JSF, Primefaces Hello World

Java EE 7, 8 archetype

- Pro vytvoření kostry Java EE 8 projektu rád používám tento archetype:
  - https://javalibs.com/archetype/com.airhacks/javaee8-essentials-archetype
- Pro Java EE 7:
  - https://javalibs.com/archetype/com.airhacks/javaee7-essentials-archetype

Dependency

- Přidejte do `pom.xml`:

```
<dependency>
  <groupId>org.primefaces</groupId>
  <artifactId>primefaces</artifactId>
  <version>6.2</version>
</dependency>
```

- Pokud nemáte aplikační server, ale webový kontejner (Tomcat, Jetty), přidejte navíc:

```
<dependency>
   <groupId>org.glassfish</groupId>
   <artifactId>javax.faces</artifactId>
   <version>2.2.8</version>
</dependency>
```

NEBO:

```
<dependency>
  <groupId>com.sun.faces</groupId>
  <artifactId>jsf-api</artifactId>
  <version>2.2.8</version>
</dependency>

<dependency>
  <groupId>com.sun.faces</groupId>
  <artifactId>jsf-impl</artifactId>
  <version>2.2.8</version>
</dependency>
```

web.xml I.

- Přidejte do `web.xml`:

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.xhtml</url-pattern>
</servlet-mapping>
```

web.xml II.

- Pokud používáte Jetty, pak navíc přidejte:

```
<listener>
  <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
```

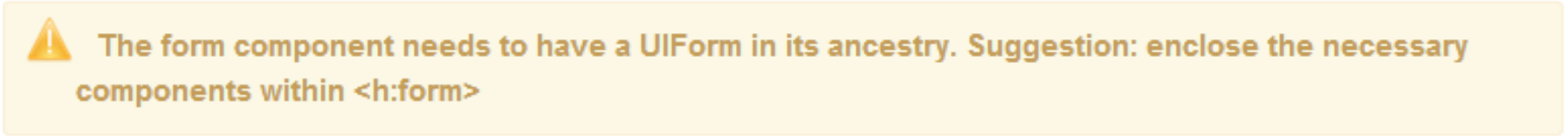
web.xml – development I.

- Při vývoji je best practice přidat do `web.xml`:

```
<!-- Time in seconds that facelets should be checked for changes since last request.  -->
<!-- A value of -1 disables refresh checking. -->
<context-param>
   <param-name>javax.faces.FACELETS_REFRESH_PERIOD</param-name>
   <param-value>0</param-value>
</context-param>

<!-- Set the project stage to "Development", "UnitTest", "SystemTest", or "Production".
-->
<!-- An optional parameter that makes troubleshooting errors much easier. -->
<!-- You should remove this context parameter before deploying to production! -->
<context-param>
   <param-name>javax.faces.PROJECT_STAGE</param-name>
   <param-value>Development</param-value>
</context-param>
```

web.xml – development II.

- V development módu se uvnitř `<h:messages />` nebo `<p:messages />` zobrazují warn hlášky jako je:



- Je vhodné tyto hlášky na každé stránce zobrazovat, abyste co nejrychleji zjistili chybu ve Vašem kódu. Proto do šablony někam nahoru dávám:

```
<p:messages severity="warn" />
```

web.xml – development vs. production

- Je nutné použít jiné nastavení context parametrů v development, produkčním, … prostředí. Jak na to?
  - Pokud používáte Maven:
    - https://community.jboss.org/wiki/HowToConfigureJavaEEApplicationToApplyDifferentSettingsinWebxmlEtcForVariousEnvironmentsByMaven
  - Rozchození pro Jetty je na následujících stránkách.

web.xml – development vs. production – Jetty I.

Dovnitř tagu `build`!!!:

```
<filters>
   <filter>${basedir}/src/main/filters/${filter.name}.properties</filter>
</filters>
<resources>
   <resource>
      <directory>src/main/webapp/WEB-INF</directory>
      <filtering>true</filtering>
      <targetPath>../jettyFilteredResources</targetPath>
   </resource>
   <resource>
      <directory>src/main/resources</directory>
      <targetPath>../classes</targetPath>
   </resource>
</resources>
```

web.xml – development vs. production – Jetty II.

- Použití Jetty pluginu (uvnitř `build` tagu):

```
<plugin>
   <groupId>org.eclipse.jetty</groupId>
   <artifactId>jetty-maven-plugin</artifactId>
   <version>9.1.3.v20140225</version>
   <configuration>
      <webAppConfig>
         <descriptor>target/jettyFilteredResources/web.xml</descriptor>
      </webAppConfig>
      <scanIntervalSeconds>3</scanIntervalSeconds>
   </configuration>
</plugin>
```

web.xml – development vs. production – Jetty III.

- Dovnitř tagu `project`:

```
<profiles>
   <profile>
      <id>dev</id>
      <activation> <activeByDefault>true</activeByDefault> </activation>
      <properties> <filter.name>dev</filter.name> </properties>
   </profile>
   <profile>
      <id>prod</id>
      <properties> <filter.name>prod</filter.name> </properties>
   </profile>
</profiles>
```

web.xml – development vs. production – Jetty IV.

- `src/main/filters/dev.properties:`

```
jsf.projectStage=Development
jsf.faceletsRefreshPeriod=0
```

- `src/main/filters/prod.properties:`

```
jsf.projectStage=Production
jsf.faceletsRefreshPeriod=-1
```

web.xml – development vs. production – Jetty V.

- web.xml:

```
<context-param>
   <param-name>javax.faces.FACELETS_REFRESH_PERIOD</param-name>
   <param-value>${jsf.faceletsRefreshPeriod}</param-value>
</context-param>

<context-param>
   <param-name>javax.faces.PROJECT_STAGE</param-name>
   <param-value>${jsf.projectStage}</param-value>
</context-param>
```

web.xml – development vs. production – Jetty VI.

- Spuštění:

```
mvn jetty:run
mvn package -P prod
```

Skip comments

- Tip: tímto způsobem se odstraní komentáře (dovnitř web.xml):

```
<context-param>
   <param-name>javax.faces.FACELETS_SKIP_COMMENTS</param-name>
   <param-value>true</param-value>
</context-param>
```

- Toto používám velice často zejména z toho důvodu, protože JSF interpretuje komentáře v XHTML souboru a tedy když v něm máte chybu, pak kvůli tomu celá stránka nefunguje.

Managed Bean

- Vytvořte třídu `Hello.java`:

```
@Named
public class Hello {

   public String getWorld() {

      return "Hello world!";

   }

}
```

Dřív se používala anotace @ManagedBean

https://stackoverflow.com/questions/4347374/backing-beans-managedbean-or-cdi-beans-named/4347707

- Použití je poté: `#{hello.world}`

Název beany je standardně název třídy s prvním písmenem lowercase.
Lze změnit pomocí:
`@Named("test")`

Poté by bylo použití: `#{test.world}`

Poznámka: Tomuto XHTML souboru se říká facelet.
Dřív (v JSF 1.X) se používaly JSP soubory.

index.xhtml

- Vytvořte soubor `index.xhtml`:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
   <h:head>
   </h:head>
   <h:body>
      #{hello.world}
   </h:body>
</html>
```

Dřív bylo nutné použít:

```
<h:outputText value="#{hello.world}" />
```

Nyní použijete `h:outputText` pouze tehdy, když nechcete escapovat HTML znaky (standardně se HTML znaky escapují):

```
<h:outputText value="#{hello.world}" escape="false" />
```

Namespaces od Java EE 6

- Od Java EE 6 můžete používat tyto namespace:

```
xmlns:h="http://xmlns.jcp.org/jsf/html"
xmlns:f="http://xmlns.jcp.org/jsf/core"
xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
xmlns:composite="http://xmlns.jcp.org/jsf/composite"
xmlns:pt="http://xmlns.jcp.org/jsf/passthrough"
xmlns:p="http://primefaces.org/ui"
```

Primefaces komponenty

- Můžete používat i JSTL knihovny, ale jejich použití není doporučené:

```
xmlns:c="http://xmlns.jcp.org/jsp/jstl/core"
xmlns:fmt="http://xmlns.jcp.org/jsp/jstl/core"
xmlns:fn="http://xmlns.jcp.org/jsp/jstl/functions"
```

- Pozor! Pokud nepoužíváte min. Java EE 6, pak nové namespace nepoužívejte!!! … nebudou fungovat ;-)

Namespaces do Java EE 6

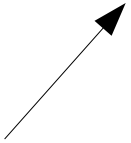
- Dříve používané namespaces:

```
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:composite="http://java.sun.com/jsf/composite"
xmlns:pt="http://java.sun.com/jsf/passthrough"
xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:fmt="http://java.sun.com/jsp/jstl/core"
xmlns:fn="http://java.sun.com/jsp/jstl/functions"
xmlns:p="http://primefaces.org/ui"
```

Facelets templates

template.xhtml I.

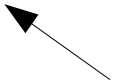
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">

<h:head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>#{title}</title>
</h:head>
```



Použití parametru

template.xhtml II.

```
    <h:body>
        <div>
            TODO MENU
        </div>
        <div>
            <ui:insert name="content">Content Section</ui:insert>
        </div>
        <div>
            TODO FOOTER
        </div>
    </h:body>
</html>
```



Tento text se dynamicky nahradí obsahem

index.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
   <ui:composition template="/template.xhtml">
    <ui:param name="title" value="index page" />
      <ui:define name="content">
         #{hello.world}
      </ui:define>
   </ui:composition>
</html>
```

Použití šablony

Definování parametru

Hodnota atributu „name“ tagu `ui:define` odpovídá hodnotě atributu „name“ tagu `ui:insert` v šabloně

Resources (CSS, JS, images)

Resources I.

