

JSF aplikace

- Součástí webové aplikace – samostatné nebo jako modul Enterprise app.
- Projekt musí obsahovat JSF knihovny
- Nastavení se provádí v konfiguračních souborech
 - '*web.xml*', '*sun-web.xml*' (závislé na aplikačním serveru)
 - '*faces-config.xml*'
 - uložené ve '*Web/WEB-INF*'
 - v NetBeans přístupné také v '*Configuration Files*'

Tvorba aplikace - přehled

- 1) Struktura aplikace, model, business metody využívající model
- 2) Webový projekt, backing beans – třídy spolupracující s uživatelským rozhraním (zpracovávají vstupy a výstupy)
- 3) Zobrazení – JSF komponenty
- 4) Vytvoření navigace
- 5) Validace
- 6) Internacionalizace
- 7) Šablonování

Základ aplikace #1

- Otevřít enterprise projekt x33eja-jsf (obsahuje EJB modul s datovým modelem a obslužnou logikou)
- WEB modul
 - File – New Project – kategorie Java Web – Web Application
 - pojmenovat např. X33eja-jsf-war a umístit do složky, ve které se nachází enterprise projekt x33eja-jsf
 - Context path ponechat na: /x33eja-jsf-war (nepoužívat CDI)
 - Přidat framework Java Server Faces
 - Ponechat výchozí knihovnu s JSF 2.0, která je součástí knihoven serveru
 - Podívat se do záložky *Configuration* na JSF servlet URL Pattern – výchozí: “/faces/”
 - Preferovaná syntaxe Facelets

Základ aplikace #2

- Vytvoří se konfigurační soubor *'web.xml'* a *'index.xhtml'*
- Dodatečné přidání frameworku JSF
 - KM na webovém modulu – *Properties* – v okně *Categories* zvolit *Frameworks* – a poté vpravo *ADD* a z dostupných vybrat *Java Server Faces*
- K enterprise projektu přidat daný WEB modul:
 - kontextové menu (KM) – na Java EE Modules enterprise projektu – *Add Java EE Module*
- Zkontrolovat připojení do databáze (součástí EJB modulu)
 - Persistence Unit resp. *'sun-resources.xml'* (ve složce *Server Resources*)
 - Username: *jpa*, password: *test*, database: *jpa_example*

Konfigurace '*web.xml*'

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <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>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
  </welcome-file-list>
</web-app>

```

"Development", "UnitTest", "SystemTest", nebo
 výchozí "Production"

kladné číslo – pořadí v jakém se má servlet načítat

JSF servlet bude zpracovávat stránky s URL '/faces/'
 - tzv. prefixové mapování, nebo s nějakou příponou
'*.jsf', **'*.faces'** nebo **'*.xhtml'** apod.

JSF servlet bude zpracovávat stránky s URL '/faces/'
 - tzv. prefixové mapování, nebo s nějakou příponou
'*.jsf' nebo **'*.faces'** apod.

Úvodní stránka

- Spustit aplikaci, zadat url '*/faces/index.xhtml*', '*index.xhtml*' a porovnat zdrojový kód, kde je chyba?
- Nastavit url-pattern na **'*.xhtml'** a úvodní stránku na **'index.xhtml'** (odstranit '*index.jsp*', pokud je vytvořen)

Backing bean #1

- Vytvořit klasickou třídu '*CoursesBean.java*' (balíček např. '*x33eja.back*')
- Vytvoření konfiguračního souboru – File – New – kategorie Java Server Faces – JSF Faces Configuration – '*faces-config.xml*'
- V JSF 1.2 nutná registrace ve '*faces-config.xml*' pod jménem '*students*': zobrazit XML, KM – JSF – Add Managed Bean

```
<managed-bean>
  <managed-bean-name>courses</managed-bean-name>
  <managed-bean-class>cz.cvut.x33eja.back.CoursesBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

- Anotace `@javax.faces.bean.ManagedBean(name="courses")`
- Doba platnosti se řídí anotacemi `SessionScoped`, `RequestScoped`, `ApplicationScoped`, `ConversationScoped`, `ViewScoped`, pro kurzy zvolit `ApplicationScoped`
- Totéž lze File – New File – Categories – JSF – **JSF Managed Bean** – vyplnit jméno a platnost

