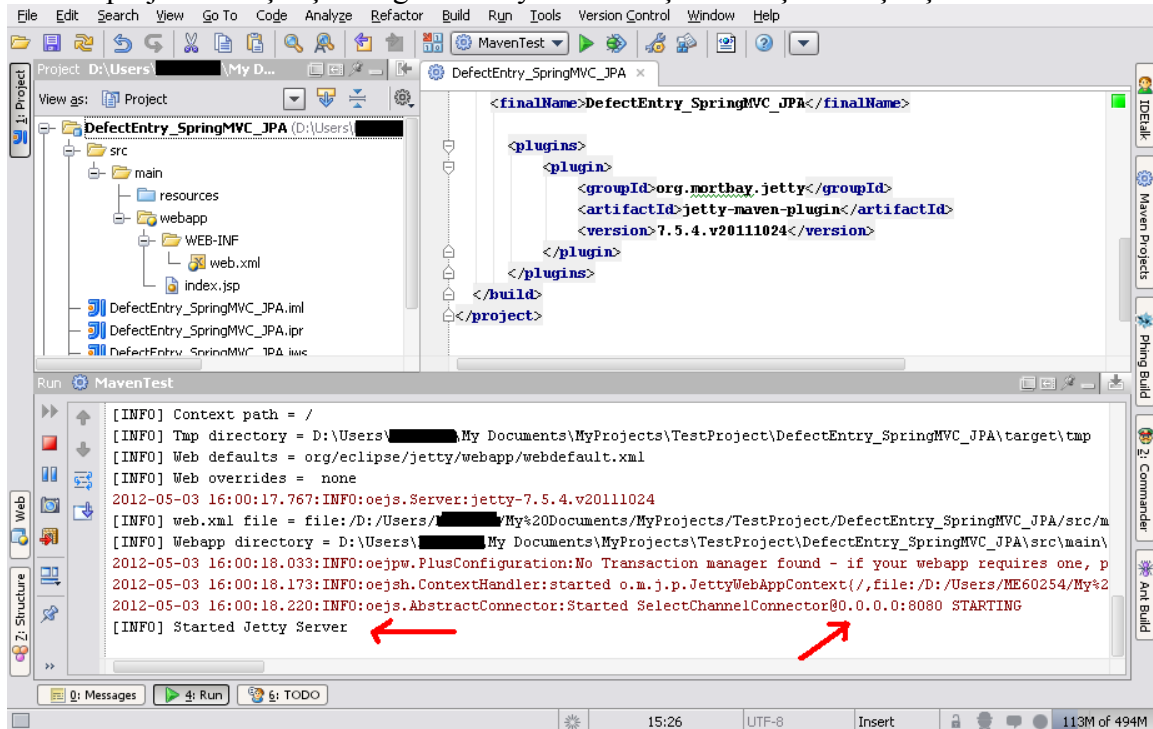
Proje ilk kez çalıştırıldığında bir sonraki şekildeki gibi bir hata alınır. Bu hata “jetty” ‘nin pom.xml dosyasına plugin olarak eklenmesi gerektiğini belirtir.



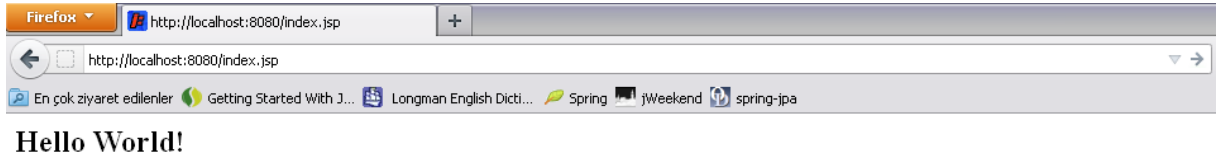
Eklenecek konfigürasyon aşağıdaki gibidir. (<build> ... </build> altına eklenir.)

```
<plugins>
  <plugin>
    <groupId>org.mortbay.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>7.5.4.v20111024</version>
  </plugin>
</plugins>
```

Ardından proje tekrar çalıştırıldığında Jetty Server başarılı bir şekilde çalışacaktır.

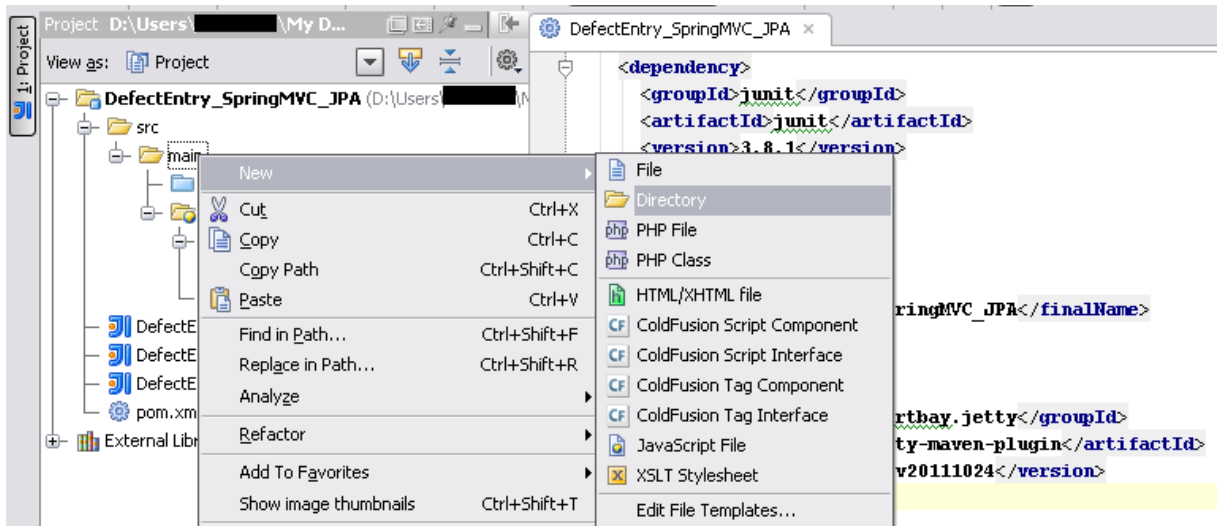


Tarayıcı açılıp adres satırına <http://localhost:8080/index.jsp> yazıldığında aşağıdaki ekran yüklenecektir.

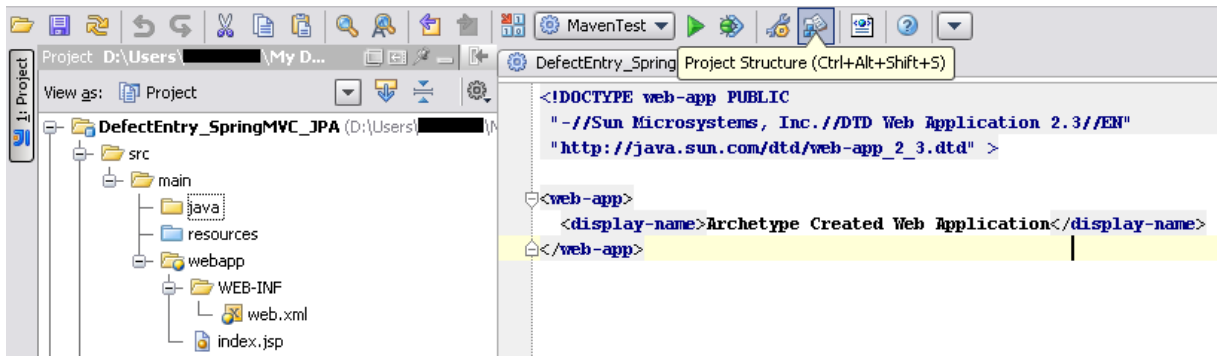


Çalışan uygulamayı durdurmak için Ctrl + F2 tuş kombinasyonu kullanılabilir.

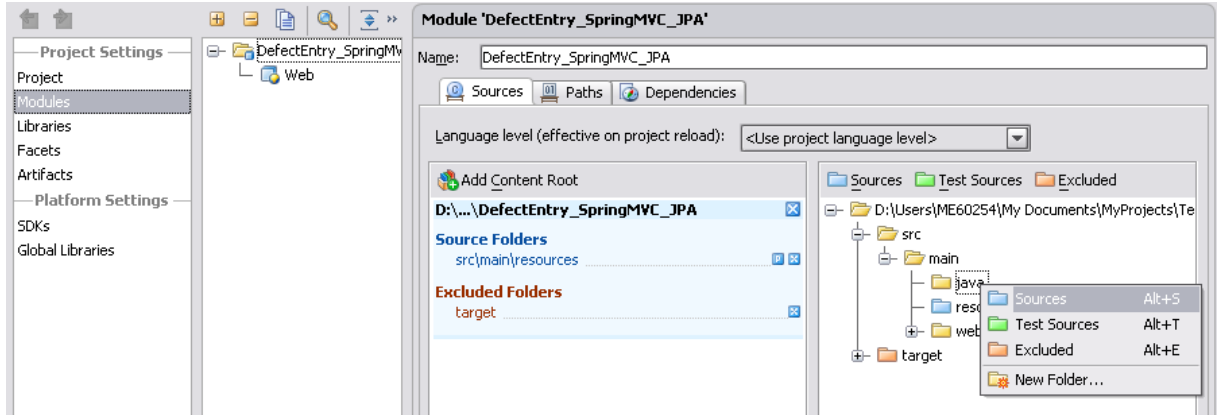
Sonraki aşamada main dizininin altına yeni bir dizin açılması gerekmektedir. Bu dizinin adı “java” dizinidir ve Java kütüphanelerinin yer alacağı dizindir.



Aşağıdaki gibi “Project Structure” menüsünden bu dizin “Sources” olarak tanımlanabilir.



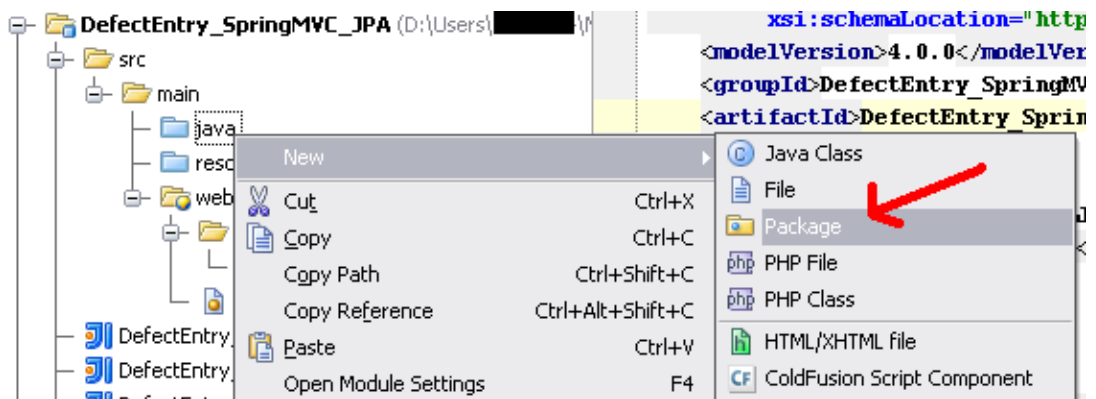
Aşağıdaki ekran görüntüsünde işlemin nasıl yapıldığı gösterilmektedir. (Bu seçenek zorunlu değildir.)



