Spring Security İle Veritabanı Kimlik Doğrulaması

Spring Security, **authentication** ve **authorization** sağlayarak uygulamamızın daha güvenli olmasını sağlar.

Bu güvenli sistemde **CSRF, session fixation** atakları gibi atakların önlenilmesi default olarak yer almaktadır. Ayrıca herhangi bir web uygulamasına kolayca entegre edilebilir olmasından dolayı kullanışlıdır.

Ayrıca birden çok authentication tipini destekler: In-memory, DAO, JDBC, LDAP vs.

Bazı spesifik URL'in hariç tutulması da kolaydır. Örneğin resimler, css dosyaları gibi statik dosyaları exclude edebiliriz.

Group ve role özelliklerine de sahip olduğu için kolaylıkla kullanıcıları gruplandırabiliriz.

Tüm bu özelliklerinden dolayı Spring Security modülünü öğrenmek web sitemizi daha profesyonel hale getirmemizi kolaylaştıracaktır.

Gerekli Dependency'ler

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>olyanren.java.web.telif</groupId>
    <artifactId>problem-solution</artifactId>
    <version>1.0</version>
    <packaging>war</packaging>
    <properties>

        <!-- Generic properties -->
        <java.version>1.6</java.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

        <spring-framework.version>4.1.4.RELEASE</spring-framework.version>
        <spring.security.version>3.2.3.RELEASE</spring.security.version>
        <!-- Hibernate / JPA -->
        <!-- <hibernate.version>4.3.5.Final</hibernate.version> -->
        <hibernate.version>4.1.2.Final</hibernate.version>
        <!-- Logging -->
        <logback.version>1.0.13</logback.version>
        <org.slf4j-version>1.7.5</org.slf4j-version>
        <jstl.version>1.2</jstl.version>
    </properties>

    <dependencies>
        <!-- Spring Security -->
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-web</artifactId>
            <version>${spring.security.version}</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-config</artifactId>
            <version>${spring.security.version}</version>
        </dependency>

        <!-- Core utilities used by other modules.
        Define this if you use Spring Utility APIs
        (org.springframework.core.*/org.springframework.util.*)-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>
        <!-- Bean Factory and JavaBeans utilities (depends on spring-core)
        Define this if you use Spring Bean APIs
        (org.springframework.beans.*)-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>
        <!-- Application Context
        (depends on spring-core, spring-expression, spring-aop, spring-beans)
        This is the central artifact for Spring's Dependency Injection Container
        and is generally always defined-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring-framework.version}</version>
            <exclusions>
                <!-- Exclude Commons Logging in favor of SLF4j -->
                <exclusion>
                    <groupId>commons-logging</groupId>
                    <artifactId>commons-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <!-- Web application development utilities applicable to both Servlet and
        Portlet Environments
        (depends on spring-core, spring-beans, spring-context)
        Define this if you use Spring MVC, or wish to use Struts, JSF, or another
        web framework with Spring (org.springframework.web.*)-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>
        <!-- Spring MVC for Servlet Environments
        (depends on spring-core, spring-beans, spring-context, spring-web)
        Define this if you use Spring MVC with a Servlet Container such as
        Apache Tomcat (org.springframework.web.servlet.*)-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>
        <!-- Transaction Management Abstraction
        (depends on spring-core, spring-beans, spring-aop, spring-context)
        Define this if you use Spring Transactions or DAO Exception Hierarchy
        (org.springframework.transaction.*/org.springframework.dao.*)-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>
        <!-- Object-to-Relation-Mapping (ORM) integration with Hibernate, JPA and iBatis.
        (depends on spring-core, spring-beans, spring-context, spring-tx)
        Define this if you need ORM (org.springframework.orm.*)-->
        <!-- Spring ORM support -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>
        <!-- Expression Language (depends on spring-core)
        Define this if you use Spring Expression APIs
        (org.springframework.expression.*)-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-expression</artifactId>
            <version>${spring-framework.version}</version>
        </dependency>

        <!-- Hibernate -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-entitymanager</artifactId>
            <version>${hibernate.version}</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>${hibernate.version}</version>
        </dependency>


        <!-- Hibernate library dependecy start -->
        <dependency>
            <groupId>dom4j</groupId>
            <artifactId>dom4j</artifactId>
            <version>1.6.1</version>
        </dependency>
        <!-- Logging -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${org.slf4j-version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <version>${org.slf4j-version}</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>${org.slf4j-version}</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.15</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.mail</groupId>
                    <artifactId>mail</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>javax.jms</groupId>
                    <artifactId>jms</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>com.sun.jdmk</groupId>
                    <artifactId>jmxtools</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>com.sun.jmx</groupId>
                    <artifactId>jmxri</artifactId>
                </exclusion>
            </exclusions>
            <scope>runtime</scope>
        </dependency>