Backing bean #2

- *KM* – Insert Code – Call Enterprise Bean – vybrat z EJB projektu '*SchoolCenter*'

```
@EJB
SchoolCenterLocal schoolCenter;

public List<Course> getAllCourses() {
    return schoolCenter.getAllCourses();
}
```

- Ke každé datové položce třídy vygenerovat getter a setter (má-li být možnost nastavovat hodnoty – pomocí Insert Code nebo Refactor – Encapsulate Fields) př. `Course selectedCourse = null; ...`
- Přístup k jiné backing bean (v příkladu nepoužito)

```
CoursesBean sb = (CoursesBean) FacesContext.getCurrentInstance()
    .getApplication().getELResolver()
    .getValue(context.getELContext(), null, "courses");
```
- Vytvořit novou SessionScoped bean: '*StudentsBean.java*' s metodou `getAllStudents`

Zobrazení dat #1 – seznam studentů

- *'index.xhtml'*
- z palety JSF (není-li zobrazena – Window - Palette) komponentu JSF Data Table From Entity
 - vybrat entitu *'x33eja.model.Student'* a zvolit odpovídající entitu z balíčku *'x33eja.model.Student'*, zvolit registrovanou bean *'students'* – automaticky se vygeneruje tabulka
- přidat namespace `xmlns:f="http://java.sun.com/jsf/core"`
- `<h:dataTable>`
 - atribut *var* s hodnotou *'item'* se odkazuje na jednotlivé property vybrané entity
 - atribut *value* obsahuje *'#{students.allStudents}'* (pokud je backing bean správně zaregistrovaná objeví se v Code completion po napsání `#{ }`)

Zobrazení dat #2

- Upravit název stránky do elementu title a h1
- Do jednoho sloupce sloučit jméno a příjmení
- U sloupce Supervisor zobrazovat jeho jméno a příjmení (nikoli výstup toString) tzn. '`<h:outputText value="#{item.hasSupervisor.firstName} #{item.hasSupervisor.surname}"/>`'
- Zapsané předměty zobrazit jako vloženou data table
- Přidat nad tabulku tlačítků *'New Student'* a do tabulky sloupec s tlačítkem *'Edit'*
- V backing bean *'students'* vytvořit metody *'newStudent'*, *'editStudent(Student s)'* a *'saveStudent'* s návratovým typem String

```
private Student student = null;

public String newStudent() {
    student = new Student();
    return "student";
}
```

```
public String saveStudent() {
    schoolCenter.updateStudent(student);
    return "students";
}

public String editStudent(Student st) {
    this.student = st;
    return "student";
}
```

Zobrazení dat #3 – nový student

- **<h:messages />** - doporučuji pro ladění umístit na každé stránce, kde se upravují data pro zobrazení možných chyb
 - v project stage Development jsou sice chyby implicitně zobrazeny, nicméně při změně na Production se bez tohoto atributu nezobrazí
 - Atributem `globalOnly="true"` zajistí, že se nebudou duplikovat do výpisu chyby komponent
- Provést deploy aplikace: *"Unable to find matching navigation case with from-view-id '/index.xhtml' for action '#{students.newStudent}' with outcome 'student'"*
- Vytvořit stránku *'student.xhtml'* pro úpravu údajů studenta (KM - Web Pages – New – Other – JSF – JSF Page – Facelets syntax), upravit název stránky a nadpis
 - komponenty `<h:form>`, `<h:panelGrid columns="3">`, `<h:outputText>`, `<h:inputText required="true">`, `<h:commandButton type="submit" value="Save" action="#{students.saveStudent}">`
 - Doplnit atribut ID, komponentu `h:message`, vyzkoušet a poté přidat atribut `Label` pro hezký výpis
 - Po nepodařeném Submit formuláře se změní URL – řešení?
- Umožnit navigaci zpět na seznam studentů (`h:commandLink`)