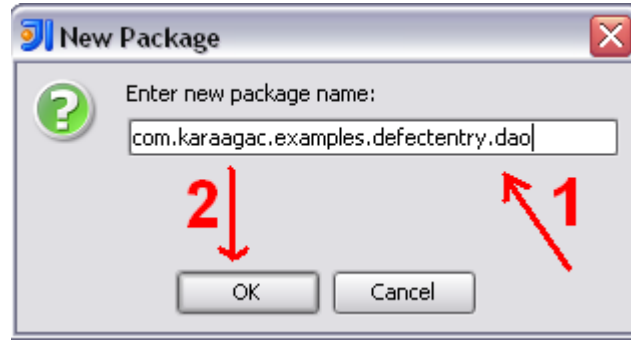
Bunun yanında pom.xml dosyasına aşağıdaki satırlar eklenerek, projenin derlenirken hangi dizindeki ayar dosyalarını okuyacağı belirtilmelidir. (<build> ... </build> altına eklenir.)

```
<resources>
  <resource>
    <directory>src/main/resources</directory>
    <includes>
      <include>*</include>
    </includes>
    <excludes>
      <exclude>*/**/*.java</exclude>
    </excludes>
  </resource>
  <resource>
    <directory>src/main/java</directory>
    <includes>
      <include>*</include>
    </includes>
    <excludes>
      <exclude>*/**/*.java</exclude>
    </excludes>
  </resource>
</resources>
```

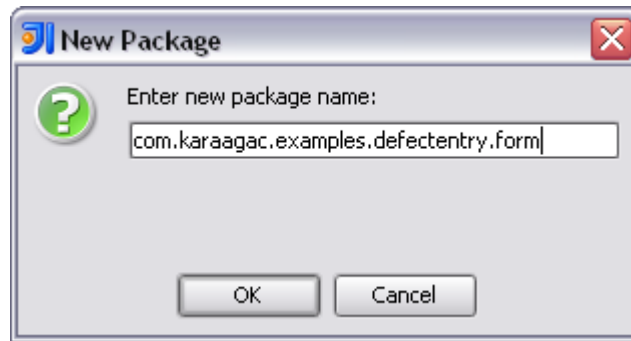
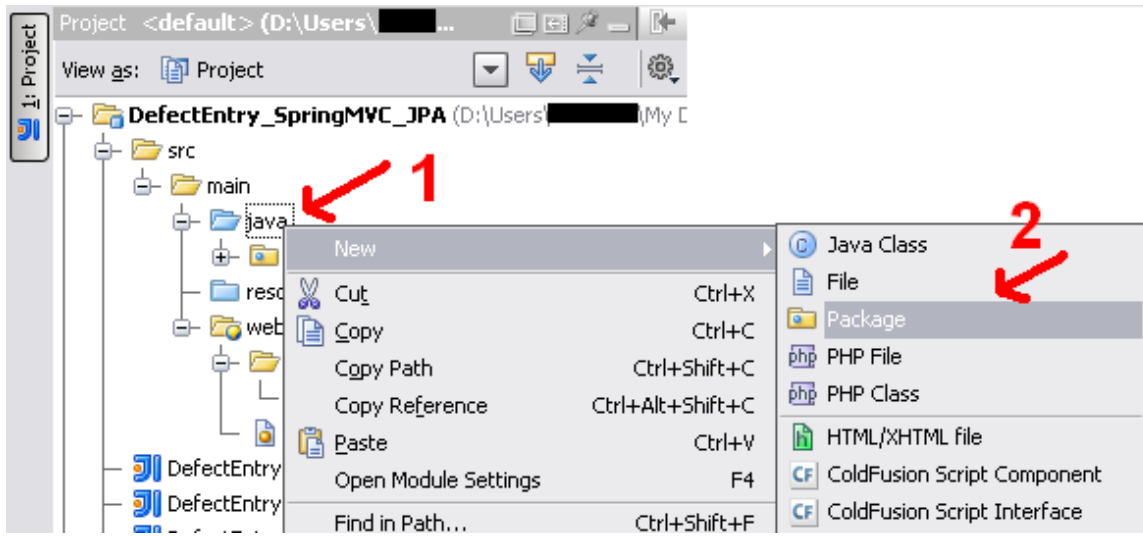
Bu işlemlerden sonra java dizininin altına şekildeki gibi yeni bir paket oluşturulur.



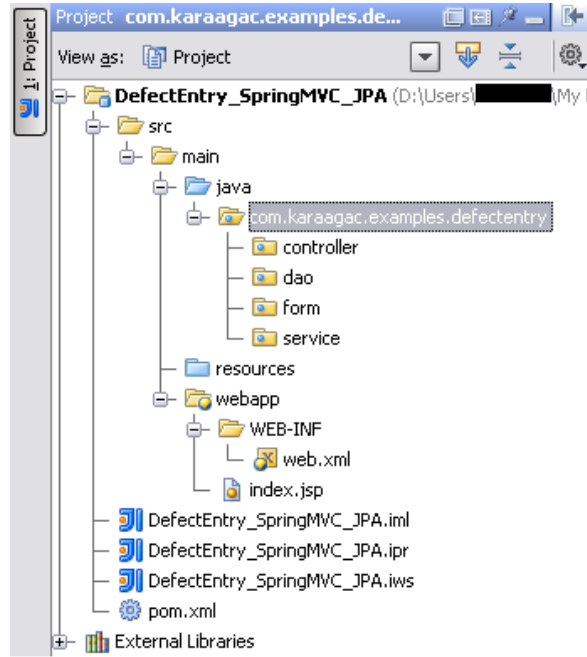
Aşağıdaki ekran görüntüsünde oluşturulan pakete verilen isim görülmektedir.



Bu işlemler tekrarlanarak bir sonraki ekran görüntüsünde olduğu gibi yeni paketler oluşturulur.



Oluşan yapı aşağıda görülmektedir.



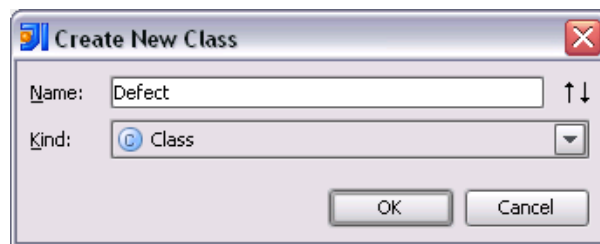
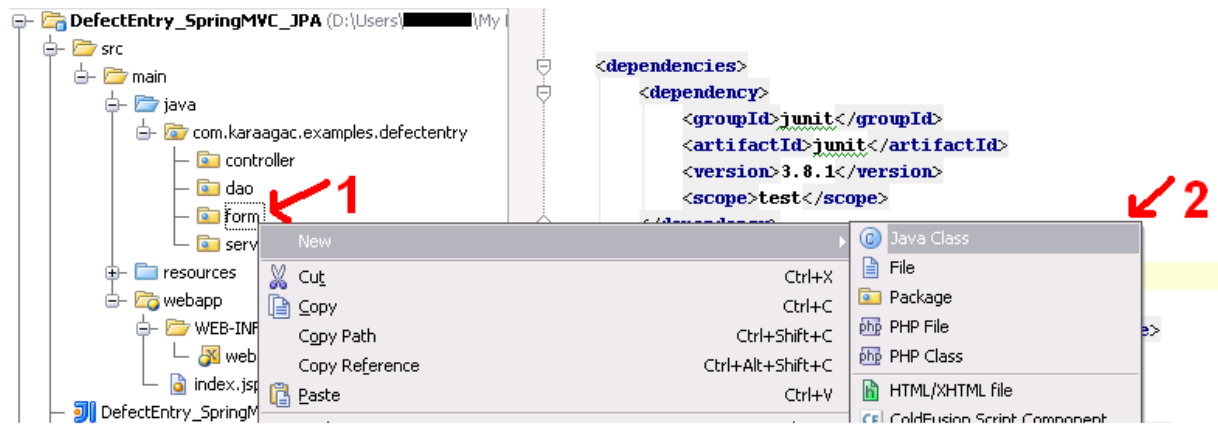
“form” dizininde; veri tabanında olması istenilen Entity ‘lerin tanımlandığı sınıflar yer alacaktır.

“dao” dizininde; veri tabanında yer alacak olan kayıtlara ait “insert”, “delete”, “update”, “select” vb. işlemleri yapacak sınıflar yer alacaktır.

“service” dizininde; veri tabanı işlemlerinin servis edildiği sınıflar yer alacaktır.

“controller” dizininde; kullanıcı işlemlerinin yürütüldüğü, gerekli yönlendirmelerin yapıldığı ve gerekli verilerin işlendiği sınıflar yer alacaktır.

Aşağıdaki gibi Form dizininde Java sınıfı oluşturulur.



Defect.java sınıfının içeriği;

```
package com.karaagac.examples.defectentry.form;
```

```
import javax.persistence.*;
```

```
/**
```

```
 * User: --
```

```
 * Date: --
```

```
 */
```

```
@Entity
```

```
@Table(name="defects")
```

```
public class Defect {
```

```
    @Id
```

```
    @Column(name="id", insertable = false, updatable = false)
```

```
    @SequenceGenerator(name="SequenceIdGenerator", sequenceName = "defects_sequence")
```

```
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SequenceIdGenerator")
```

```
    private Integer id;
```

```
    @Column(name="defectcode")
```

```
    private String defectCode;
```

```
    @Column(name="defectdescription")
```

```
    private String defectDescription;
```

```
    @Column(name="defecttype")
```

```
    private String defectType;
```

```
    public Integer getId() {
```

```
        return id;
```

```
    }
```

```
    public void setId(Integer id) {
```

```
        this.id = id;
```

```
    }
```

```
    public String getDefectCode() {
```

```
        return defectCode;
```

```
    }
```

```
    public void setDefectCode(String defectCode) {
```

```
        this.defectCode = defectCode;
```

```
    }
```

```
    public String getDefectDescription() {
```

```
        return defectDescription;
```

```
    }
```

```
    public void setDefectDescription(String defectDescription) {
```

```
        this.defectDescription = defectDescription;
```

```
    }
```

```
    public String getDefectType() {
```

```
        return defectType;
```

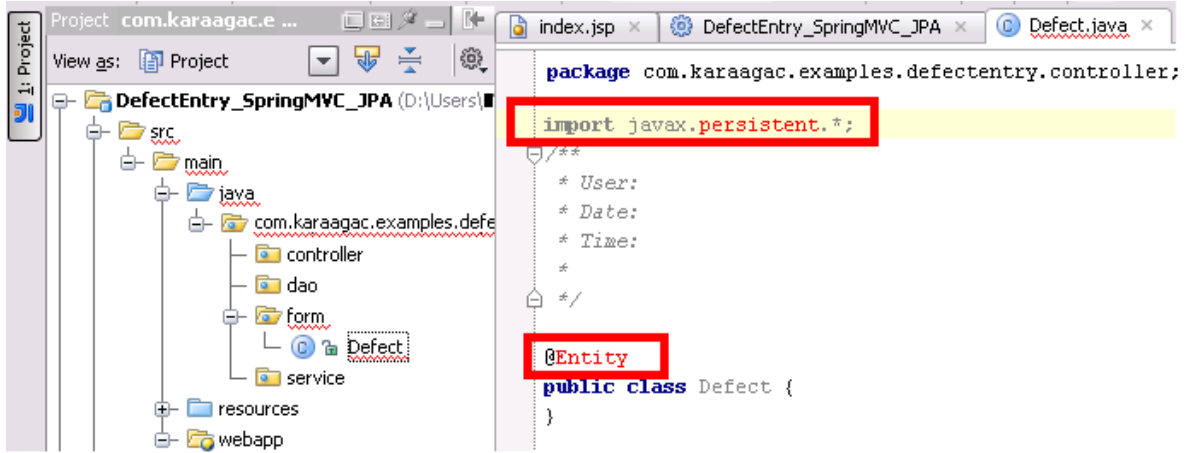
```
    }
```

```
    public void setDefectType(String defectType) {
```

```
        this.defectType = defectType;
```

```
}  
}
```

Aşağıdaki gibi bir uyarıyla karşılaşılması normaldir.



