Spring Security in JUnit Test

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"classpath:/applicationContext.xml",
        "classpath:/mvc-dispatcher-servlet.xml",
        "classpath:/spring-security.xml"})
@WebAppConfiguration
public class UserControllerTest {

    @Autowired
    private UserService userService;
    @Autowired
    WebApplicationContext wac;
    private MockMvc mockMvc;

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);
        this.mockMvc= MockMvcBuilders.webAppContextSetup(wac).apply(springSecurity()).build();
    }

    @Test
    @WithMockUser(username = "myuce")
    public void saveBelge() throws Exception {
        Belge belge = new GumrukServiceUtilTest().createBelge();
        mockMvc.perform(post("/kullanici/belge/kaydet")
                .flashAttr("belge", belge))
                .andExpect(status().isOk());
    }

    @Test
    public void register() throws Exception {
        User user = createUser();
        saveUser(user);
        assertEquals(user.getUsername(), getUserFromDB(user.getUsername()).getUsername());
    }

    private User createUser() {
        User user = new User();
        user.setUsername(RandomStringUtils.random(4, true, false));
        user.setTitle(RandomStringUtils.random(20, true, false));
        user.setName(RandomStringUtils.random(5, true, false));
        user.setSurname(RandomStringUtils.random(5, true, false));
        user.setPassword(RandomStringUtils.random(8, false, true));
        user.setTelephone(RandomStringUtils.random(10, false, true));
        user.setEmail(RandomStringUtils.random(4, true, false) + "@telifhaklari.gov.tr");
        user.setAddress(RandomStringUtils.random(50, true, true));
        return user;
    }

    private void saveUser(User user) throws Exception {
        mockMvc.perform(post("/kayit")
                .param("captcha", "10")
                .with(csrf())
                .sessionAttrs(getCaptcha())
                .flashAttr("user", user))
                .andExpect(status().isOk());
    }


    private Map<String, Object> getCaptcha() {
        Map<String, Object> sessionAttrs = new HashMap<String, Object>();
        sessionAttrs.put("rand1", "5");
```

```
62          sessionAttrs.put("rand2", "5");
63          return sessionAttrs;
64      }
65
66      private User getUserFromDB(String username) {
67          return userService.getUser(username);
68      }
69
70      @Test
71      public void saveAdmin() throws Exception {
72          User user = createUser();
73          UserRole userRole=new UserRole();
74          userRole.setUserRoleId(1);
75          user.setUserRoleList(Arrays.asList(userRole));
76          saveAdmin(user);
77          assertEquals(user.getUsername(), getUserFromDB(user.getUsername()).getUsername());
78      }
79
80      private void saveAdmin(User user) throws Exception {
81          mockMvc.perform(post("/yonetim/home/yonetici/ekle/kaydet")
82                  .with(csrf())
83                  .session((MockHttpSession) getHttpSession("SaFv", "14093171"))
84                  .flashAttr("user", user))
85                  .andExpect(status().isOk());
86      }
87
88      @Test
89      public void passwordRemind() throws Exception {
90          User user = createUser();
91          saveUser(user);
92          mockMvc.perform(post("/sifre-hatirlatma")
93                  .param("username", user.getUsername())
94                  .param("email", user.getEmail())
95                  .param("captcha", "10")
96                  .with(csrf())
97                  .sessionAttrs(getCaptcha()))
98                  .andExpect(status().isOk());
99      }
100
101     @Test
102     public void updateUserInfo() throws Exception {
103         User user = createUser();
104         saveUser(user);
105         String username = user.getUsername();
106         String password = user.getPassword();
107         User replacedUserInfo = createUser();
108         user.setName(replacedUserInfo.getName());
109         user.setPassword(replacedUserInfo.getPassword());
110         user.setTelephone(replacedUserInfo.getTelephone());
111         user.setEmail(replacedUserInfo.getEmail());
112         mockMvc.perform(post("/kullanici/bilgilerim/kaydet")
113                 .flashAttr("user", user)
114                 .session((MockHttpSession) getHttpSession(username, password))
115                 .with(csrf()))
116                 .andExpect(status().isOk());
117     }
118
119     @Test
120     public void bilgilerim() throws Exception {
121         User user = createUser();
122         saveUser(user);
123         mockMvc.perform(get("/kullanici/bilgilerim")
124                 .session((MockHttpSession) getHttpSession(user.getUsername(),
125                         user.getPassword())))
126                 .andExpect(status().isOk());
127     }
128
```

```
    private HttpSession getHttpSession(String username, String password) throws Exception {
        return mockMvc.perform(post("/login")
                .param("username", username)
                .param("password", password)
                .param("captcha", "10")
                .with(csrf())
                .sessionAttrs(getCaptcha()))
                .andDo(print())
                .andExpect(status().is(HttpStatus.FOUND.value()))
                .andReturn()
                .getRequest()
                .getSession();
    }

    @Test
    public void changePassword() throws Exception {
        User user = createUser();
        saveUser(user);
        String newPassword = createUser().getPassword();
        mockMvc.perform(post("/kullanici/sifre-degistirme")
                .param("oldPassword", user.getPassword())
                .param("newPassword", newPassword)
                .param("retypedNewPassword", newPassword)
                .session((MockHttpSession) getHttpSession(user.getUsername(),
user.getPassword()))
                .with(csrf()))
                .andDo(print())
                .andExpect(status().isOk());
    }
}
```