# Tapestry5AcegiNoAnnotations

Tapestry 5 and Acegi (Spring Security) can work together without a Tapestry-Acegi integration library.

- Acegi acts as a filter, like Tapestry.
- You Tapestry pages don't have to know very much about Acegi in order to use it

Here is a sample web.xml. Notice the ordering of the filters. Also note the two XML files I reference here. I chose to break up the main acegi document and the acegi datasource into different files. An Acegi configuration can get messy quickly, especially for newcomers who don't know exactly what options they need.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
        <display-name>MyApp</display-name>

        <description>Installation Executive Information System</description>

        <context-param>
                <!-- The only significant configuration for Tapestry 5, this informs Tapestry
                        of where to look for pages, components and mixins. -->
                <param-name>tapestry.app-package</param-name>
                <param-value>com.mycompany.myapp.wui.tapestry</param-value>
                <description>
                        This is where to look for pages, components and mixins
                </description>
        </context-param>

        <!--  spring bean definition -->
        <context-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>classpath*:applicationContext-acegisecurity.xml,classpath*:applicationContext-
datasource.xml</param-value>

                <description>spring bean definition</description>
        </context-param>

        <!-- filters stack -->
        <!-- notice Acegi comes before Tapestry -->
        <!-- ***************** Acegi Filter **********************-->
        <filter>
                <filter-name>Acegi Filter Chain Proxy</filter-name>
                <filter-class>
                        org.acegisecurity.util.FilterToBeanProxy
                </filter-class>
                <init-param>
                        <param-name>targetBean</param-name>
                        <param-value>filterChainProxy</param-value>
                </init-param>
        </filter>

        <!-- tapestry filter -->
        <filter>
                <filter-name>app</filter-name>
                <filter-class>org.apache.tapestry.TapestryFilter</filter-class>
        </filter>

        <filter-mapping>
                <filter-name>Acegi Filter Chain Proxy</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>


        <!-- Tapestry filter mapping -->
        <filter-mapping>
                <filter-name>app</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>
```

```
        <!--  Spring listener -->
        <listener>
                <listener-class>
                        org.acegisecurity.ui.session.HttpSessionEventPublisher
                </listener-class>
        </listener>
        <listener>
                <listener-class>
                        org.springframework.web.context.ContextLoaderListener
                </listener-class>
        </listener>


        <!--  Session Configuration -->
        <!--  NOTE:must come before JNDI references -->
        <session-config>
                <session-timeout>30</session-timeout>
        </session-config>

        <error-page>
                <error-code>403</error-code>
                <location>/AccessDenied</location>
        </error-page>
        <error-page>
                <error-code>404</error-code>
                <location>/Start</location>
        </error-page>

