

JSF УПРАВЛЯЕМИ КОМПОНЕНТИ



Съдържание

Част 1:

- JSF въведение
- JSF правила за навигация
- **JSF управляеми компоненти (managed beans)**
- JSF компоненти на графичен потребителски интерфейс

След лекцията ще знаете

- Какво е Java Beans
- Видове JavaBeans
- На какви правила се подчиняват
- Видове декларации - примери
- Механизъм на управление
- Области на видимост и време на живот
- Понятие и работа с компоненти в сесии
- Видове свойства в компонент и достъп: четене/писане/вътрешни
- Достъп до свойства и методи през JSF страниците
- Примери и демонстрации

Дефиниция (1 / 2)

- ***„Managed Beans, lightweight container-managed objects (POJOs) with minimal requirements, support a small set of basic services” (such as resource injection, lifecycle callbacks, and interceptors.)***

Дефиниция (2/2)

- “Managed Beans **represent a generalization of the managed beans specified by JavaServer Faces technology and can be used anywhere in a Java EE application, not just in web modules.**
- *The Managed Beans specification is part of the Java EE 7 platform specification (JSR342). The Java EE 7 platform requires Managed Beans 1.0” [1]*

MANAGED BEANS и JSF

- JSF улеснява работата с JavaBeans
- Описват се посредством класове
- Създават се при първо обръщение към тях
- За целта компонентите, които ще се управляват от контейнера **ТРЯБВА ДА** се обявят като такива
 - ▣ **например:** да бъдат описани във файл-дескриптор

Демо (може и в края на лекцията)

- Направете управляем компонент
- Конфигурирайте потребителско име на компонента

Принципи (POJO)

package models;//вариант 1

```
class Student {  
    String firstName;  
    public Student(){}  
    public String getName() {  
        return firstName;  
    }  
    public void setName(String name) {  
        return firstName;  
    }  
}
```


Пример – дескриптор (например faces-config.xml)

1.1. Name of
Managed Bean
(accessible in JSF)

<managed-bean>

<managed-bean-name>**student**</managed-bean-name>

<managed-bean-class>**models.Student**</managed-bean-class>

<managed-bean-scope>**request**</managed-bean-scope>

<managed-property>

<property-name>name</property-name>

<property-class>java.lang.String</property-class>

<value>*Student Name*</value>

</managed-property>

</managed-bean>

Основни
Характеристики
на ManagedBean

1.2. Name of
class

Основни
свойства (0 или повече)

