

JSF & Spring

Integrate Spring & JSF

- Vytvořte WEB-INF/faces-config.xml:

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

<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                                   http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
               version="2.2">

    <application>

        <el-resolver>org.springframework.web.jsf.el.SpringBeanFacesELResolver</el-resolver>

    </application>

</faces-config>
```

Managed Bean

- Managed Bean poté vypadá následovně:

@Named

@Scope("request")

public class ItemController {

@Inject

private ItemService itemService;

}

Také je možné použít @Component, ale pak JBoss Tools takovou beanu nenabízí uvnitř XHTML souboru.

Také je možné použít @Autowired

Nezapomenout na nastavení scope, jinak bude bean standardně singleton!

Poznámka: Pro podporu těchto anotací mimo aplikační server přidejte do pom.xml:

```
<dependency>
  <groupId>javax.inject</groupId>
  <artifactId>javax.inject</artifactId>
  <version>1</version>
</dependency>
```

View scope I.

- View scope je velice důležitý a často používaný, ale Spring takový scope nemá. Jak ho vytvořit?

```
import java.util.Map;
```

```
import javax.faces.context.FacesContext;
```

```
import org.springframework.beans.factory.ObjectFactory;
```

```
import org.springframework.beans.factory.config.Scope;
```

```
public class ViewScope implements Scope {
```

View scope II.

```
public Object get(String name,  
    @SuppressWarnings("rawtypes") ObjectFactory objectFactory) {  
    Map<String, Object> viewMap = getViewMap();  
    Object bean = viewMap.get(name);  
    if (bean == null) {  
        bean = objectFactory.getObject();  
        viewMap.put(name, bean);  
    }  
    return bean;  
}
```

View scope III.

```
public Object remove(String name) {  
    Map<String, Object> viewMap = getViewMap();  
    Object bean = viewMap.get(name);  
    if (bean != null) {  
        viewMap.remove(name);  
    }  
    return bean;  
}  
  
public void registerDestructionCallback(String name, Runnable callback) {  
}
```

View scope IV.

```
public Object resolveContextualObject(String key) {  
    return null;  
}
```

```
private Map<String, Object> getViewMap() {  
    return FacesContext.getCurrentInstance().getViewRoot().getViewMap();  
}
```

```
public String getConversationId() {  
    return null;  
}
```

```
}
```

Zapojení ViewScope

- Přidejte do root contextu:

```
<bean class="org.springframework.beans.factory.config.CustomScopeConfigurer">
  <property name="scopes">
    <map>
      <entry key="view">
        <bean class="com.java.vlog.service.ViewScope" />
      </entry>
    </map>
  </property>
</bean>
```


Další typy integrace

- Existují i další typy integrace:
 - <http://stackoverflow.com/questions/24320109/how-to-integrate-spring-and-jsf>

Spring Security & JSF

pom.xml I.

- Přidejte do pom.xml:

```
<properties>

    <spring.security.version>3.2.4.RELEASE</spring.security.version>

</properties>

<dependencies>

    <dependency>

        <groupId>org.springframework.security</groupId>

        <artifactId>spring-security-web</artifactId>

        <version>${spring.security.version}</version>

    </dependency>
```

- Pozor! Tyto různé knihovny závisí na různých knihovnách (a verzích) Spring frameworku. Dbejte na to, abyste měli v classpath od celého Spring frameworku stejné verze!!!

pom.xml II.

```
<dependency>
```

```
    <groupId>org.springframework.security</groupId>
```

```
    <artifactId>spring-security-config</artifactId>
```

```
    <version>${spring.security.version}</version>
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>org.springframework.security</groupId>
```

```
    <artifactId>spring-security-taglibs</artifactId>
```

```
    <version>${spring.security.version}</version>
```

```
</dependency>
```

pom.xml III.

```
<dependency>
```

```
  <groupId>org.springframework.webflow</groupId>
```

```
  <artifactId>spring-faces</artifactId>
```

```
  <version>2.4.0.RELEASE</version>
```

```
  <exclusions>
```

```
    <exclusion>
```

```
      <artifactId>spring-js</artifactId>
```

```
      <groupId>org.springframework.webflow</groupId>
```

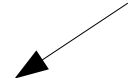
```
    </exclusion>
```

```
  </exclusions>
```

```
</dependency>
```

```
</dependencies>
```

Toto není zapotřebí



web.xml

- Přidejte do web.xml:

```
<context-param>
```

```
    <param-name>javax.faces.FACELETS_LIBRARIES</param-name>
```

```
    <param-value>/WEB-INF/springsecurity.taglib.xml</param-value>
```

```
</context-param>
```

```
<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>
```

Spring security taglibs soubor I.

- Vytvořte soubor WEB-INF/springsecurity.taglib.xml:

```
<?xml version="1.0"?>

<!DOCTYPE facelet-taglib PUBLIC

"-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"

"http://java.sun.com/dtd/facelet-taglib_1_0.dtd">

<facelet-taglib>

<namespace>http://www.springframework.org/security/tags</namespace>

<tag>

<tag-name>authorize</tag-name>

<handler-class>org.springframework.faces.security.FaceletsAuthorizeTagHandler</handler-class>

</tag>

<function>

<function-name>areAllGranted</function-name>

<function-class>org.springframework.faces.security.FaceletsAuthorizeTagUtils</function-class>

<function-signature>boolean areAllGranted(java.lang.String)</function-signature>

</function>
```

Spring security taglibs soubor II.

```
<function>
```

```
<function-name>areAnyGranted</function-name>
```

```
<function-class>org.springframework.faces.security.FaceletsAuthorizeTagUtils</function-class>
```

```
<function-signature>boolean areAnyGranted(java.lang.String)</function-signature>
```

```
</function>
```

```
<function>
```

```
<function-name>areNotGranted</function-name>
```

```
<function-class>org.springframework.faces.security.FaceletsAuthorizeTagUtils</function-class>
```

```
<function-signature>boolean areNotGranted(java.lang.String)</function-signature>
```

```
</function>
```

```
<function>
```

```
<function-name>isAllowed</function-name>
```

```
<function-class>org.springframework.faces.security.FaceletsAuthorizeTagUtils</function-class>
```

```
<function-signature>boolean isAllowed(java.lang.String, java.lang.String)</function-signature>
```

```
</function>
```

```
</facelet-taglib>
```


Spring Security konfigurace I.

- Do root contextu (standardně WEB-INF/applicationContext.xml) přidejte:

```
<import resource="security.xml"/>
```

- Vytvořte soubor WEB-INF/security.xml:

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

```
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.springframework.org/schema/security"
```

```
xsi:schemaLocation="http://www.springframework.org/schema/security
```

```
http://www.springframework.org/schema/security/spring-security-3.2.xsd
```

```
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
```

Spring Security konfigurace II.

```
<http auto-config="true" use-expressions="true">
    <intercept-url pattern="/**" access="hasAnyRole('ROLE_USER','ROLE_ADMIN')" />
</http>

<authentication-manager>
    <authentication-provider>
        <user-service>
            <user name="jirka" password="jirka" authorities="ROLE_USER" />
            <user name="admin" password="admin" authorities="ROLE_ADMIN" />
        </user-service>
    </authentication-provider>
</authentication-manager>

</beans:beans>
```

Použití I.

- Poté je možné používat Spring security standardním způsobem uvnitř XHTML souboru:

```
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:security="http://www.springframework.org/security/tags">

<h:head> </h:head>

<h:body>

    <security:authorize access="hasRole('ROLE_ADMIN')">

        Tento text se zobrazí pouze administrátorovi

    </security:authorize>

</h:body>

</html>
```

Použití II.

- Nebo pomocí JSF atributu rendered:

```
<h:form id="formGrid" rendered="#{security:areAllGranted('ROLE_ADMIN')}">  
</h:form>
```

Spring & Servlet I.

- Pro generování PDF / obrázků / ... můžete použít servlety. Jak provést integraci servletu se Springem? Existuje několik možností.
- Nejlepší způsob:

```
@WebServlet("/image")

public class IconServlet extends HttpServlet {

    @Autowired

    private MyService myService;

    @Override

    public void init() throws ServletException {

        SpringBeanAutowiringSupport.processInjectionBasedOnCurrentContext(this);

    }

}
```

Poznámka: Třída Servletu musí být „oscanovaná“ pomocí `<context:component-scan />`

Spring & Servlet II.

- Obecný low level přístup je:

```
@WebServlet("/icon")
```

```
public class IconServlet extends HttpServlet {
```

```
private MyService myService;
```

```
@Override
```

```
public void init() throws ServletException {
```

```
    WebApplicationContext applicationContext =
```

```
        WebApplicationContextUtils.getWebApplicationContext(getServletContext());
```

```
    myService = applicationContext.getBean(MyService.class);
```

```
}
```