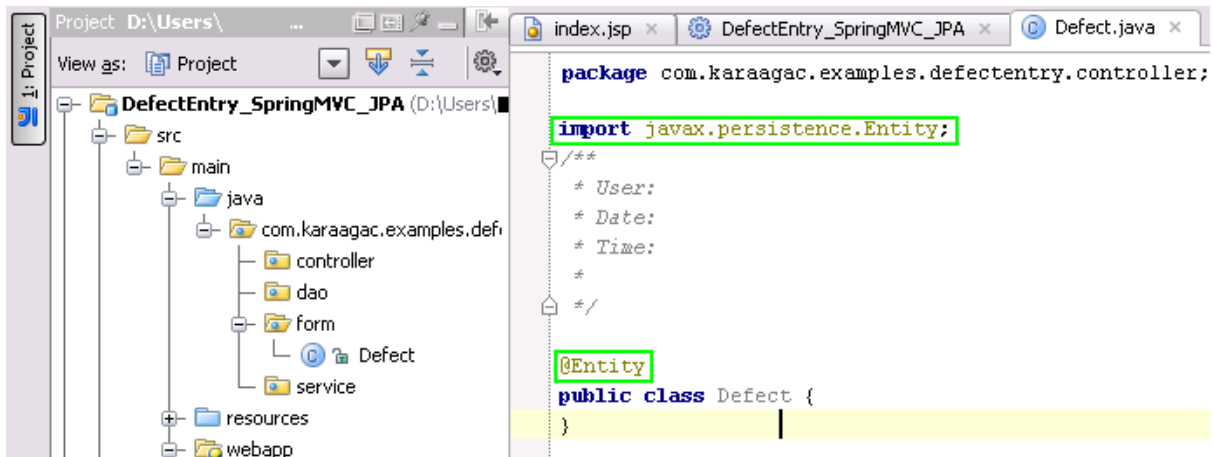
Bu durumda, pom.xml içerisine javax.persistence kütüphanelerinin kullanıldığının belirtilmesi gerekir.

Not: "pom.xml" dosyasının tam içeriği ilerleyen kısımlarda verilmiştir. (<dependencies>...</dependencies> altına eklenmelidir.)

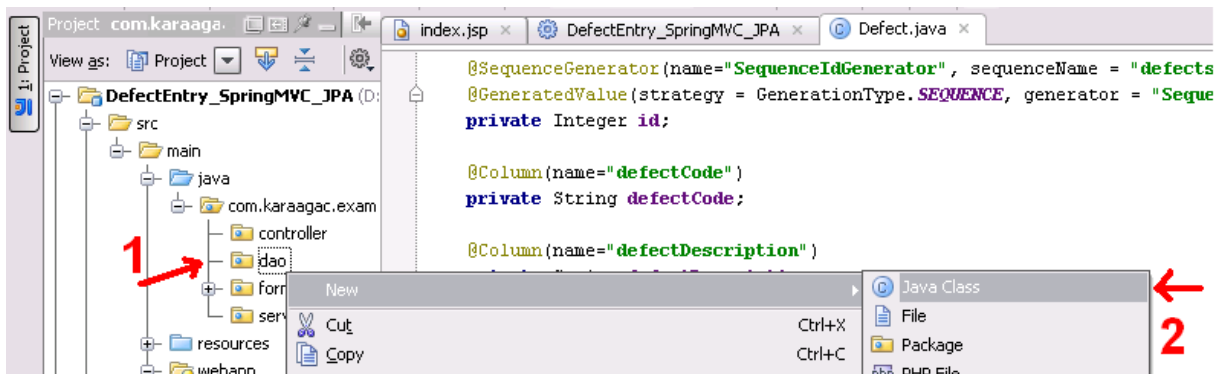
```
<dependencies>  
  <dependency>  
    <groupId>junit</groupId>  
    <artifactId>junit</artifactId>  
    <version>3.8.1</version>  
    <scope>test</scope>  
  </dependency>  
  <dependency>  
    <groupId>javax.persistence</groupId>  
    <artifactId>persistence-api</artifactId>  
    <version>1.0.2</version>  
  </dependency>  
</dependencies>  
  
<dependency>  
  <groupId>javax.persistence</groupId>  
  <artifactId>persistence-api</artifactId>  
  <version>1.0.2</version>  
</dependency>
```

Not: Geliştirilen proje bir Maven projesi olduğu için gerekli kütüphane tanımlamalarının pom.xml içerisinde yapılması gerekmektedir.

Bu işlemden sonra kod satırı normale dönecektir.



Bir sonraki adımda “dao” içerisine yeni bir interface oluşturulur.



Oluşturulan Interface ‘in içeriği aşağıdaki gibidir.

DefectDao.java

```
package com.karaagac.examples.defectentry.dao;
```

```
import com.karaagac.examples.defectentry.form.Defect;
import java.util.List;
```

```
/**
```

```
 * User: --
```

```
 * Date: --
```

```
 */
```

```
public interface DefectDao {
```

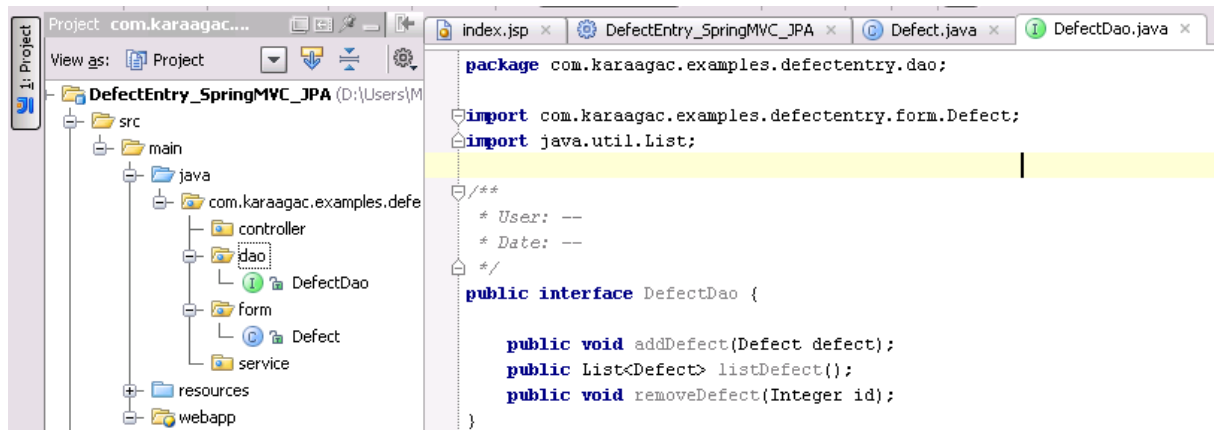
```
    public void addDefect(Defect defect);
```

```
    public List<Defect> listDefect();
```

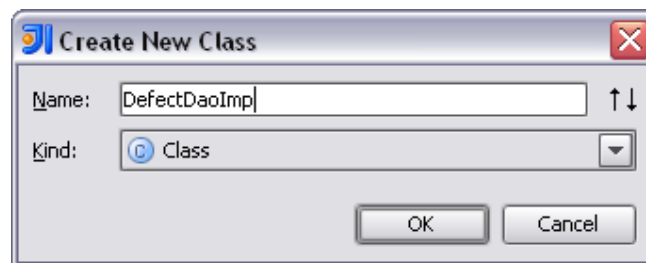
```

    public void removeDefect(Integer id);
}

```



Daha sonra bu sınıfın implemente edildiği DefectDaoImp sınıfını oluşturuyoruz.



DefectDaoImp.java Source

```

package com.karaagac.examples.defectentry.dao;

import com.karaagac.examples.defectentry.form.Defect;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.*;
import java.util.List;

public class DefectDaoImp implements DefectDao{

    EntityManager entityManager;

    @PersistenceContext
    public void setEntityManager(EntityManager entityManager)
    {
        this.entityManager = entityManager;
    }

    @Transactional
    public void addDefect(Defect defect) throws EntityExistsException{

        Query query = entityManager.createQuery("from Defect def where def.defectCode = :defectcode");
        query.setParameter("defectcode", defect.getDefectCode());

        List<Defect> defectList = query.getResultList();

        if(! defectList.isEmpty())
        {
            throw new EntityExistsException("Defect code already defined.");
        }
    }
}

```

```

    }

    entityManager.persist(defect);
}

public List<Defect> listDefect() {
    List<Defect> defectList = entityManager.createQuery("from Defect").getResultList();
    return defectList;
}

@Transactional
public void removeDefect(Integer id) {

    Defect defect = null;
    defect = entityManager.find(Defect.class,id);

    if(defect != null)
    {
        entityManager.remove(defect);
    }
}
}

```

“pom.xml” içerisinde aşağıdaki değişiklikler yapılmalıdır.

<project> ... </project> altına eklenecek özellikler

```

<properties>
    <org.springframework.version>3.1.0.RELEASE</org.springframework.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