Managed
bean

2.1. Name of
property

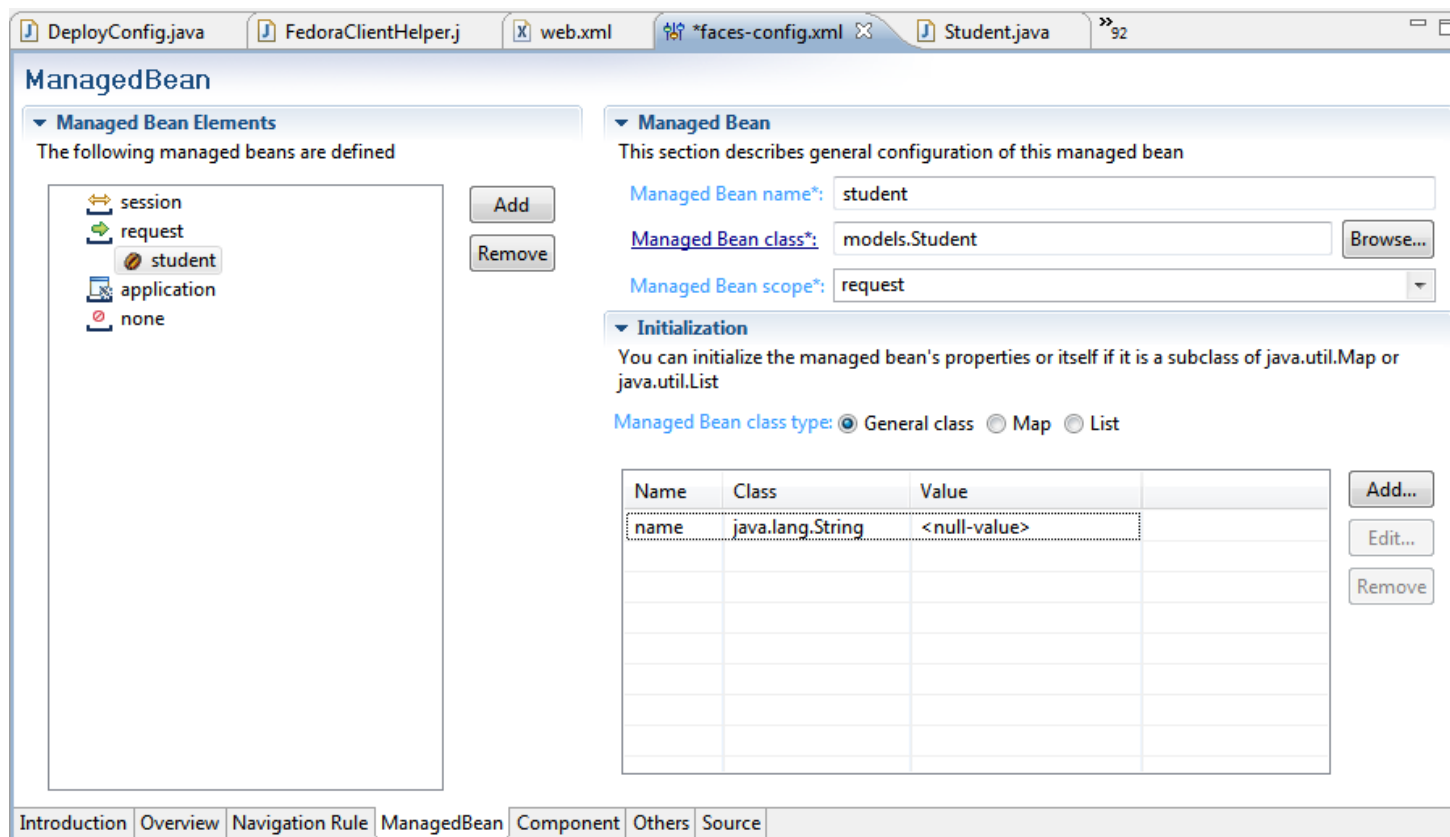
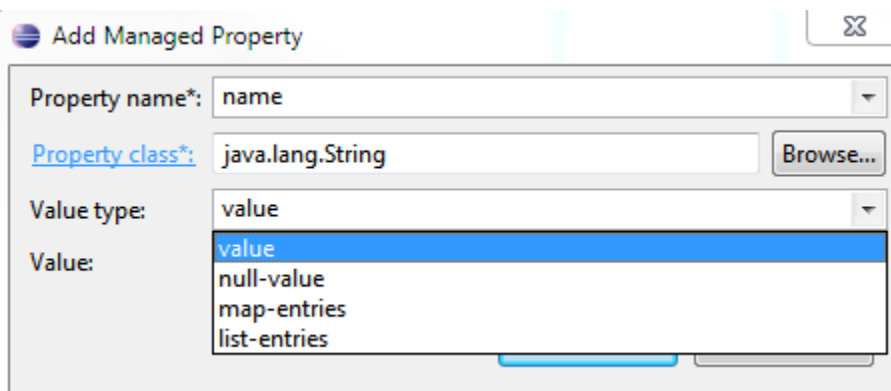
2.3. Initial value
of property

1.3. Scope of Bean

2.2. Type of
property

Add Managed Bean and properties (Eclipse)

Могат да се описват и графично посредством IDE:



Резултат в xml дескриптор

```
<managed-bean>
  <managed-bean-name>student</managed-bean-name>
  <managed-bean-class>models.Student</managed-
bean-class>
  <managed-bean-scope>request</managed-bean-
scope>
  <managed-property>
    <property-name>name</property-name>
    <property-
class>java.lang.String</property-class>
    <null-value />
  </managed-property>
</managed-bean>
```

Може да има стойност по подразбиране

<value>Empty Name</value>

MANAGED BEANS (свойства)