        <dependency>
            <groupId>commons-collections</groupId>
            <artifactId>commons-collections</artifactId>
            <version>3.2.1</version>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.9</version>
        </dependency>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.4</version>
        </dependency>
        <!-- Servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.1</version>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>

        <!-- jstl for jsp page -->
        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>${jstl.version}</version>
        </dependency>

        <!--Hibernate validator-->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>5.1.3.Final</version>
        </dependency>


    </dependencies>
    <build>
        <plugins>

            <plugin>
                <groupId>org.mortbay.jetty</groupId>
                <artifactId>jetty-maven-plugin</artifactId>
```

```
                    <configuration>
                        <scanIntervalSeconds>1</scanIntervalSeconds>
                        <connectors>
                            <connector
implementation="org.eclipse.jetty.server.nio.SelectChannelConnector">
                                <port>9090</port>
                                <maxIdleTime>60000</maxIdleTime>
                            </connector>
                        </connectors>
                        <stopKey>foo</stopKey>
                        <stopPort>9999</stopPort>
                    </configuration>
                </plugin>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.2</version>
                    <configuration>
                        <source>1.6</source>
                        <target>1.6</target>
                        <useIncrementalCompilation>false</useIncrementalCompilation>
                    </configuration>
                </plugin>
            </plugins>
        </build>
</project>
```

web.xml Ayarları

`web.xml dosyasına Spring Security filter eklememiz gereklidir:`

```
<!-- Spring Security Filter-->
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

DelegatingFilterProxy authentication ile ilgili işlemlerin yapıldığı sınıftır.

`Session timeout ayarlaması yapalım.`

```
<session-config>
    <session-timeout>15</session-timeout>
</session-config>
```

Otomatik olarak kullanıcı 15 dakika boyunca inaktif olursa, session timeout olacaktır. Eğer timeout olmadan önce bir dialog gösterip kullanıcıya devam edip etmediğini sormak isterseniz şu linkteki javascript kodlarını kullanabilirsiniz:
http://stackoverflow.com/questions/23930541/warn-the-user-before-session-timeout

`web.xml de Spring context loader tanımlamalıyız:`

```
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/mvc-dispatcher-servlet.xml
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
```

```
</servlet>
```

DispatcherServlet, Spring MVC uygulamasının front controller'ıdır. Yani tüm web request'leri bu servlet'e gider. Bu servlet ise uygun controller'e isteği yönlendirir.
Son olarak web.xml dosyasında context param aşağıdaki gibi tanımlayalım:

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring-security.xml,/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
</context-param>
```

web.xml dosyasının tam hali aşağıdaki gibi olur:

```
<web-app id="WebApp_ID" version="2.4"
        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">

    <display-name>Spring MVC Security Application</display-name>
    <!-- Tüm Servlet ve Filter siniflarinda kullanilacak Spring Container'ı yaratir-->
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>
    <!-- Spring MVC DispatcherServlet-->
    <servlet>
        <servlet-name>mvc-dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>
                /WEB-INF/mvc-dispatcher-servlet.xml
            </param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>mvc-dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>


    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-security.xml,/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
    </context-param>
    <!-- Spring Security Filter-->
    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