        <!--  JNDI References -->
        <resource-ref>
                <description>A datasource such as a database.</description>
                <res-ref-name>jdbc/MyDatasource</res-ref-name>
                <res-type>javax.sql.DataSource</res-type>
                <res-auth>Container</res-auth>
        </resource-ref>
</web-app>
```

Here is applicationContext-acegisecurity.xml (I chose this name, it can be whatever you load in web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema
/beans/spring-beans-2.0.xsd"
        default-autowire="no">


        <!-- ======================== FILTER CHAIN ======================= -->
        <!--  Note: I got rid of the 'rememberMe' and 'switchUser' filters you see in a lot of examples.  They
were confusing the debugging, and now that I understand what's going on, they'll be easy to add in later if I
need them -->
        <!-- exceptionTranslationFilter, after anonymous -->
        <bean id="filterChainProxy"
                class="org.acegisecurity.util.FilterChainProxy">
                <property name="filterInvocationDefinitionSource">
                        <value>
                                CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
                                PATTERN_TYPE_APACHE_ANT
                                /**=httpSessionContextIntegrationFilter,logoutFilter,
SSOauthenticationProcessingFilter,anonymousProcessingFilter,exceptionTranslationFilter,
filterInvocationInterceptor
                        </value>
                </property>
        </bean>
```

```
        <!-- ======================== AUTHENTICATION ======================== -->
        <!-- ======================== AUTHENTICATION Manager ======================== -->

        <bean id="authenticationManager"
                class="org.acegisecurity.providers.ProviderManager">
                <property name="providers">
                        <list>
                                <ref local="SiteminderAuthenticationProvider" />
                                <!-- <ref local="anonymousAuthenticationProvider" />-->
                        </list>
                </property>
        </bean>

        <!-- ======================== AUTHENTICATION Providers ======================== -->

        <bean id="SiteminderAuthenticationProvider"
                class="org.acegisecurity.providers.siteminder.SiteminderAuthenticationProvider">
                <property name="hideUserNotFoundExceptions" value="false" />
                <property name="userDetailsService">
                        <ref bean="userDetailsServiceSSO" />
                </property>
                <property name="userCache">
                        <bean
                                class="org.acegisecurity.providers.dao.cache.EhCacheBasedUserCache">
                                <property name="cache">
                                        <bean
                                                class="org.springframework.cache.ehcache.EhCacheFactoryBean">
                                                <property name="cacheManager">
                                                        <bean
                                                                class="org.springframework.cache.ehcache.
EhCacheManagerFactoryBean" />
                                                </property>
                                                <property name="cacheName" value="userCache" />
                                        </bean>
                                </property>
                        </bean>
                </property>
        </bean>

        <!-- Automatically receives AuthenticationEvent messages -->
        <bean id="loggerListener"
                class="org.acegisecurity.event.authentication.LoggerListener" />

        <bean id="basicProcessingFilter"
                class="org.acegisecurity.ui.basicauth.BasicProcessingFilter">
                <property name="authenticationManager">
                        <ref local="authenticationManager" />
                </property>
                <property name="authenticationEntryPoint">
                        <ref local="basicProcessingFilterEntryPoint" />
                </property>
        </bean>

        <!-- Essentially Unused unless you're using Basic Authentication, which we're not -->
        <bean id="basicProcessingFilterEntryPoint"
                class="org.acegisecurity.ui.basicauth.BasicProcessingFilterEntryPoint">
                <property name="realmName">
                        <value>Service Entry Realm</value>
                </property>
        </bean>



        <bean id="anonymousAuthenticationProvider"
                class="org.acegisecurity.providers.anonymous.AnonymousAuthenticationProvider">
                <property name="key">
                        <value>foobar</value>
                </property>
        </bean>

        <bean id="httpSessionContextIntegrationFilter"
```

```
                class="org.acegisecurity.context.HttpSessionContextIntegrationFilter">
        </bean>


        <!-- ==================== HTTP REQUEST SECURITY ==================== -->
        <bean id="anonymousProcessingFilter"
                class="org.acegisecurity.providers.anonymous.AnonymousProcessingFilter">
                <property name="key">
                        <value>foobar</value>
                </property>
                <property name="userAttribute">
                        <value>anonymousUser,ROLE_ANONYMOUS</value>
                </property>
        </bean>

        <bean id="exceptionTranslationFilter"
                class="org.acegisecurity.ui.ExceptionTranslationFilter">
                <property name="authenticationEntryPoint">
                        <ref bean="formLoginAuthenticationEntryPoint" />
                </property>
                <property name="accessDeniedHandler">
                        <bean
                                class="org.acegisecurity.ui.AccessDeniedHandlerImpl">
                                <property name="errorPage" value="/AccessDenied" />
                        </bean>
                </property>
        </bean>

        <bean id="formLoginAuthenticationEntryPoint"
                class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilterEntryPoint">