Konfigurace '*faces-config.xml*' #1

- File – New – Other- JSF – JSF Faces Configuration (ponechat výchozí název i umístění do WEB-INF)
- Možnost registrovat managed beans, validátory, konvertory ...
- Tvorba pravidel – vizuálně nebo v XML
 - KM - JSF – Insert – Navigation Rule: from view ID: *'/*'*
 - KM - JSF – Insert – Navigation Case: *'edit-student'*
 - Pravidlo může mít více případů (success, failed)
- Implicitní navigace
 - název stránky bez přípony *' .xhtml'*
 - bez deklarace ve *faces-config.xml* se nemění URL

Konfigurace '*faces-config.xml*' #2

```
<faces-config version="1.2"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">
```

```
<navigation-rule>
  <from-view-id>/index.jsp</from-view-id>
  <navigation-case>
    <from-outcome>student</from-outcome>
    <to-view-id>/student.xhtml</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <from-view-id>/*</from-view-id>
  <navigation-case>
    <from-outcome>index</from-outcome>
    <to-view-id>/index.xhtml</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>
</navigation-rule>
</faces-config>
```

← Navigační string (návratová hodnota z metody definované v action)

← Při přesměrování změnit URL

← Navigace je platná pro všechny stránky

Komponenta h:selectManyListbox

- Přidání zapsaných předmětů

Backing Bean:

```
public List<SelectedItem> getAllTeachersSelectList() {
    List<SelectedItem> items = new ArrayList<SelectedItem>();
    for (Teacher t : ssl.getAllTeachers()) {
        items.add(new SelectedItem(t, t.getSurname() + " " + t.getFirstName()));
    }
    return items;
}
```

student.xhtml:

```
<h:outputText value="Enrolled in:" />
<h:selectManyListbox id="enrolledIn"
    value="#{students.student.enrolledIn}" label="Enrolled in">
    <f:selectItems value="#{courses.allCourses}" />
</h:selectManyListbox>
<h:message for="enrolledIn" />
```

JSF - konvertor

- **Conversion Error setting value 'x33eja.model.Course@475b7d' for 'null Converter'**
– nepovedla se automatická konverze (String, Integer apod.)
- Třída s anotací **@FacesConverter**(value = "course") implementující interface Converter
- Implicitní metody `getAsObject` a `getAsString`
- Vygenerování lookup: Insert Code – Call EJB – zvolit SchoolCenter

```
@FacesConverter(value = "course")
public class CourseConverter implements javax.faces.convert.Converter {
    SchoolCenterLocal schoolCenter = lookupSchoolCenterLocal();

    @Override
    public Object getAsObject(FacesContext context, UIComponent component, String value) {
        return schoolCenter.getCourse(value);
    }

    @Override
    public String getAsString(FacesContext context, UIComponent component, Object value) {
        return String.valueOf(((Course) value).getCourseId());
    }

    private SchoolCenterLocal lookupSchoolCenterLocal() {
        ...
    }
}
```

- Komponentě přidat converter atribut nebo `f:converter` element

```
<h:selectManyListbox id="enrolledIn" value="#{students.student.enrolledIn}" label="Enrolled in">
    <f:selectItems value="#{courses.allCourses}" />
    <f:converter converterId="course" />
</h:selectManyListbox>
```

Použití SelectItem

- Přidání předmětů již funguje – komponenta selectManyListbox nezobrazuje hezky názvy předmětů, nicméně výsledek je na seznamu studentů zobrazen správně
- Změna '*List<Course> getAllCourses()*' na:

```
private List<SelectedItem> allCourses;

public List<SelectedItem> getAllCourses() {
    allCourses = new ArrayList<SelectedItem>();
    List<Course> courses = schoolCenter.getAllCourses();
    for (Course c : courses) {
        allCourses.add(new SelectItem(c, c.getName()));
    }
    return allCourses;
}
```

- Možno místo objektu vložit jen primární klíč, ale pak k tomu odpovídajícím způsobem opravit konvertor.

