PV230 - cvičení 5 - příklady + postup

PV230 - cvičenie 5

Pridanie Eventov

jsp/detail/view.jsp

- pridáme tlačidlo, ktoré na kliknutie vyvolá akciu (portlet:actionURL), ktorá bude znamenať, že užívateľ si chce kúpiť daný produkt:

```
<div class="buttons">
    <portlet:actionURL name="<%=ACTION_ADD_TO_BASKET%>" var="addToBasketURL">
        <portlet:param name="<%=PARAM_PRODUCT_ID%>" value="${product.id}"/>
    </portlet:actionURL>
    <input type="button" onclick="location.href = '${addToBasketURL}'" value="<f:message
key="ss-msg-buy"/>"/>
</div>
```

DetailPortlet.java

- pridáme spracovanie tejto akcie, ktoré spôsobí vyvolanie portletovej udalosti (eventu), ktorú bude spracovávať náš ďalší portlet

```
    @ProcessAction(name=ACTION_ADD_TO_BASKET)
    public void actionBuy(ActionRequest request, ActionResponse response) throws
PortletException, IOException {
        String productId = request.getParameter(PARAM_PRODUCT_ID);
        long id = Long.parseLong(productId);
        ProductDTO product = ServiceProvider.getCatalogService().getProductById(id);
        response.setEvent(EVENT_BUY_PRODUCT, product);
        response.setRenderParameter(PARAM_PRODUCT_ID, productId);
    }
```

portlet.xml

- pridáme definíciu portletovej udalosti do portlet.xml (pozor na umiestnenie)

```
    <event-definition>
        <name>buyProductEvent</name>
        <value-type>eu.ibacz.pv230.backend.dto.ProductDTO</value-type>
    </event-definition>
```

- deklarujeme, že portlet DetailPortlet túto udalosť odosiela

```
    <supported-publishing-event>
        <name>buyProductEvent</name>
    </supported-publishing-event>
```

Konfigurácia Spring Portlet MVC

pom.xml

- pridáme dependency na knižnicu Spring Portlet MVC:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc-portlet</artifactId>
    <version>3.0.3.RELEASE</version>
</dependency>
```

- upravíme <portlet.hotdeploy> aby ukazoval na správny adresár

web.xml

- upravíme web.xml, tak aby jeho obsah bol:

```
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-context/applicationContext.xml</param-value>
    </context-param>

    <listener>

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

    <servlet>
        <servlet-name>ViewRendererServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.ViewRendererServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>ViewRendererServlet</servlet-name>
        <url-pattern>/WEB-INF/servlet/view</url-pattern>
    </servlet-mapping>
```

applicationContext.xml

- vytvoríme súbor WEB-INF/spring-context/applicationContext.xml, ktorý obsahuje všeobecnú konfiguráciu Springu pre celú aplikáciu
- pridáme importy na ďalšie Springové konfiguračné súbory, ktoré sa nachádzajú v backende a nastavujú pripojenie na databázu:

```
    <import resource="classpath*:META-INF/liferay-data-source.xml"/>
    <import resource="classpath*:META-INF/spring-backend.xml"/>
```

Tvorba 'portletu'

- vytvoríme triedu `eu.ibacz.pv230.simpleshop.portlet.basket.BasketViewController` s anotáciami @Controller a @RequestMapping("VIEW")
- pridáme si atribút BasketService, pomocou ktorého budeme pracovať s košíkom. Tento atribút nám bude Spring injectovať pomocou Dependency Injection:

```
    @Autowired
    private BasketService basketService;
```

- pridáme metódu obsluhujúcu render požiadavky portletu - podobne ako je metóda `doView` z Portlet API:

```
    @RenderMapping
    public String renderDefault(RenderRequest request, Model model) {
        BasketDTO basket = basketService.getBasket(request.getRemoteUser());
        model.addAttribute(ATTRIBUTE_BASKET, basket);
        return JSP_VIEW;
    }
```

- nakonfigurujeme ViewResolver v `applicationContext.xml`

```
    <bean id="viewResolver" class=
"org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
        <property name="contentType" value="text/html;charset=UTF-8" />
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>
```

- vytvoríme JSP stránku s portletom, na umiestnení ktoré sme určili v metóde `renderDefault`:

```
<%@include file="../init.jspf" %>
<%@page import="static eu.ibacz.pv230.simpleshop.portlet.basket.BasketConstants.*" %>

<div id="${ns}basketView" class="basket-portlet">

    <div>
        <span class="label"><spring:message code="label-product-count"/>:</span>
        <span><c:out value="${basket.itemCount}"/></span>
    </div>
    <div class="divider"/>
    <div>
        <span class="label"><spring:message code="label-total-price"/>:</span>
        <span><c:out value="${basket.totalPrice}"/></span>
    </div>

    <div class="buttons">
        <portlet:renderURL var="showBasketURL" windowState="maximized">
            <portlet:param name="<%=PARAM_PAGE%>" value="<%=PAGE_BASKET_MAXIMIZED%>"/>
        </portlet:renderURL>
        <input type="button" onclick="location.href = '${showBasketURL}'" value=
"<spring:message code='label-show-basket'/>"/>
    </div>

</div>
```

- vytvoríme konfiguračný súbor Springu určený pre tento portlet - `spring-context/portlet/basket-portlet.xml`:

```
    <context:component-scan base-package="eu.ibacz.pv230.simpleshop.portlet.basket"/>

    <bean id="messageSource" class=
"org.springframework.context.support.ResourceBundleMessageSource">
        <property name="useCodeAsDefaultMessage" value="true" />
        <property name="basenames">
            <list>
                <value>content.basket</value>
            </list>
        </property>
    </bean>
```

- pridáme definíciu portletu do `portlet.xml`:

```
    <portlet>
        <portlet-name>Basket</portlet-name>
        <portlet-class>org.springframework.web.portlet.DispatcherPortlet</portlet-class>
        <init-param>
            <name>contextConfigLocation</name>
            <value>/WEB-INF/spring-context/portlet/basket-portlet.xml</value>
        </init-param>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>VIEW</portlet-mode>
        </supports>
        <supported-locale>en</supported-locale>
        <supported-locale>cs</supported-locale>
        <resource-bundle>content.basket</resource-bundle>
        <supported-processing-event>
            <name>buyProductEvent</name>
        </supported-processing-event>
    </portlet>
```

- pridáme definíciu portletu do `liferay-portlet.xml`:

```
    <portlet>
        <portlet-name>Basket</portlet-name>
        <instanceable>true</instanceable>
        <header-portlet-css>/css/common.css</header-portlet-css>
        <header-portlet-css>/css/basket.css</header-portlet-css>
    </portlet>
```

- pridáme definíciu portletu do `liferay-display.xml`:

```
<display>
    <category name="SimpleShop">
        ...
        <portlet id="Basket" />
        ...
    </category>
</display>
```

- máme funkčný (dúfam ) portlet!

Spracovanie udalostí

- deklarovali sme v `portlet.xml`, že náš portlet je schopný prijímať udalosť `buyProductEvent`, naimplementujeme teda jej spracovanie
- pridáme metódu oanotovanú `@EventMapping`, ktorá udalosť spracuje:

```
    @EventMapping(EVENT_BUY_PRODUCT)
    public void eventBuyProduct(EventRequest request, EventResponse response) {
        ProductDTO product = (ProductDTO) request.getEvent().getValue();
        basketService.addProductToBasket(product, request.getRemoteUser());
    }
```

- máme fungujúci košík!