spring-security.xml Dosyası

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
            xmlns:beans="http://www.springframework.org/schema/beans"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
5                 xmlns:mvc="http://www.springframework.org/schema/mvc"
6                 xsi:schemaLocation="http://www.springframework.org/schema/beans
7                  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
8                  http://www.springframework.org/schema/security
9                  http://www.springframework.org/schema/security/spring-security-3.2.xsd
10                 http://www.springframework.org/schema/mvc
11                 http://www.springframework.org/schema/mvc/spring-mvc.xsd">
12
13
14      <http pattern="/resources/**" security="none"/>
15
16      <!-- enable use-expressions -->
17      <http auto-config="true" use-expressions="true">
18
19          <intercept-url pattern="/admin/home**" access="hasRole('ROLE_ADMIN')"/>
20
21          <form-login
22             always-use-default-target="true"
23             login-processing-url="/checkUser"
24             login-page="/admin/index"
25             default-target-url="/admin/home"
26             authentication-failure-url="/admin/index?error"
27             username-parameter="userName"
28             password-parameter="password"/>
29
30          <logout logout-success-url="/admin/index?logout"/>
31          <!-- enable csrf protection -->
32          <csrf/>
33      </http>
34
35      <!-- Select users and user_roles from database -->
36      <authentication-manager>
37          <authentication-provider user-service-ref="customAdminDetailsService">
38              <password-encoder hash="plaintext">
39              </password-encoder>
40          </authentication-provider>
41      </authentication-manager>
42
43  </beans:beans>
```

14. satırda `<http pattern="/resources/**" security="none"/>` ifadesi ile **/resources** dosyasındaki **css, js gibi** dosyaların security işlemine maruz kalması önlenir.

17. satırda `<http auto-config="true" use-expressions="true">` ifadesi ile 19. satırdaki `<intercept-url>` elementindeki access attribute'sinin Spring tarafından algılanması sağlanır. Ayrıca web.xml'de aşağıdaki gibi tanımlanan **springSecurityFilterChain** bean'in yüklenmesini de sağlar. Detaylı bilgi için şu linke bakabilirsiniz : **http://stackoverflow.com/questions/6725234/whats-the-point-of-spring-mvcs-delegatingfilterproxy**

```
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

19. satırdaki `<intercept-url pattern="/admin/home**" access="hasRole('ROLE_ADMIN')"/>` elementin pattern attribute değeri, access attribute ile belirtilen **ROLE_ADMIN** erişim hakkına sahip kullanıcıların erişeceği url'yi temsil eder. `"/admin/home**"` ifadesindeki çift yıldız (**) kullanımı sayesinde `admin/home/example.jsp` veya `admin/home/example/.../example.jsp`url'ler ifade edilir. Yani alt klasörlerle de match olur. Bu tarz regex Apache Ant framework'üne aittir.

22. satırdaki `always-use-default-target="true"` ifade ile login olduktan sonra, her zaman 25. satırdaki url'ye yönlendirilmesi sağlanır.

23. satırdaki ifade ile login işleminin yapılacağı url tanımlanır. Eğer bu ifade kullanılmazsa default değer **j_spring_security_check**

kullanılır. Bunun dezavantajı şudur: Bu değer login.jsp sayfasında form elementinin action attribute'sinde kullanılacağı için client sayfanın kaynak kodunda bu değeri görür. Bu ise spring framework kullandığımızı ortaya çıkarır. Hangi teknolojinin kullanıldığını kullanıcıdan saklamak güvenlik önlemidir.

24. satırdaki `login-page` attribute'sinin aldığı değer bizim login formumuzun olduğu jsp dosyasıdır. jsp uzantısının yazılmasına gerek olmadığı için yazılmamıştır.

26. satırdaki ifade ile giriş başarısız olduğu zaman **error** parametresi ile hata mesajını yazdırabiliriz.

27. satırdaki **username-parameter** attribute'sinin aldığı değer 23. satırdaki mantıkla aynıdır. Eğer bu parametre kullanılmasaydı default değer **j_username** kullanılırdı. Yine bu sayede kullanıcı Spring framework'u kullandığımızı anlayabilecekti. Aynı şekilde 29. satırdaki attribute kullanılmasaydı default değer **j_password** kullanılırdı.

30. satırda **logout** yapıldığı zaman gidilecek url belirtilir.

32. satırda `<csrf>` aktif edilmektedir. Bu konu ile ilgili bilgiye **http://docs.spring.io/spring-security/site/docs/3.2.0.CI-SNAPSHOT/reference/html/csrf.html** adresinden ulaşabilirsiniz.

36. satırdaki **customAdminDetailsService** bean aşağıdaki gibi tanımlanmıştır:

```java
@Service
@Transactional(readOnly = true)
public class CustomAdminDetailsService implements UserDetailsService {

     @Autowired
     private AdminDao adminDAO;

     public UserDetails loadUserByUsername(String login)
     throws UsernameNotFoundException {

         Admin domainAdmin = adminDAO.getAdmin(login);

         boolean enabled = true;
         boolean accountNonExpired = true;
         boolean credentialsNonExpired = true;
         boolean accountNonLocked = true;

         return new User(
         domainAdmin.getUserName(),
         domainAdmin.getPassword(),
         enabled,
         accountNonExpired,
         credentialsNonExpired,
         accountNonLocked,
         getAuthorities(domainAdmin.getAdminRole().getAdminRoleId())
         );
     }

     public Collection<? extends GrantedAuthority> getAuthorities(Integer role) {
         List<GrantedAuthority> authList = getGrantedAuthorities(getRoles(role));
         return authList;
     }

     public List<String> getRoles(Integer role) {

         List<String> roles = new ArrayList<String>();

         if (role.intValue() == 1) {
             roles.add("ROLE_MODERATOR");
             roles.add("ROLE_ADMIN");
         } else if (role.intValue() == 2) {
             roles.add("ROLE_MODERATOR");
         }
         return roles;
     }