                <property name="loginFormUrl" value="/authentication/LoginPage" />
                <property name="forceHttps" value="false" />
        </bean>

        <!-- ==================== Not using Siteminder ==================== -->

        <bean id="authenticationProcessingFilter"
                class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilter">
                <property name="authenticationManager"
                        ref="authenticationManager" />
                <property name="authenticationFailureUrl"
                        value="/authentication/LoginFailure" />
                <property name="defaultTargetUrl" value="/Start" />
                <property name="filterProcessesUrl"
                        value="/j_acegi_security_check" />
        </bean>

        <!--         <property name="formUsernameParameterKey" value="j_username" />
                <property name="siteminderPasswordHeaderKey" value="SM_USER" />
        -->
        <!-- ==================== USING SITEMINDER ==================== -->
        <bean id="SSOauthenticationProcessingFilter"
                class="org.acegisecurity.ui.webapp.SiteminderAuthenticationProcessingFilter">
                <!--
NOTE: I _think_ this is where you can intercept the filter, for stripping form variables from the request
<bean id="SSOauthenticationProcessingFilter"
                        class="com.mycompany.myapp.filter.MyAppSiteminderAuthenticationProcessingFilter">-->
                <property name="authenticationManager"
                        ref="authenticationManager" />
                <property name="authenticationFailureUrl"
                        value="/authentication/LoginFailure" />
                <property name="defaultTargetUrl" value="/Start" />
                <property name="filterProcessesUrl"
                        value="/j_acegi_security_check" />
                <property name="siteminderUsernameHeaderKey" value="smuser" />
                <property name="siteminderPasswordHeaderKey" value="smuser" />

        </bean>


        <bean id="httpRequestAccessDecisionManager"
```

```
                class="org.acegisecurity.vote.AffirmativeBased">
                <property name="allowIfAllAbstainDecisions">
                        <value>false</value>
                </property>
                <property name="decisionVoters">
                        <list>
                                <ref bean="roleVoter" />
                        </list>
                </property>
        </bean>

        <!-- An access decision voter that reads ROLE_* configuration settings -->
        <bean id="roleVoter" class="org.acegisecurity.vote.RoleVoter" />


        <!-- Note the order that entries are placed against the objectDefinitionSource is critical.
                The FilterSecurityInterceptor will work from the top of the list down to the FIRST pattern that
matches the request URL.
                Accordingly, you should place MOST SPECIFIC (ie a/b/c/d.*) expressions first, with LEAST
SPECIFIC (ie a/.*) expressions last -->
        <bean id="filterInvocationInterceptor"
                class="org.acegisecurity.intercept.web.FilterSecurityInterceptor">
                <property name="authenticationManager">
                        <ref bean="authenticationManager" />
                </property>
                <property name="accessDecisionManager">
                        <ref bean="httpRequestAccessDecisionManager" />
                </property>
                <property name="objectDefinitionSource">
                        <value>
                                CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
                                PATTERN_TYPE_APACHE_ANT
                                /**/*.js=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /**/*.jpg=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /**/*.gif=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /**/*.png=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /**/*.htc=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /**/*.ico=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /**/*.css=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /css/**/=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /images/**=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /js/**/=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /layout/**=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /authentication/**=ROLE_ANONYMOUS,ROLE_ADMIN,ROLE_STANDARD
                                /**/*setuppage=ROLE_STANDARD,ROLE_ADMIN
                                /**/*reportpage=ROLE_STANDARD,ROLE_ADMIN
                                /setup/**=ROLE_STANDARD,ROLE_ADMIN
                                /reports/**=ROLE_STANDARD,ROLE_ADMIN
                                /authentication/loginsuccess=ROLE_ADMIN,ROLE_STANDARD
                                /**=ROLE_ADMIN,ROLE_STANDARD
                        </value>
                </property>
        </bean>

        <!--
                <bean id="userDetailsService" class="com.mycompany.myapp.accounts.util.security.
AuthenticationSSOImpl" />
        -->
        <bean id="userDetailsServiceSSO"
                class="com.mycompany.myapp.accounts.util.security.AuthenticationJdbcDaoSSOImpl">
                <property name="dataSource">
                        <ref bean="dataSource" />
                </property>
                <property name="userInfoObjectTypes">
                        <list>
                                <value>Admin</value>
                                <value>Standard</value>
                        </list>
                </property>
        </bean>
```

```
        <bean id="logoutFilter"
                class="org.acegisecurity.ui.logout.LogoutFilter">
                <constructor-arg value="/authentication/LogoutSuccess" />
                <!-- URL redirected to after logout -->
                <constructor-arg>
                        <list>
                                <!--<ref bean="rememberMeServices"/>-->
                                <bean
                                        class="org.acegisecurity.ui.logout.SecurityContextLogoutHandler" />
                        </list>
                </constructor-arg>
        </bean>
</beans>
```