```

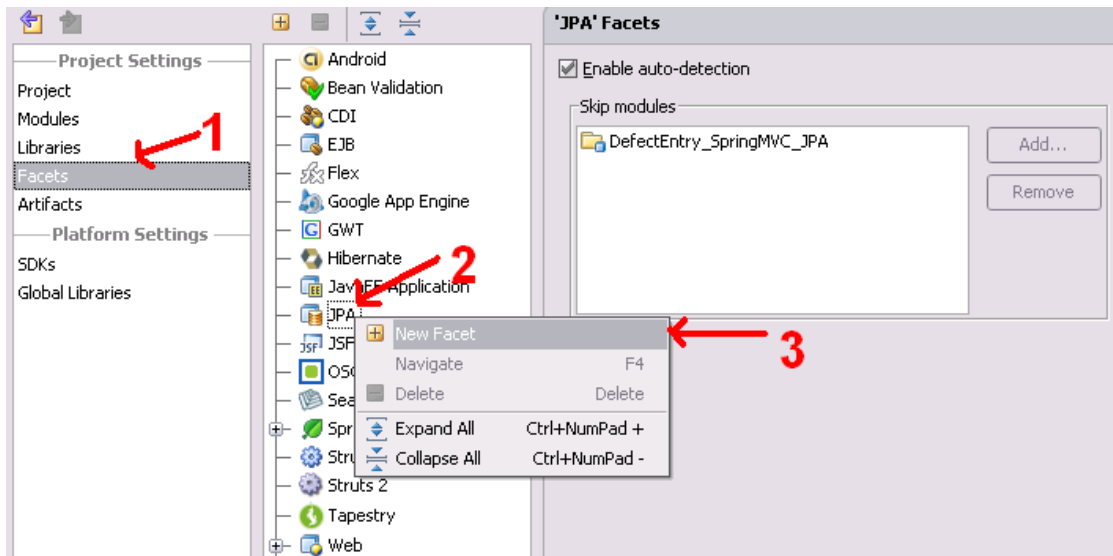
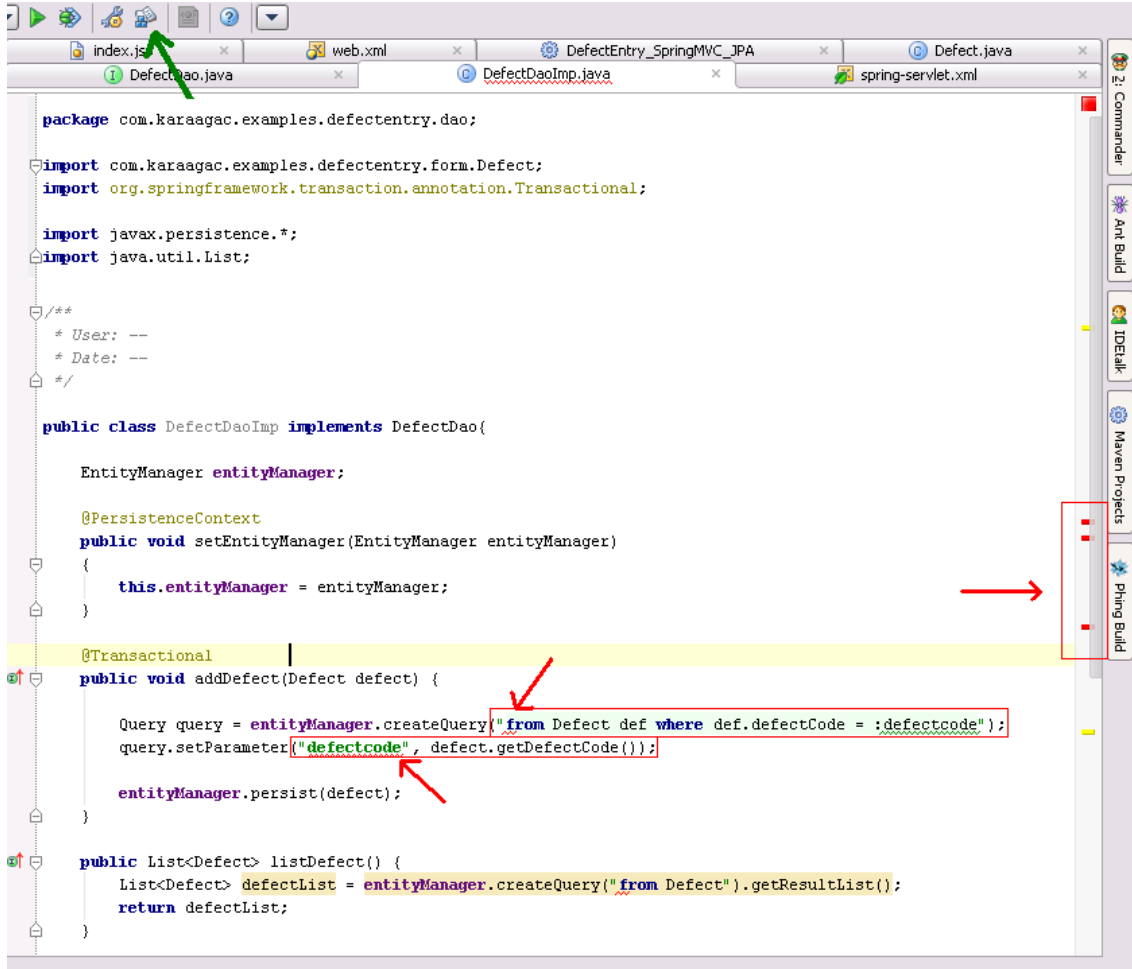
<dependencies> ... </dependencies> altına eklenecek kütüphaneler

```

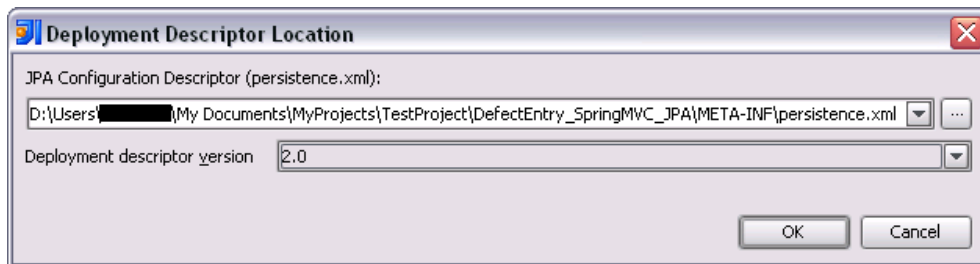
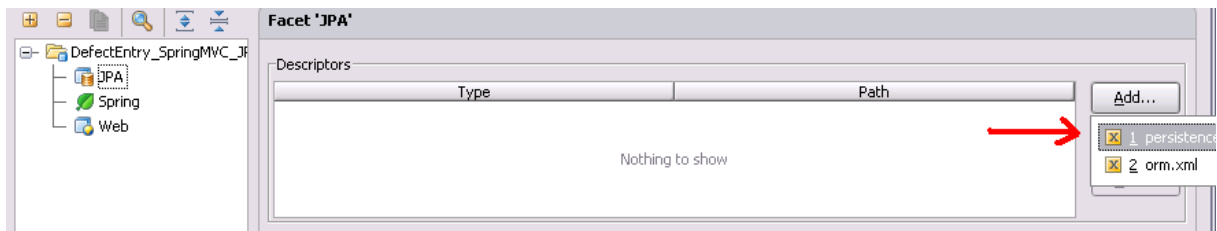
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>${org.springframework.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${org.springframework.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${org.springframework.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${org.springframework.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>${org.springframework.version}</version>
</dependency>

```

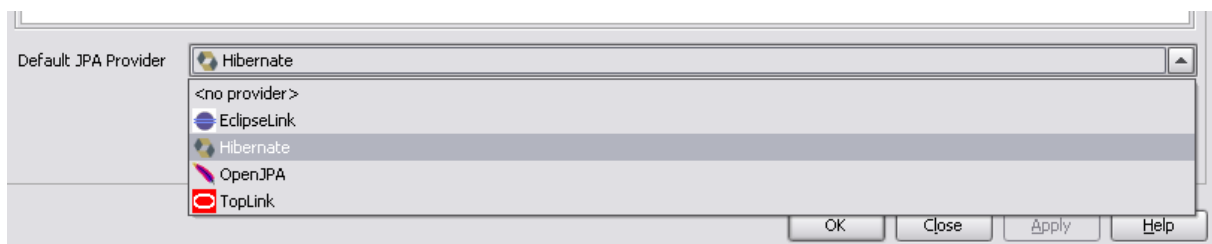
Bu kısımda yazılan kodlar doğru olduğu halde IntelliJ bize uyarı vermektedir. Bu uyarıyı ortadan kaldırmak için aşağıdaki adımlar uygulanabilir. (Yeşil ok:Project Structure)



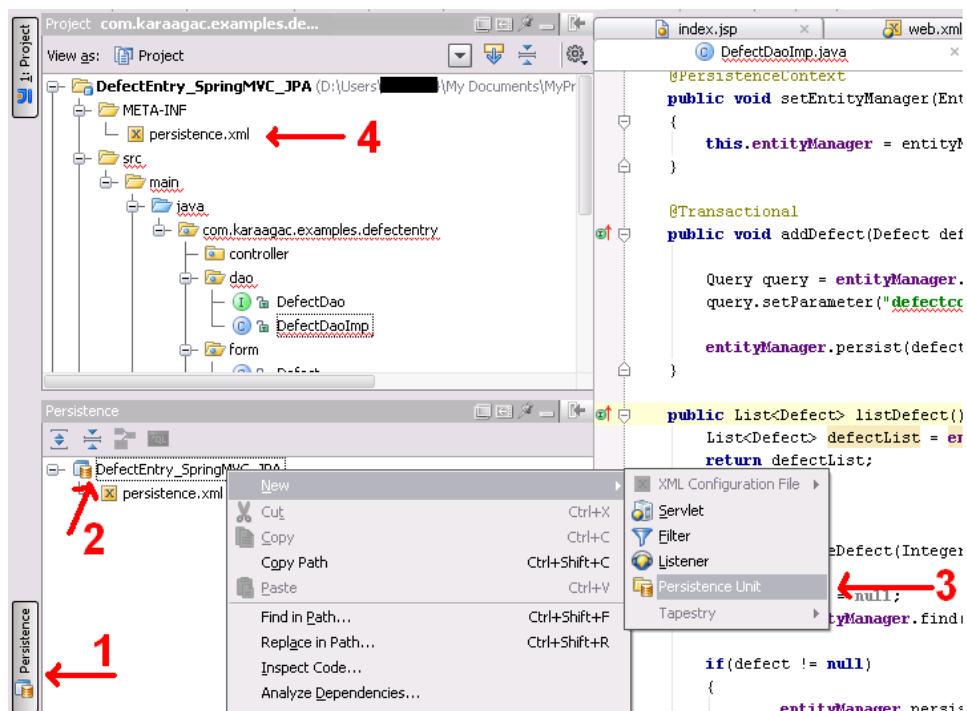
Sonraki ekranda aşağıdaki işlem yapılmalıdır.



“Default JPA Provider” “Hibernate” olarak değiştirilmelidir.

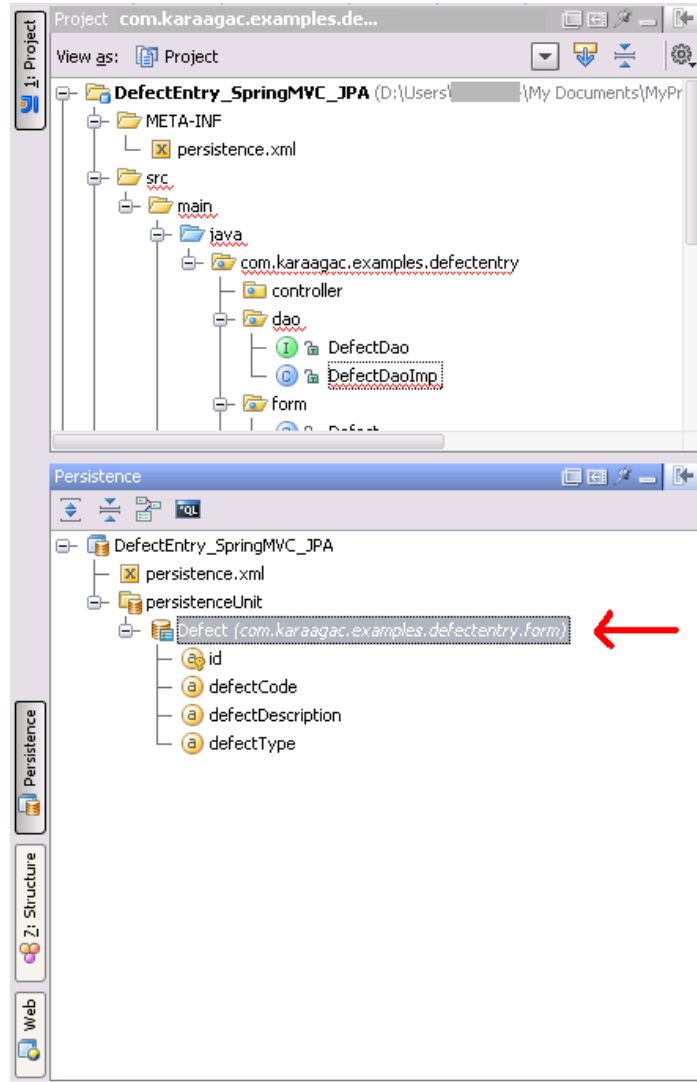


Daha sonra aşağıdaki işlemler yapılmalıdır.

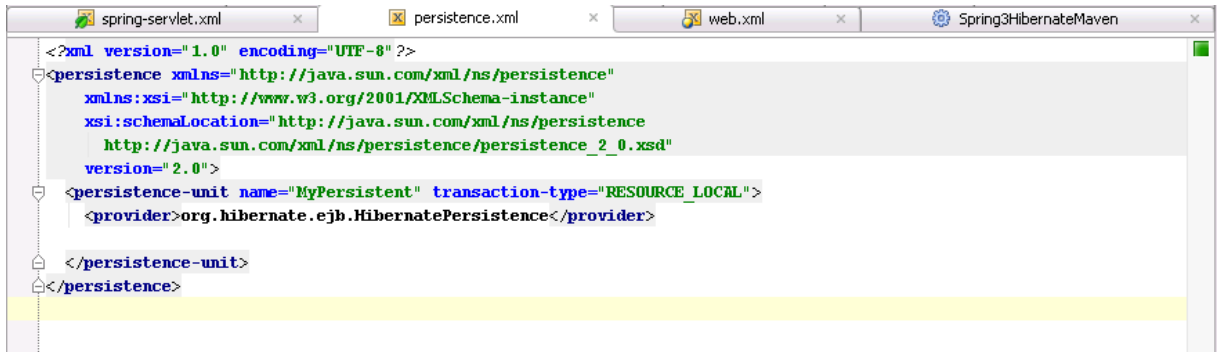


Persistence Unit ismini “Defect” olarak veriyoruz.

Bu işlemler JPA konfigürasyonu için gereklidir.



Oluşan “persistence.xml” dosyasının içeriği aşağıdaki gibidir.