Komponenta selectOneListbox

- Přidání supervizora

Backing Bean:

```
public List<SelectItem> getAllTeachersSelectList() {
    List<SelectItem> items = new ArrayList<SelectItem>();
    for (Teacher t : ssl.getAllTeachers()) {
        items.add(new SelectItem(t, t.getSurname() + " " + t.getFirstName()));
    }
    return items;
}
```

student.xhtml:

```
<h:selectOneListbox value="#{students.student.hasSupervisor}" size="1">
    <f:selectItems value="#{students.allTeachersSelectList}" />
</h:selectOneListbox>
```


Zjednodušení práce se SelectItem

- Místo `<f:selectItems value="#{students.allCourses}" />`, kde `allCourses` vrací list `SelectItem` je možné použít list a vše dodefinovat ve stránce:

```
<f:selectItems value="#{courses.allCourses}"  
var="course" itemValue="#{course}"  
itemLabel="#{course.name}" />
```

- Výhoda: přehlednější kód v managed bean
- Nevýhoda: při každém použití nutno definovat výstup, takže ve větší aplikaci může dojít k různorodému zobrazování výstupu

Zobrazení detailu

- Dříve s výjimkou data modelu možnost použít `h:commandLink` s vloženým parametrem pomocí: `<f:param name="id" value="111" />`.
- Možno vkládat argumenty:

index.xhtml

```
<h:column>
  <h:commandButton value="Edit" action="#{students.editStudent(item)}" />
</h:column>
```

StudentsBean.java

```
public String editStudent(Student student) {
    this.student = student;
    return "student";
}
```

student.xhtml

- Přidat atribut **disabled** a neumožnit jeho editaci:
 - `disabled="#{students.student.birthNumber != null}"`
- Některé informace možné skrýt pomocí atributu **rendered**="`#{!empty(...)}`"
- Co chybí dobrému formuláři na detailu? Tlačítko 'Zpět'
 - `<h:commandLink action="index" value="Back to index" immediate="true" />`

DataModel

- Vytvořit DataModel a nastavit jej jako value do students.jsp (místo listu)

```
public DataModel getAllStudentsModel() {  
    allStudentsModel = new ListDataModel(getAllStudents());  
    return allStudentsModel;  
}
```

- Tlačítko edit (h:commandButton) volá akci:

```
public String editStudent() {  
    student = (Student) allStudentsModel.getRowData();  
    return "edit-student";  
}
```

JSF - validace

- Atribut required – nenulová hodnota vstupu
- f:validateLength
- Definice vlastního validátoru:

Java:

```
@FacesValidator(value="birthnumberValidator")
public class BirthnumberValidator implements javax.faces.validator.Validator {

    @Override
    public void validate(FacesContext context, UIComponent component, Object value) throws
ValidatorException {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

Dříve v JSF 1.2:

faces-config.xml:

```
<validator>
    <validator-id>BirthdateValidator</validator-id>
    <validator-class>cz.cvut.x33eja.util.BirthdateValidator</validator-class>
</validator>
```

JSF:

- atribut 'validator="birthnumberValidator"' nebo komponenta:

```
<f:validator validatorId="birthnumberValidator"/>
```
- Zobrazení chybové hlášky na daném místě pomocí: `<h:message for="bn" />`

JSF - lokalizace

- File – New – category Other – Properties File – **Translation**: Name: translation, folder: x33eja
- KM – Add Locale cs_CZ, KM – Add Property – Key, Value

```
students=Students
student_add=Add Student
student_edit=Edit
```

- Na vybraném properties souboru – KM – Open a doplnit překlady pro ostatní jazyky

- **faces-config.xml**