- Resources (css, image, javascript) mohou v jednom z následujících umístění:
  - `src/main/webapp/resources/`
  - `META-INF/resources/`
- Vložení CSS do stránky:

```
<h:outputStylesheet library="css" name="default.css" />
```

Kam s tím? Ihned za tag `<h:body>`
Proč? Aby byl soubor `default.css` načten až po generovaných CSS souborech a tudíž jste v něm mohli přepisovat jejich hodnoty.

- NEBO:

```
<link rel="stylesheet" type="text/css" href="#{resource['css:default.css']}" />
```

Přístup k resources pomocí EL

Resources II.

- Vložení obrázku do stránky:
  - Musí existovat:

    `src/main/webapp/resources/images/logo.jpg`
  - Poté je možné přidat obrázek do stránky tímto způsobem:

```
<h:graphicImage value="/resources/images/logo.jpg" />
```

  NEBO:

```
<h:graphicImage value="#{resource['images:logo.jpg']}" />
```

- Vložení JavaScriptu do stránky:
  - Musí existovat:

    `src/main/webapp/resources/js/jquery.min.js`
  - Poté je možné přidat JS do stránky tímto způsobem:

```
<h:outputScript library="js" name="jquery.min.js" />
```

Resources III.

- Poznámky:
  - Resources mohou být i v podadresářích.
  - Je stále možné používat klasické HTML tagy (link, script, img) pro include CSS, JS a obrázků.

Expression Language

- Expression language výrazy mohou být uvnitř `${}` nebo `#{}`
- Co z toho použít?
  - Výrazy `${}` jsou vyhodnoceny ihned a je možné je použít pouze pro čtení, používaly se dříve.
  - Výrazy `#{}` mohou být vyhodnoceny v různých fázích cyklu stránky.
    - Používá se obvykle u JSF managed bean.
    - Nemusí sloužit pouze pro čtení, ale i uložení změn do managed bean.
    - Je doporučené používat pouze `#{}`
    - Optimalizační tip: vyhněte se použití více logiky uvnitř EL výrazů.

c:forEach vs. ui:repeat I.

- Foreach cyklus můžete udělat pomocí `c:forEach` nebo pomocí `ui:repeat`. Co z toho použít? Pokud možno vždy `ui:repeat`, protože se vyhnete jednomu problému:
- Máme dva záznamy v databázi.

`ItemManager:`

```
public List<Item> getItems() {
   return itemService.findAll();
}
```

- `items.xhtml:`

```
<c:forEach items="#{itemController.items}" var="item">
   <a href="item-detail.xhtml?id=#{item.id}">#{item.name}</a> <br />
</c:forEach>
```

- **Metoda `getItems()` se zavolá 5 krát, tudíž se vykoná 5 SELECTů do databáze!**

c:forEach vs. ui:repeat II.

- Pokud místo `c:forEach` použijete `ui:repeat`, pak se metoda `getItems()` zavolá pouze jednou, tudíž se vykoná pouze jeden SELECT:

```
<ui:repeat var="item" value="#{itemController.items}">
    <a href="item-detail.xhtml?id=#{item.id}">#{item.name}</a> <br />
</ui:repeat>
```

- Obecně je dobré v JSF pokud možno JSTL nepoužívat vůbec, vyhnete se kupě problémů.
  - http://drewdev.blogspot.cz/2008/08/cforeach-with-jsf-could-ruin-your-day.html
- Nevýhoda `ui:repeat`:
  - Dovnitř tagu `value` je možné vkládat pouze instance typu `java.util.List` nebo statické pole, nikoli `java.util.Collection`!

c:if vs rendered

- JSTL tag `c:if` je vhodné nepoužívat, místo toho se používá atribut `rendered`, který je možné nastavit na spoustě komponent, které generují HTML kód:

```
<h:panelGroup rendered="#{empty basket.items}">
   Please select some items into basket
</h:panelGroup>
<h:panelGroup rendered="#{!empty basket.items}">
   <ui:repeat var="oi" value="#{basket.items}">
      #{oi.item.name}: #{oi.quantity}x <br />
   </ui:repeat>
</h:panelGroup>
```

Další JSF knihovny

- Knihovna OmniFaces obsahuje celou řadu užitečných utility metod:
  - http://omnifaces.org/
- Pokud potřebujete URL Rewrite (jako .htaccess), pak použijte knihovnu PrettyFaces:
  - http://ocpsoft.org/prettyfaces/