Bu içerik aşağıdaki gibi değiştirilmelidir.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">
  <persistence-unit name="MyPersistent" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>com.karaagac.examples.defectentry.form.Defect</class>

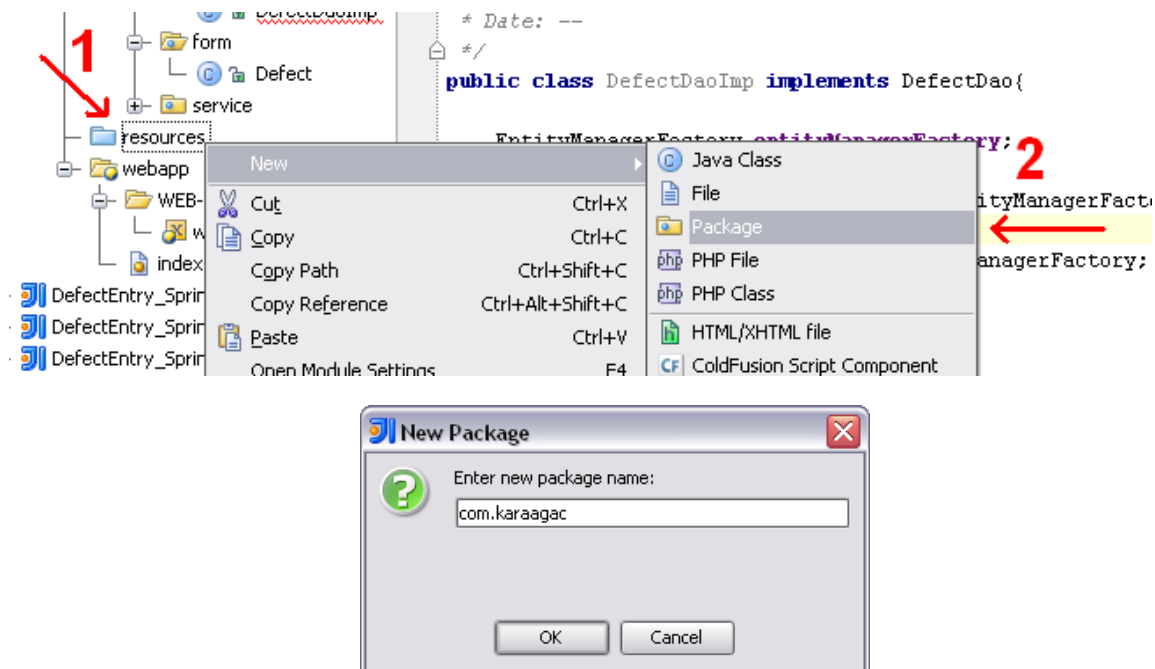
    <properties>
      <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="true"/>
    </properties>

  </persistence-unit>
</persistence>
```

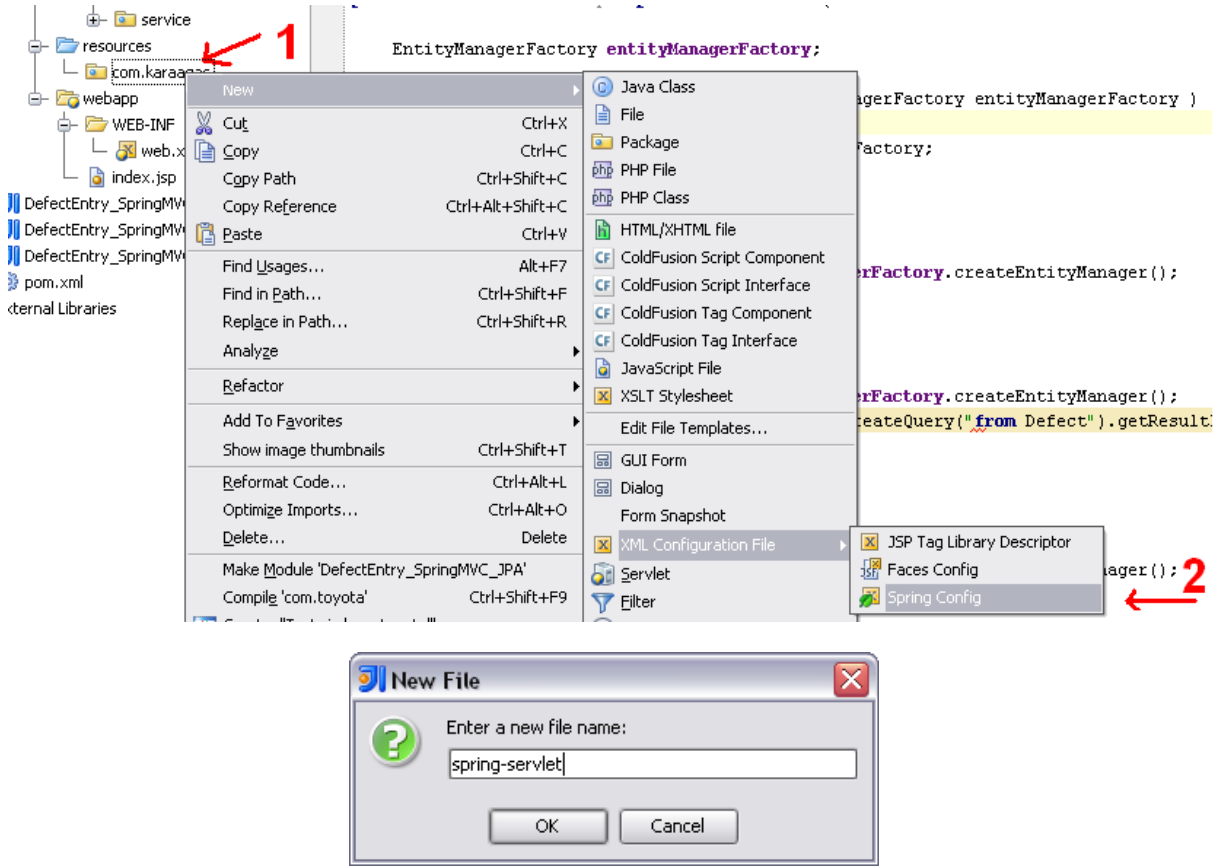
“pom.xml” içerisine kullanılan kütüphaneler için aşağıdaki satırlar eklenmelidir. (<dependencies> ... </dependencies> altına)

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>3.5.6-Final</version>
</dependency>
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>persistence-api</artifactId>
  <version>1.0.2</version>
</dependency>
```

Resources dizininin altına yeni bir com.karaagac dizini oluşturulur. Bu dizin altında spring kanfigurasyonu yapılacaktır.



Aşağıdaki ekran görüntüsünde bir sonraki aşamada yapılması gerekenler listelenmektedir.



spring-servlet.xml dosyası.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xmlns:lang="http://www.springframework.org/schema/lang"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee.xsd
http://www.springframework.org/schema/lang
http://www.springframework.org/schema/lang/spring-lang.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util.xsd">

    <context:annotation-config />
    <context:component-scan base-package="com.karaagac.examples.defectentry.controller"/>
```

```

        <bean id="jspViewResolver"
            class="org.springframework.web.servlet.view.InternalResourceViewResolver">
                <property name="viewClass"
                    value="org.springframework.web.servlet.view.JstlView" />
                <property name="prefix" value="/WEB-INF/jsp/" />
                <property name="suffix" value=".jsp" />
            </bean>

        <bean id="messageSource"
            class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
                <property name="basename" value="classpath:messages" />
                <property name="defaultEncoding" value="UTF-8" />
            </bean>

        <bean id="propertyConfigurer"
            class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"
            p:location="classpath:jdbc.properties"/>

        <tx:annotation-driven transaction-manager="transactionManager" />

        <bean id="transactionManager"
            class="org.springframework.orm.jpa.JpaTransactionManager">
                <property name="entityManagerFactory" ref="entityManagerFactory"/>
            </bean>

        <bean id="entityManagerFactory"
            class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"
            p:dataSource-ref="datasource" p:persistenceUnitName="MyPersistent">

            <property name="jpaVendorAdapter">
                <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
                    p:generateDdl="false"
                    p:showSql="${flrs.jpa.showSql}"
                    p:databasePlatform="${jdbc.dialect}"/>
            </property>
        </bean>

        <bean id="datasource"
            class="com.mchange.v2.c3p0.ComboPooledDataSource"
            p:driverClass="${jdbc.driverClassName}"
            p:jdbcUrl="${jdbc.databaseurl}"
            p:user="${jdbc.username}"
            p:maxIdleTime="${jdbc.maxIdleTime}"
            p:maxPoolSize="${jdbc.maxPoolSize}"
            p:password="${jdbc.password}"
            p:preferredTestQuery="${flrs.jpa.showSql}"/>

        <bean id="defectService"
            class="com.karaagac.examples.defectentry.service.DefectServiceImpl">
                <property name="defectDAO" ref="defectDao"/>
            </bean>
        <bean id="defectDao"
            class="com.karaagac.examples.defectentry.dao.DefectDaoImp">
            </bean>

    </beans>

```

Burada kırmızı renkle belirtilen kütüphane hatasını kaldırmak için “c3p0” kütüphanesini pom.xml içerisine eklenmelidir.

```
<bean id="entityManagerFactory"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"
      p:dataSource-ref="dataSource" p:persistenceUnitName="MyPersistent">

    <property name="jpaVendorAdapter">
      <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
            p:generatedDdl="false"
            p:showSql="${flrs.jpa.showSql}"
            p:databasePlatform="${jdbc.dialect}" />
    </property>
</bean>

<bean id="dataSource"
      class="com.mchange.v2.c3p0.ComboPooledDataSource"
      p:driverClass="${jdbc.driverClassName}"
      p:jdbcUrl="${jdbc.databaseurl}"
      p:user="${jdbc.username}"
      p:maxIdleTime="${jdbc.maxIdleTime}"
      p:maxPoolSize="${jdbc.maxPoolSize}"
      p:password="${jdbc.password}"
      p:preferredTestQuery="${flrs.jpa.showSql}" />

<bean id="defectService"
      class="com.karaagac.examples.defectentry.service.DefectServiceImpl">
  <property name="defectDAO" ref="defectDao" />
</bean>

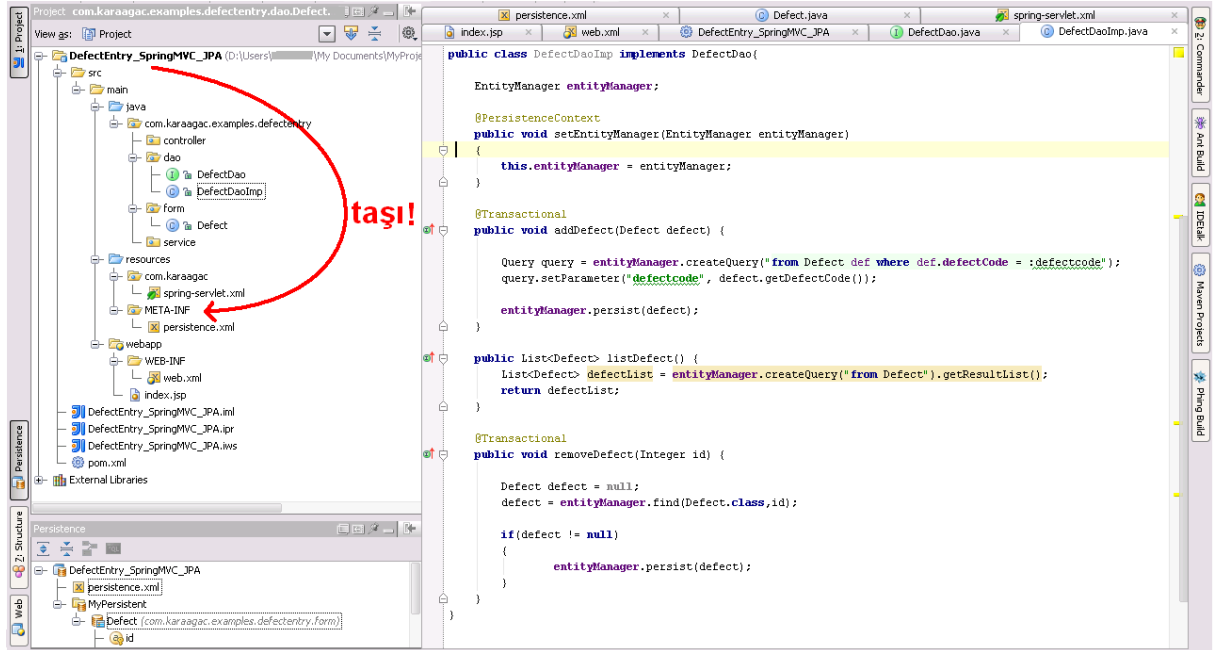
</beans>
```

```
<artifactId>persistence-api</artifactId>
<version>1.0.2</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>3.5.6-Final</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>3.5.6-Final</version>
</dependency>
<dependency>
  <groupId>c3p0</groupId>
  <artifactId>c3p0</artifactId>
  <version>0.9.1</version>
</dependency>
</dependencies>
```

<dependency> ... </dependency> içerisine aşağıdaki satırlar eklenmelidir.