```
<application>
  <locale-config>
    <default-locale>en_GB</default-locale>
    <supported-locale>cs_CZ</supported-locale>
  </locale-config>
  <resource-bundle>
    <base-name>x33eja.translation</base-name>
    <var>msgs</var>
  </resource-bundle>
</application>
```

← Při nefunkčnosti přepínání v prohlížeči nahradit stejným klíčovým stringem např. Pouze “en” nebo “cs”. Stejně přejmenovat properties soubor.

- **JSF**

Použití: <h:outputText value="#{msgs.students}"/>

Nebo: <f:loadBundle basename="translation" var="msgs" /> - nefunguje doplňování kódu

Šablonovací systém #1

- Dříve nutno přidat samostatnou knihovnu Facelets <https://facelets.dev.java.net/> nyní součástí JSF 2
- Doporučená XML syntaxe – XHTML soubory
- Namespace: `xmlns:ui="http://java.sun.com/jsf/facelets"`
- Šablony i fragmenty stránky umísťujeme do WEB-INF, aby nebyly samostatně přístupné
- Web Pages – New File – kategorie JSF – Facelets Template
 - Name: x33eja-template, Folder: WEB-INF a vybrat rozvržení
 - Vytvoří se navíc adresář 'resources' obsahující CSS definice

Šablonovací systém #2

- Příklad šablony:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link href="./resources/css/default.css" rel="stylesheet" type="text/css" />
    <link href="./resources/css/cssLayout.css" rel="stylesheet" type="text/css" />
    <title>#{pageTitle}</title>
  </h:head>
  <h:body>
    <div id="top">
      <ui:insert name="top">Komponenta studium</ui:insert>
    </div>
    <div>
      <div id="left">
        <ui:insert name="left">
          <ui:include src="/WEB-INF/includes/menu.xhtml"/>
        </ui:insert>
      </div>
      <div id="content" class="left_content">
        <ui:insert name="content">Vychozí obsah</ui:insert>
      </div>
    </div>
    <div id="bottom">
      <ui:insert name="bottom">&copy; X33EJA 2011</ui:insert>
    </div>
  </h:body>
</html>
```

Šablonovací systém #3

- File – New – Java Server Faces – **Facelets Template Client**
 - File Name: students, zvolit šablonu (ve WEB-INF)
 - Jako root tag ponechat html – cokoli mimo ui:composition následně nebude zobrazeno
- **Příklad klienta šablony** (obsah mimo element ui:composition není zobrazen):

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">

  <body>
    <ui:composition template="./WEB-INF/includes-templates/x33eja-template.xhtml">
      <ui:param name="pageTitle" value="Správa studentů" />
      <ui:define name="content">
        <h:form>
          obsah
        </h:form>
      </ui:define>
    </ui:composition>
  </body>
</html>
```


Šablonovací systém #4

- Prosté vložení souboru
 - `<ui:include src="./WEB-INF/..." />`
 - Vhodné pro vložení menu – stránka obsahující element `<ui:composition>`
 - Přidat namespace `xmlns:ui="http://java.sun.com/jsf/facelets"`

Přehled možných výjimek #1

- *'org.apache.jasper.JasperException: java.lang.RuntimeException: Cannot find FacesContext'*
 - ve web.xml – servlet mapping – Faces Servlet zkontrolovat url-pattern:
 - Pro: **'/faces/*'** nutné jej také zadat do URL např. *'aplikace-war/welcomeJSF.jsp'* nahradit: *'aplikace-war/faces/welcomeJSF.jsp'*
 - Pro **'*.jsf'** místo *'aplikace-war/welcomeJSF.jsp'* nastavit *'aplikace-war/welcomeJSF.jsf'*
- *'j_id_id47: Validation Error: Value is not valid'*
 - h:selectOneListbox (ID konkrétní komponenty lze zjistit např. pomocí pluginu FireBug kliknutím na daný element stránky)
 - Chyba se zobrazí, je-li na stránce komponenta h:messages
 - Pro danou entitu implementovat equals viz entita Person (KM v dané entitě - Insert Code – Equals ... a zvolit např. primární klíč)

Přehled možných výjimek #2

- *'java.io.NotSerializableException: x33eja.model. ...'*
 - implementovat pro entity rozhraní Serializable
- *"Unable to find matching navigation case with from-view-id '/index.xhtml' for action '#{students.newStudent}' with outcome 'student'"*
 - Ověřit, že existuje.xhtml soubor referencovaný v outcome
- *"Caused by: Exception: org.eclipse.persistence.exceptions.ValidationException*
Exception Description: Missing descriptor for [class java.lang.String]. Verify that the descriptor has been properly registered with the Session."
 - Máte implementovaný konvertor?

Přehled možných výjimek #3

- *'java.util.MissingResourceException: Can't find bundle for base name translation, locale cs_CZ'*
 - Doplnit správný balíček obsahující properties file např.:
<base-name>x33eja.translation</base-name>