19

- Достъпа до тях се осъществява посредством **EL**
 - ▣ Достъп до свойство
 - Стойността се прочита и се обработва в зависимост от тага (например, визуализира)
 - Трябва да съществува съответен `getXXX()` метод
 - Ако бъде променена от потребителя тя автоматично се променя и в обекта (след като формата бъде изпратена)
 - Трябва да съществува съответен `setXXX()` метод

Пример: достъп до свойство на компонент

```
<h:inputText id= "student_name" value="#{student.name}" />
```

Заб. Достъп до свойството **name** от компонента **student**

MANAGED BEANS

- Достъпа до тях се осъществява посредством **EL**
 - ▣ Достъп до метод
 - След изпращането на формата и задаването на свързаните с нея свойства се извиква дадения метод
 - Трябва да е публичен, да връща String и да е без аргументи:

```
public String doSomething() { ... }
```
 - Изпълнява определена логика
 - Връща резултатен низ (outcome), чрез който **JSF** избира следващата страница

Пример: извикване на методи

Пример: извикване на метод на компонент

```
<h:commandButton value="Change name"  
value="#{student.changeName}" />
```

*Заб. Извикване на метод `String changeName()`; от компонента **student***

```
//class Student...  
//....  
public String changeName() {  
    //check for first letter is capital letter or just make all CAPITAL letters of name  
    this.setName(this.name.toUpperCase()); //  
    return null; //stay on current page  
}
```

*Заб. Извикване на метод `String changeName()`; от компонента **student***

ОБХВАТ

- Определя времето на живот на даден **managed bean**

- Възможни са четири стойности:

- ▣ request

- Съществуват в рамките на заявката: изпращане на заявка и генериране на отговор

- ▣ session

- Съществуват в рамките на сесията на потребителя (между различните заявки от един и същи потребител), докато тя не бъде преустановена (invalidate)

- ▣ application

- В рамките на приложението, достъпни до всички

- ▣ none

- Създава се всеки път, когато се направи обръщение към него, не се записва

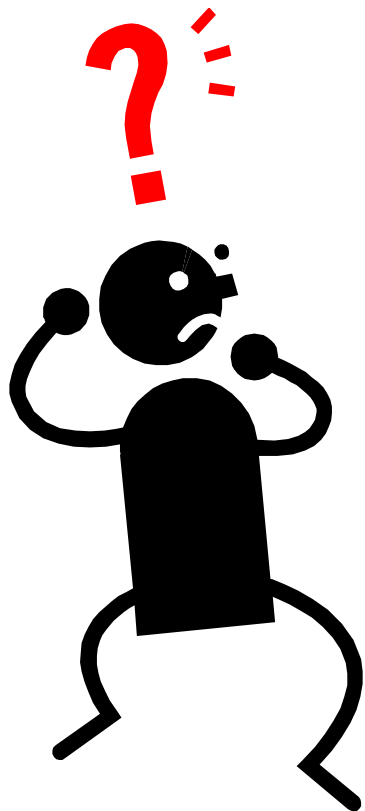
MANAGED BEANS: достъп

- Създадените **managed bean** се съхраняват в асоциативни контейнери
 - ▣ Могат да бъдат достигнати чрез:
 - **EL** от изгледа (view)
 - Чрез контекста на **JSF (FacesContext)**
 - Чрез заявката (**HttpServletRequest**)

В тази лекция бяха разгледани

- Концепция за бизнес компоненти
- Създаване на managed beans
- Обхват на бизнес компонентите
- Пример
- Начини на извличане на данни от компоненти и свързване

Въпроси



Задачи за изпълнение (и демо)

- Задача 1: Решете задачата на адрес <https://netbeans.org/kb/docs/web/jsf20-intro.html>
- **Задача 2***: Разширете примера за персонални данни, като направите управляем компонент (person) с ден/месец/година на раждане, да се взима текущата година (дата) и да се показва на колко години сте - посредством метод checkMyYears()
- ***Забележка 1: * може и като демонстрация в час да се направи (ако не – за домашно по желание);***

Заб.2: Ползвайте за вход компонента h:inputText

```
<h:inputText id="bYear" value="#{person.year}" />
```