Note in the above file that these values are unique to my application:

```
...
                <property name="siteminderUsernameHeaderKey" value="smuser" />
                <property name="siteminderPasswordHeaderKey" value="smuser" />
...
```

What you are telling Acegi is the name of the field in the header. It might be just "username" and "password", or it could be something like "myorgid", etc. It depends on what application is putting the credentials in the header. Note that both values for mine are smuser. I'm doing it this way because in my application, I never see the user's password if the user is coming in through a SSO session, I only get the user id. This is probably not a good idea for a final release.

Here is applicationContext-acegidatasource.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE beans PUBLIC
    "-//SPRING//DTD BEAN//EN"
    "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
        <bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
        <property name="resourceRef">
                <value>true</value>
        </property>
        <property name="jndiName">
                <value>java:comp/env/jdbc/IEISMeta</value>
        </property>
        </bean>
</beans>
```

Here I just reference the JNDI name that is defined in your web.xml and context.xml. If you are just getting started, replace this with "in memory dao" given in the Acegi framework samples.

Here is LoginComponent.java. The interesting stuff is in getLink().

```
package com.mycompany.myapp.wui.tapestry.components.authentication;

import com.mycompany.myapp.accounts.util.security.CustomUser;
import com.mycompany.myapp.model.Constants;

import org.apache.tapestry.Link;
import org.apache.tapestry.MarkupWriter;
import org.apache.tapestry.annotations.ApplicationState;
import org.apache.tapestry.annotations.Component;
import org.apache.tapestry.annotations.PageAttached;
import org.apache.tapestry.annotations.Retain;
import org.apache.tapestry.corelib.components.Form;
import org.apache.tapestry.corelib.components.PasswordField;
import org.apache.tapestry.corelib.components.TextField;
import org.apache.tapestry.internal.services.LinkImpl;
```

```
import org.apache.tapestry.ioc.Messages;
import org.apache.tapestry.ioc.annotations.Inject;
import org.apache.tapestry.services.Request;
import org.apache.tapestry.services.RequestGlobals;
import org.apache.tapestry.services.Response;
import org.slf4j.Logger;

public class LoginComponent {

	@Inject
	private Logger logger;
	@Inject
	private RequestGlobals _requestGlobals;

	@Component(id = "Login")
	private Form form;

	@Inject
	private Messages messages;


	// Normally not retained, just want to prove that the output value is always
	// the blank string.
	@Retain
	private String password;

	@SuppressWarnings("unused")
	@Component(id = "password")
	private PasswordField passwordField;

	@Inject
	private Request request;

	@Inject
	private Response response;

	@ApplicationState
	private CustomUser user;

	private boolean userExists;

	private String username;

	@SuppressWarnings("unused")
	@Component(id = "username")
	private TextField usernameField;

	private void DiscardOldSession() {
		if (userExists) {
			user = null;
		}
	}

	private Link getLink() {
		logger.info("request.getContextPath() " + request.getContextPath());
		Link link = new LinkImpl(response, request.getContextPath(), Constants.J_ACEGI_SECURITY_CHECK);
		link.addParameter(Constants.J_USERNAME, username);
		link.addParameter(Constants.J_PASSWORD, password);
		return link;
	}

	public Messages getMessages() {
		return this.messages;
	}

	/**
	 * @return the password
	 */
	public String getPassword() {
		return password;
	}
```

```
        /**
         * @return the user
         */
        public CustomUser getUser() {
                return user;
        }

        /**
         * @return the username
         */
        public String getUsername() {
                return username;
        }
        /**
         * @return the userExists
         */
        public boolean isUserExists() {
                return userExists;
        }

        public Link onSuccessFromLogin() {
                DiscardOldSession();
                return getLink();
        }

        @PageAttached
        void pageAttached(MarkupWriter writer) {
                String errorParamAsString = _requestGlobals.getRequest().getParameter("error");
                boolean error = Boolean.parseBoolean(errorParamAsString);
                if (error) {
                        form.recordError(messages.format("login-error-message", new Object[] {}));
                }
        }

        /**
         * @param password
         *            the password to set
         */
        public void setPassword(String password) {
                this.password = password;
        }

        /**
         * @param username
         *            the username to set
         */
        public void setUsername(String username) {
                this.username = username;
        }
}
```

Notes about this example:

My application has Single Sign On (SSO) handled by another web server, using SiteMinder. By the time users get to my application in production, a user has already logged in. This puts a username value into the request header, and my application sniffs for that on the login page (not the login component)

If the login page sees the username in the header, and the user exists in the datasource, the user gets redirected to the start page.
If the login page does not see the username field in the header, or the username in the header is not in the datasource, the Login Page renders the Login Component (the component has the form on it, and requests an Acegi link when you click the login button)

So the form authentication is a back up in case SSO has failed or is disabled (such as on a development machine). In production you would normally disable the form authentication (unless you need a backdoor) and rely solely on the SSO to provide the username.