```

```java
    public static List<GrantedAuthority> getGrantedAuthorities(List<String> roles) {
        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();

        for (String role : roles) {
            authorities.add(new SimpleGrantedAuthority(role));
        }
        return authorities;
    }

}
```

Controller Sınıfı

```java
@Controller
@RequestMapping(value = "/admin", method = RequestMethod.GET)
public class AdminController {
    @Autowired(required = true)
    private ProblemService problemService;
    @Autowired(required = true)
    private AdminService adminService;

    @RequestMapping(value = "/index", method = RequestMethod.POST)
    public ModelAndView login(){
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        ModelAndView modelAndView;
        if (!(auth instanceof AnonymousAuthenticationToken)) {
            modelAndView=new ModelAndView("redirect:/admin/home");
            modelAndView.addObject("problem",new Problem());
            return modelAndView;
        }else{
            modelAndView=new ModelAndView("/admin/index");
        }
        modelAndView.addObject("problem",new Problem());
        return modelAndView;
    }

    @RequestMapping(value = "/home", method = RequestMethod.POST)
    public String home(@ModelAttribute("problem") Problem problem) {
        if (problem != null && problem.getContent() != null) {
            problem.setCreationDate(new Date());
            problemService.saveOrUpdate(problem);
        }

        return "admin/home/index"
    }
}
```

mvc-dispatcher-servlet.xml Dosyası

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:tx="http://www.springframework.org/schema/tx"
      xmlns:mvc="http://www.springframework.org/schema/mvc"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context-3.0.xsd
       http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc.xsd">

```

```xml
    <context:component-scan base-package="olyanren"/>
    <tx:annotation-driven transaction-manager="transactionManager" />
    <!--MVC resources annotasyonu ile mvc annotation-driven birlikte kullanilir
    amaci statik dosyalarin dizinini belirtmektir-->
    <mvc:resources mapping="/resources/**" location="/resources/"/>
    <mvc:annotation-driven />
    <bean id="transactionManager"
         class="org.springframework.orm.hibernate4.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory"/>
    </bean>

    <bean id="sessionFactory"
         class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="packagesToScan" value="olyanren.java.web.telif.problemsolution.model"/>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
                <prop key="hibernate.connection.CharSet">utf8</prop>
                <prop key="hibernate.connection.characterEncoding">utf8</prop>
                <prop key="hibernate.connection.useUnicode">true</prop>
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>

    </bean>
    <mvc:interceptors>
        <bean id="webContentInterceptor"
class="org.springframework.web.servlet.mvc.WebContentInterceptor">
            <property name="cacheSeconds" value="0"/>
            <property name="useExpiresHeader" value="false"/>
            <property name="useCacheControlHeader" value="true"/>
            <property name="useCacheControlNoStore" value="true"/>
        </bean>
    </mvc:interceptors>
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
         destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url"
                  value="jdbc:mysql://localhost:3306/SpringSecurityTest?autoReconnect=true&
                   useUnicode=true&createDatabaseIfNotExist=true&characterEncoding=utf-8"/>
        <property name="username" value="root"/>
        <property name="password" value="12"/>

    </bean>

    <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"/>
        <property name="suffix" value=".jsp"/>
    </bean>
</beans>
```

Bu config dosyasında veritabanı bağlantısı, transaction manager ve temel ayarlar tanımlanmıştır.

Login işleminden sonra tarayıcının geri tuşuna basıldığında tekrar login sayfasına yönlendirilmemesi için ve logout olduktan sonra, önceki sayfaya geri dönmesini engellemek için **webContentInterceptor** bean kullanılır.

Sonuç

Bu yazımızda Spring Security'inin web uygulamasına nasıl entegre edildiğinden bahsedilmiştir. Bu yazıda sadece veritabanı kullanılarak login işlemi anlatıldı. Spring Security veritabanı ile login işleminin yanısıra in-memory, DAO, LDAP gibi farklı authentication tiplerini de destekler. Bu özellikleri ile Spring Security, Spring framework'ünün en yararlı modüllerinden birisidir.