```
<dependency>
  <groupId>c3p0</groupId>
  <artifactId>c3p0</artifactId>
  <version>0.9.1</version>
</dependency>
```

META-INF dizininin yeri doğru değildir. Bu yüzden resources altına taşınmalıdır.



Aşağıdaki gibi “servis” dizininin altına gerekli sınıf ve interface dosyaları oluşturulmalıdır.



DefectService.java Code:

```
package com.karaagac.examples.defectentry.service;
```

```
import com.karaagac.examples.defectentry.form.Defect;  
import java.util.List;
```

```
/**  
 * User: ---  
 * Date: ---  
 */
```

```
public interface DefectService {  
  
    public void addDefect(Defect defect);  
    public List<Defect> listDefect();  
    public void removeDefect(Integer id);  
}
```

DefectServiceImpl.java Code;

```
package com.karaagac.examples.defectentry.service;
```

```
import com.karaagac.examples.defectentry.dao.DefectDao;
import com.karaagac.examples.defectentry.form.Defect;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
```

```
/**
```

```
 * User: ---
```

```
 * Date: ---
```

```
 */
```

```
public class DefectServiceImpl implements DefectService{
```

```
    private DefectDao defectDao;
```

```
    public void setDefectDAO(DefectDao defectDao)
```

```
    {
```

```
        this.defectDao = defectDao;
```

```
    }
```

```
    public DefectDao getDefectDAO()
```

```
    {
```

```
        return this.defectDao;
```

```
    }
```

```
        @Transactional
```

```
        public void addDefect(Defect defect) {
```

```
            defectDao.addDefect(defect);
```

```
        }
```

```
        public List<Defect> listDefect() {
```

```
            return defectDao.listDefect();
```

```
        }
```

```
        @Transactional
```

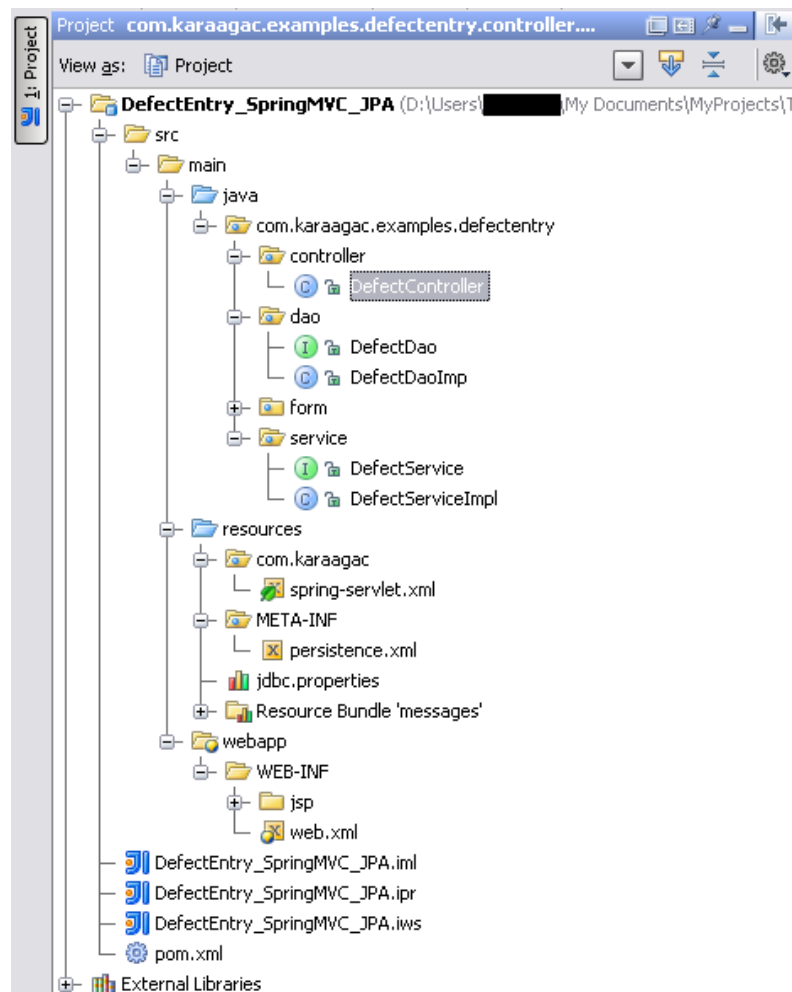
```
        public void removeDefect(Integer id) {
```

```
            defectDao.removeDefect(id);
```

```
        }
```

```
    }
```

Daha sonra controller dizininde DefectController sınıfı yer almalıdır.



DefectController.java Code;

```
package com.karaagac.examples.defectentry.controller;

import com.karaagac.examples.defectentry.form.Defect;
import com.karaagac.examples.defectentry.service.DefectService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import javax.persistence.EntityExistsException;
import java.util.Map;

/**
 * User: ---
 * Date: ---
 */

@Controller
public class DefectController {

    @Autowired
    private DefectService defectService;
```

```

@RequestMapping("/")
    public String welcomeHandler() {
        return "redirect:/index";
    }

    @RequestMapping("/index")
    public String listDefects(Map<String, Object> map) {

        map.put("defect", new Defect());
        map.put("defectList", defectService.listDefect());

        return "defect";
    }

@RequestMapping("/eindex")
    public String eListDefects(Map<String, Object> map) {

        map.put("error", "Hata verisi girilmedi! ( Ilgili veri zaten var. )");
        map.put("defect", new Defect());
        map.put("defectList", defectService.listDefect());

        return "defect";
    }

    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public String addDefect(@ModelAttribute("defect")
        Defect defect, BindingResult result){
        try{
            defectService.addDefect(defect);
        }catch (EntityExistsException ex)
        {
            return "redirect:/eindex";
        }

        return "redirect:/index";
    }

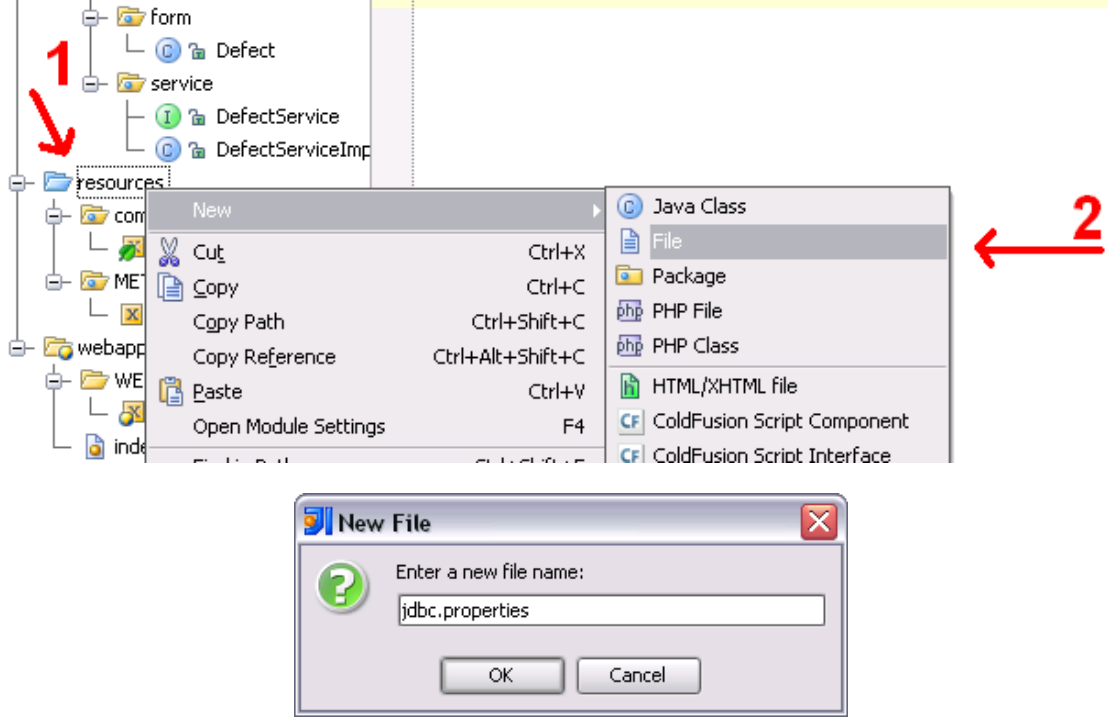
    @RequestMapping("/delete/{defectId}")
    public String deleteDefect(@PathVariable("defectId")
        Integer defectId) {

        defectService.removeDefect(defectId);

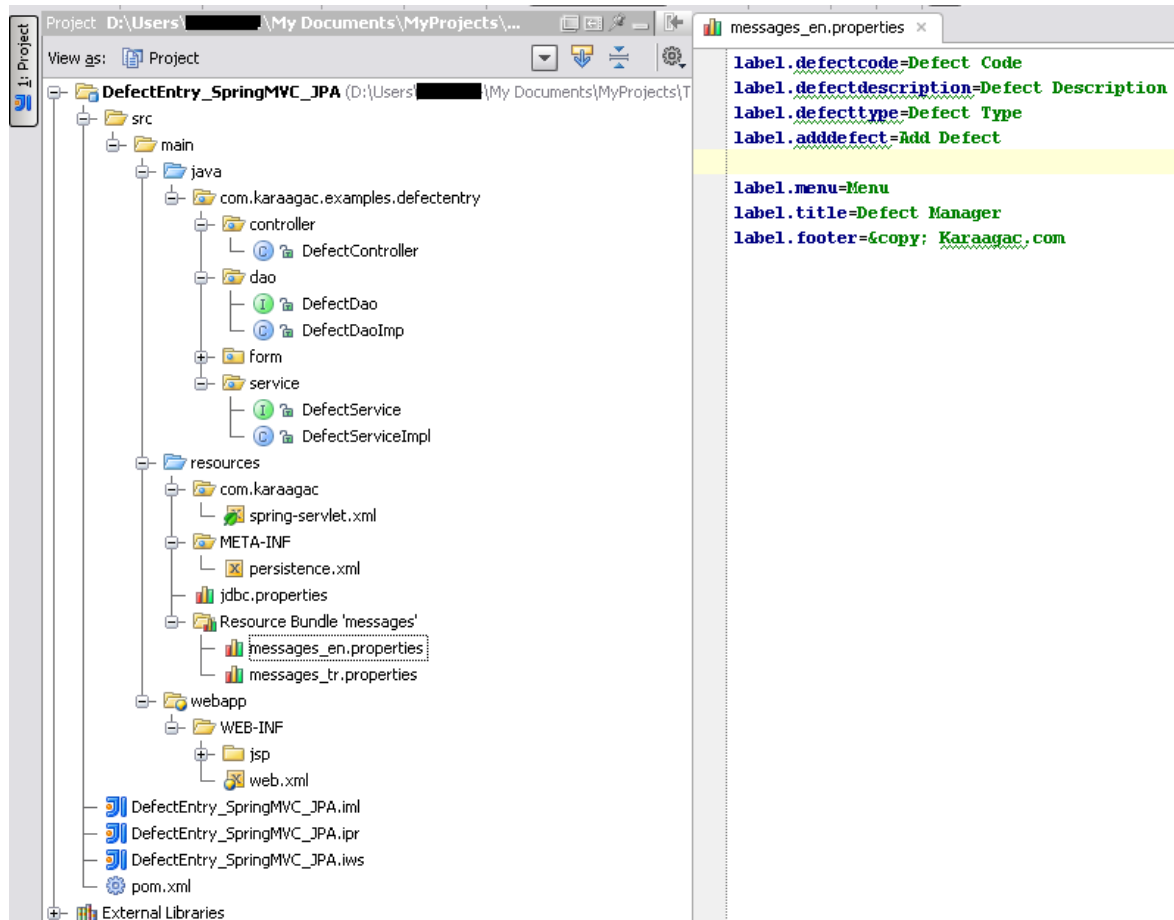
        return "redirect:/index";
    }
}

```


Sonraki aşamada veritabanı konfigürasyonunun okunacağı dosya resources altına oluşturulur.



Aynı şekilde dil dosyalarının okunacağı dosyalar da aynı yolla oluşturulur.



jdbc.properties dosyası;

```
jdbc.driverClassName= org.postgresql.Driver
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
jdbc.databaseurl=jdbc:postgresql://localhost:5432/MyDB
jdbc.schema=public
jdbc.username=ufuk
jdbc.password=1qaz2wsx
jdbc.maxIdleTime=180000
jdbc.maxPoolSize=100
flrs.jpa.showSql=true
```

message_en.properties dosyası;

```
label.defectcode=Defect Code
label.defectdescription=Defect Description
label.defecttype=Defect Type
label.adddefect=Add Defect
```

```
label.menu=Menu
label.title=Defect Manager
label.footer=&copy; Karaagac.com
```

message_tr.properties dosyası;

```
label.defectcode=Hata Kodu
label.defectdescription=Hata Tanimi
label.defecttype=Hata Tipi
label.adddefect=Hata Ekle
```

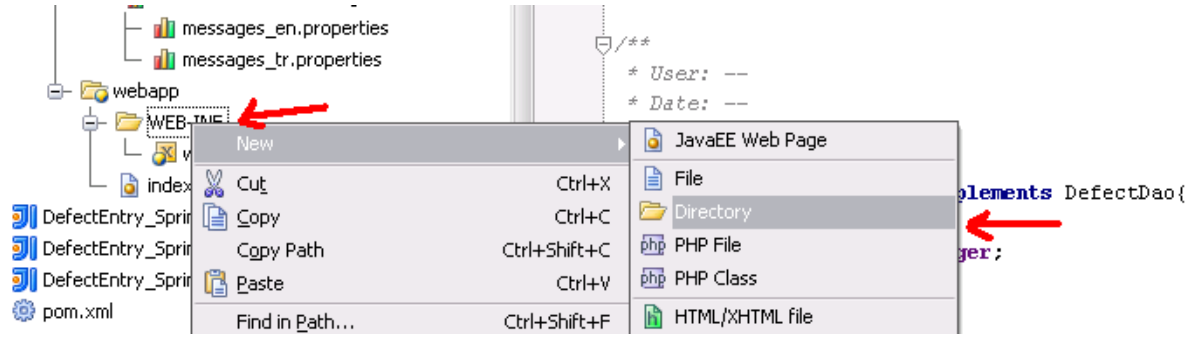
```
label.menu=Menu
label.title=Hata Yöneticisi
label.footer=&copy; Karaagac.com
```

Değişiklikler yapıldıktan sonraki görüntü aşağıda gösterilmektedir.

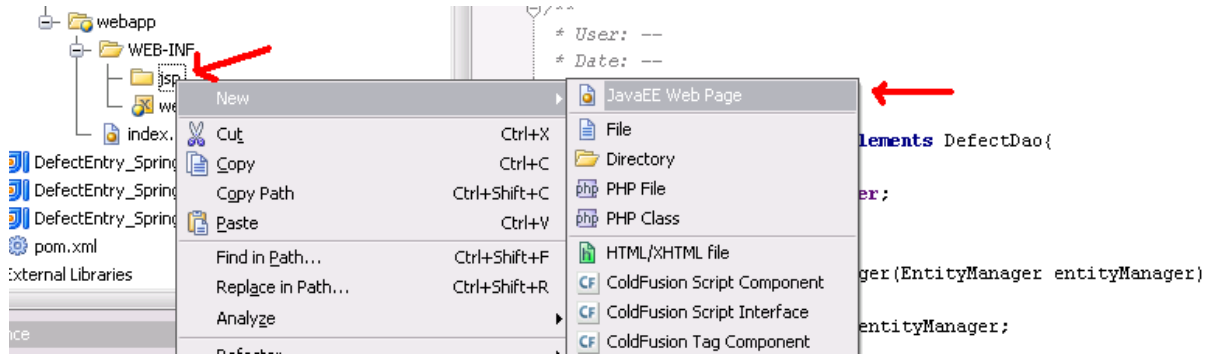
```
<bean id="datasource"
      class="com.mchange.v2.c3p0.ComboPooledDataSource"
      p:driverClass="${jdbc.driverClassName}"
      p:jdbcUrl="${jdbc.databaseurl}"
      p:user="${jdbc.username}"
      p:maxIdleTime="${jdbc.maxIdleTime}"
      p:maxPoolSize="${jdbc.maxPoolSize}"
      p:password="${jdbc.password}"
      p:preferredTestQuery="${flrs.jpa.showSql}" />

<bean id="defectService"
      class="com.karaagac.examples.defectentry.service.DefectServiceImpl">
  <property name="defectDAO" ref="defectDao" />
</bean>
<bean id="defectDao"
      class="com.karaagac.examples.defectentry.dao.DefectDaoImpl">
</bean>
```

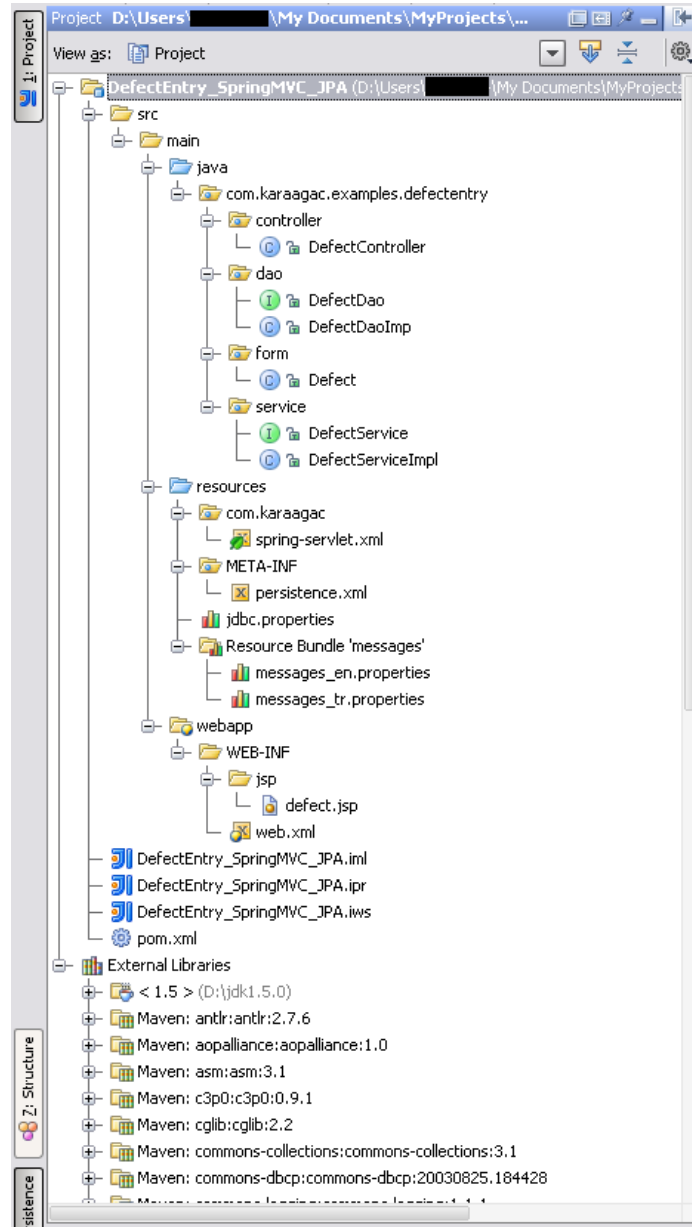
WEB-INF dosyasının altına “jsp” dizini oluşturulmalıdır. Bu dizin altında görüntülemek istenilen sayfalar yer alacaktır.



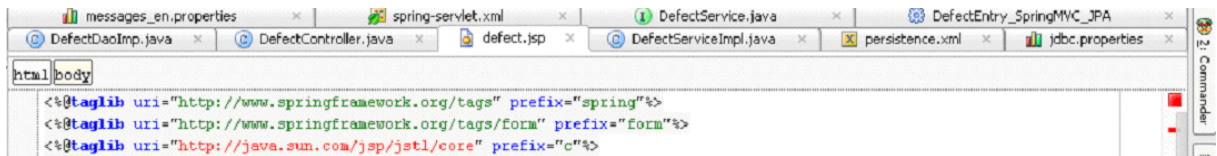
“jsp” dizininin altına defect.jsp sayfası oluşturulur.

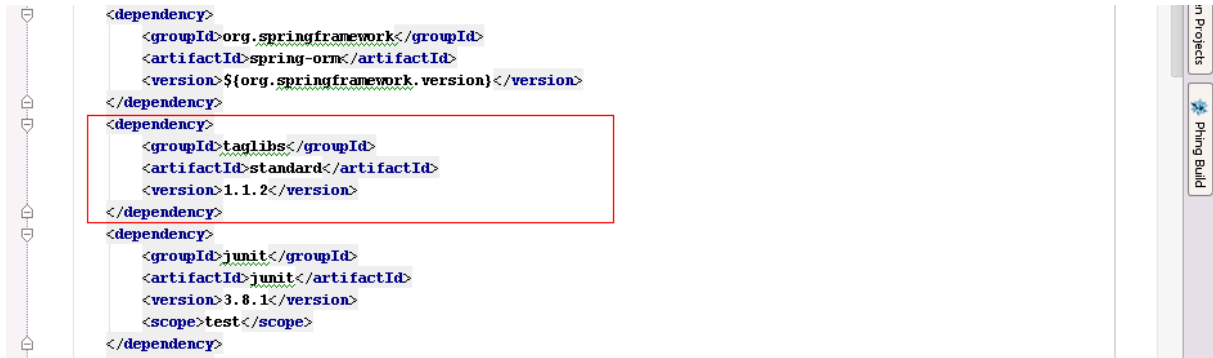


Son olarak oluşması gereken dizin yapısı aşağıda görülmektedir.



“taglibs” leri kullanabilmek için gerekli kütüphane pom.xml içerisine yazılmalıdır.

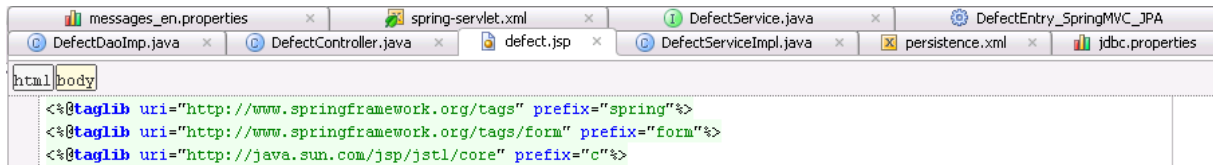




Yapılan değişikliktan sonraki ekran görüntüsü aşağıda gösterilmektedir.

<dependencies> ... </dependencies> altına aşağıdaki satırlar eklenmelidir.

```
<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
```



defect.jsp sayfasının içeriği;

```
<% @taglib uri="http://www.springframework.org/tags" prefix="spring"%>
<% @taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<% @taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

```
<!doctype html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>DefectExample</title>
  <style type="text/css">
    body {
      font-family: sans-serif;
    }
    .data, .data td {
      border-collapse: collapse;
      width: 100%;
      border: 1px solid #aaa;
      margin: 2px;
      padding: 2px;
    }
    .data th {
      font-weight: bold;
      background-color: #5C82FF;
      color: white;
    }
  </style>
```

```
</head>
<body>
```

<h2>Defect Manager</h2>

<form:form method="post" action="add" commandName="defect" >

```
    <table>
    <tr>
        <td><form:label path="defectCode"><spring:message
code="label.defectcode"/></form:label></td>
        <td><form:input path="defectCode" size="30" maxlength="20"/></td>
    </tr>
    <tr>
        <td><form:label path="defectDescription"><spring:message
code="label.defectdescription"/></form:label></td>
        <td><form:input path="defectDescription" size="90" maxlength="60" /></td>
    </tr>
    <tr>
        <td><form:label path="defectType"><spring:message
code="label.defecttype"/></form:label></td>
        <td><form:input path="defectType" size="30" maxlength="20" /></td>
    </tr>
    <tr>
        <td colspan="2">
            <input type="submit" class="textForm" value="<spring:message
code="label.adddefect"/>" />
        </td>
    </tr>
</table>
</form:form>
```

<h4 style="color: red;">\${error}</h4>

<h3>Defects</h3>

<c:if test="\${!empty defectList}">

<table class="data">

<tr>

<th>Defect Code</th>

<th>Defect Description</th>

<th>Defect Type</th>

<th> </th>

</tr>

<c:forEach items="\${defectList}" var="defect">

<tr>

<td style="width: 10%">\${defect.defectCode}</td>

<td style="width: 80%">\${defect.defectDescription}</td>

<td style="width: 10%">\${defect.defectType}</td>

<td>delete</td>

</tr>

</c:forEach>

</table>

</c:if>

</body>

</html>

pom.xml konfigürasyon sayfası;

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>Spring3HibernateMaven</groupId>
  <artifactId>Spring3HibernateMaven</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <description></description>
  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>*</include>
        </includes>
        <excludes>
          <exclude>*/*.java</exclude>
        </excludes>
      </resource>
      <resource>
        <directory>src/main/java</directory>
        <includes>
          <include>*</include>
        </includes>
        <excludes>
          <exclude>*/*.java</exclude>
        </excludes>
      </resource>
    </resources>
    <plugins>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.0</version>
      </plugin>

      <plugin>
        <groupId>org.mortbay.jetty</groupId>
        <artifactId>jetty-maven-plugin</artifactId>
        <version>7.5.4.v20111024</version>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>${org.springframework.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${org.springframework.version}</version>
    </dependency>
  </dependencies>
</project>
```

```

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${org.springframework.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${org.springframework.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>${org.springframework.version}</version>
</dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>3.5.6-Final</version>
  </dependency>
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>persistence-api</artifactId>
  <version>1.0.2</version>
</dependency>
<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.1.2</version>
</dependency>
<dependency>
  <groupId>javax.validation</groupId>
  <artifactId>validation-api</artifactId>
  <version>1.0.0.GA</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>9.1-901.jdbc3</version>
</dependency>
<dependency>
  <groupId>commons-dbcp</groupId>
  <artifactId>commons-dbcp</artifactId>
  <version>20030825.184428</version>
</dependency>
<dependency>
  <groupId>commons-pool</groupId>
  <artifactId>commons-pool</artifactId>
  <version>20030825.183949</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.5.11</version>

```



```

        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <version>1.5.11</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.5.11</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.14</version>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
    <dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>
        <version>0.9.1</version>
    </dependency>
</dependencies>
<properties>
    <org.springframework.version>3.1.0.RELEASE</org.springframework.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
</project>

```

“web.xml” dosyasının içeriği aşağıdaki gibi değiştirilmelidir.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <display-name>Example</display-name>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:com/karaagac/spring-servlet.xml</param-value>
    </context-param>

    <servlet>
        <servlet-name>spring</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:com/karaagac/spring-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

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

```

```
</servlet-mapping>
```

```
<filter>
```

```
  <filter-name>encoding-filter</filter-name>
```

```
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
```

```
  <init-param>
```

```
    <param-name>encoding</param-name>
```

```
    <param-value>ISO-8859-15</param-value>
```

```
  </init-param>
```

```
  <init-param>
```

```
    <param-name>forceEncoding</param-name>
```

```
    <param-value>true</param-value>
```

```
  </init-param>
```

```
</filter>
```

```
<filter-mapping>
```

```
  <filter-name>encoding-filter</filter-name>
```

```
  <url-pattern>/*</url-pattern>
```

```
</filter-mapping>
```

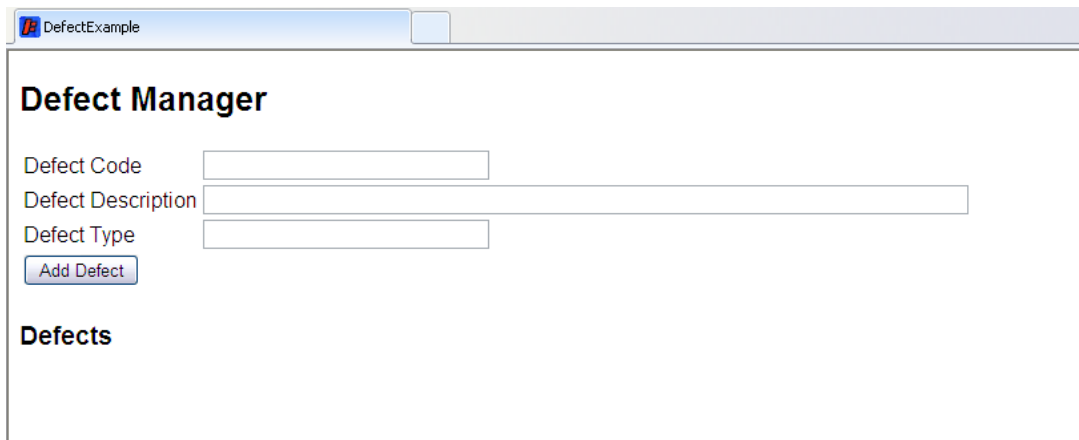
```
<listener>
```

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

```
</listener>
```

```
</web-app>
```

Tüm herşey düzgün yapılmışsa proje çalıştırıldığında aşağıdaki ekranının yüklenmesi gerekmektedir.



DefectExample

Defect Manager

Defect Code

Defect Description

Defect Type

Defects

Kayıt girildiğinde oluşan ekran aşağıda görülmektedir.

DefectExample

Defect Manager

Defect Code

Defect Description

Defect Type

Add Defect

Defects

Defect Code	Defect Description	Defect Type	
B115	Sol Kapı Çizik	S4Ç116	delete
K913	Ön Sol Sinyal Çalışmıyor	I16T41	delete

Aynı hata kodundan tekrar girildiğinde oluşan hata aşağıda görülmektedir.

Defect Manager

Hata Kodu

Hata Tanımı

Hata Tipi

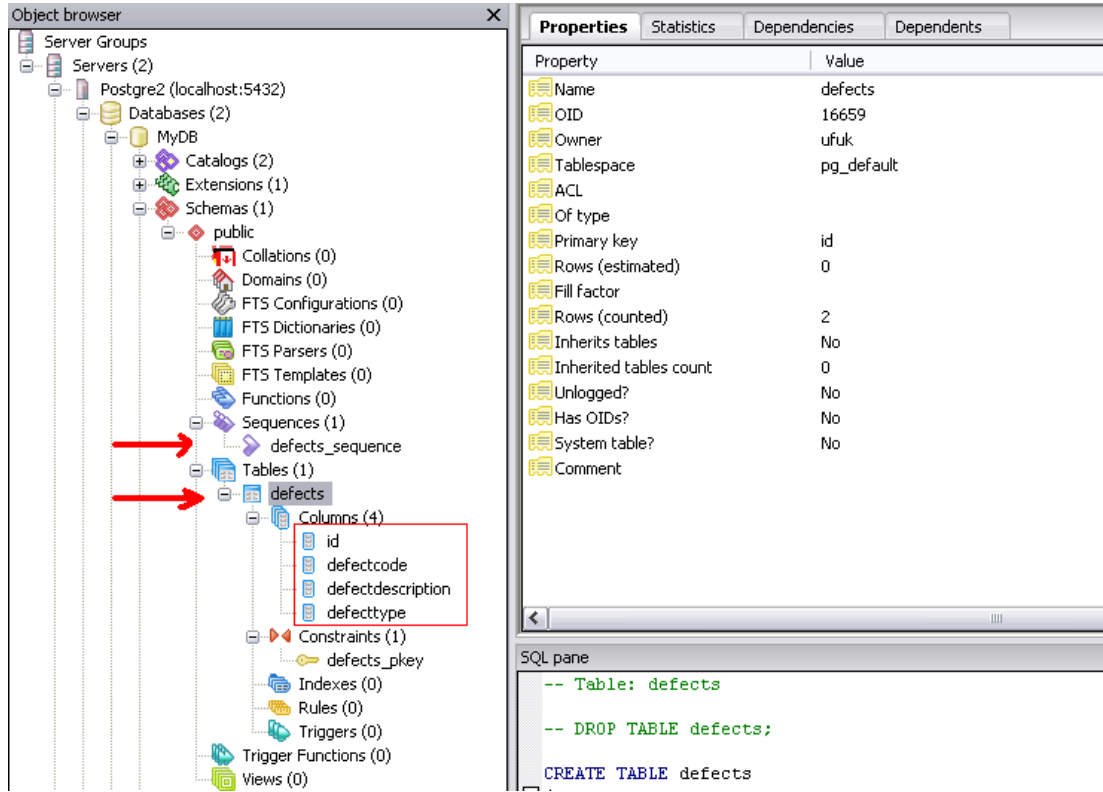
Hata Ekle

Hata verisi girilmedi! (İlgili veri zaten var.)

Defects

Defect Code	Defect Description	Defect Type	
B115	Sol Kapı Çizik	S4Ç116	delete
K913	Ön Sol Sinyal Çalışmıyor	I16T41	delete

Veri tabanında gerçekleşen değişiklikler aşağıdaki ekran görüntüsünde görülmektedir.



NOT: Uygulama çalıştığında veri tabanında ilgili tablo yoksa oluşturur, diğer durumda var olanı kullanır.

“pom.xml” içerisinde kullanılan diğer bağımlılıklar ve açıklamaları

commons-dbcp : Commons Database Connection PoolingMany Apache projects support interaction with a relational database. Creating a new connection for each user can be time consuming (often requiring multiple seconds of clock time), in order to perform a database transaction that might take milliseconds. Opening a connection per user can be unfeasible in a publicly-hosted Internet application where the number of simultaneous users can be very large. Accordingly, developers often wish to share a "pool" of open connections between all of the application's current users. The number of users actually performing a request at any given time is usually a very small percentage of the total number of active users, and during request processing is the only time that a database connection is required. The application itself logs into the DBMS, and handles any user account issues internally.

```
<dependency>
  <groupId>commons-dbcp</groupId>
  <artifactId>commons-dbcp</artifactId>
  <version>20030825.184428</version>
</dependency>
```

commons-pool : Commons Object Pooling Library Pool provides an Object-pooling API, with three major aspects: A generic object pool interface that clients and implementors can use to provide easily interchangeable pooling implementations. A toolkit for creating modular object pools. Several general purpose pool implementations.

```
<dependency>
  <groupId>commons-pool</groupId>
  <artifactId>commons-pool</artifactId>
  <version>20030825.183949</version>
</dependency>
```

org.slf4j : The Simple Logging Facade for Java or (SLF4J) serves as a simple facade or abstraction for various logging frameworks, e.g. java.util.logging, log4j and logback, allowing the end user to plug in the desired logging framework at deployment time.

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.5.11</version>
</dependency>
```

Jakarta Commons Logging (JCL)

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jcl-over-slf4j</artifactId>
  <version>1.5.11</version>
</dependency>
```

log4j : (Log for Java) Apache log4j is a Java-based logging utility.

```
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.14</version>
  <scope>runtime</scope>
</dependency>
```