Демо

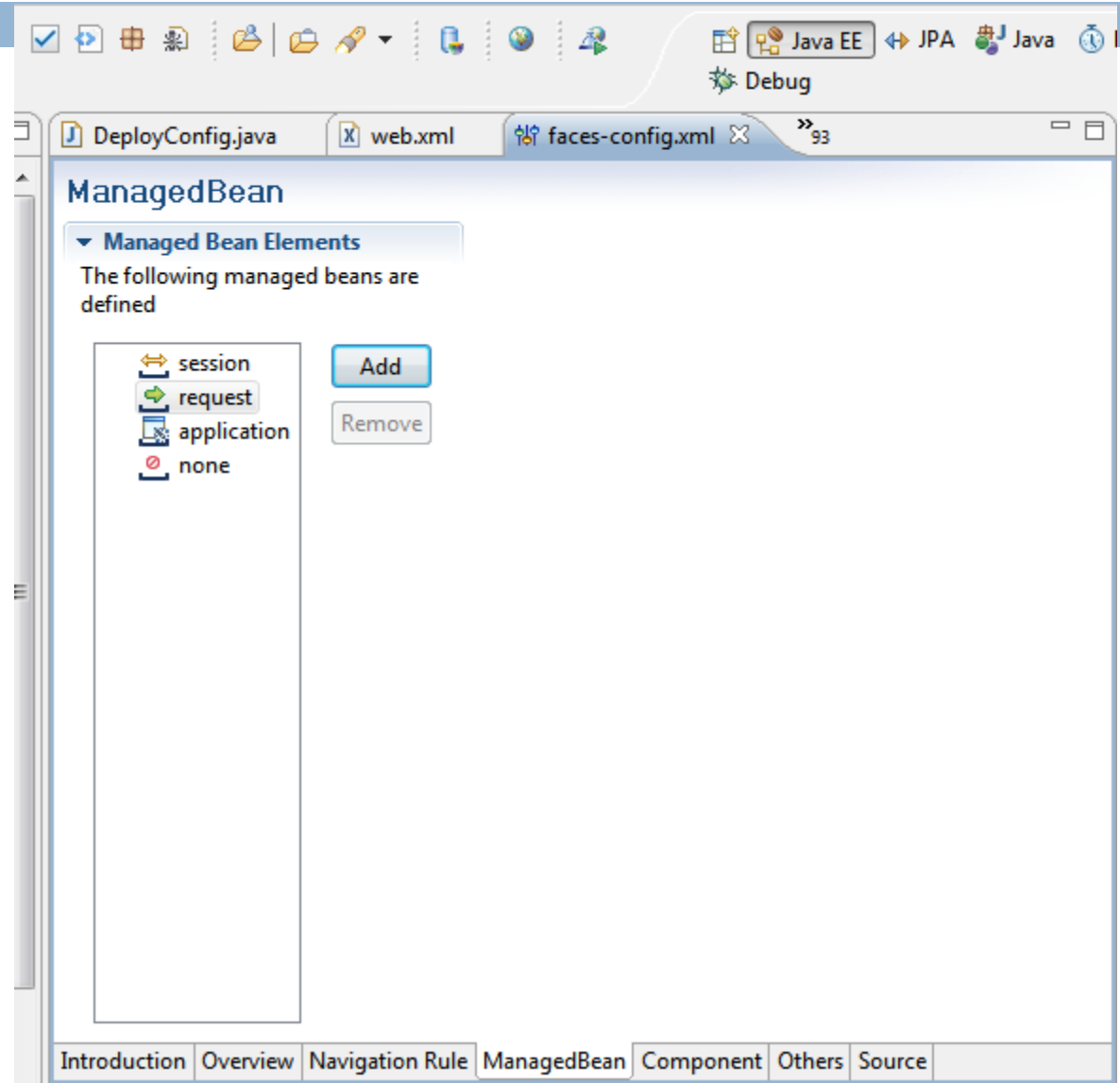
- Задача: Аз съм студент и съм решил да предам домашно по inetAndWww....
 - ▣ - моля въведете година
 - ▣ - въведете преподавател
 - ▣ - въведете номер на упражнение (седмица)
 - ▣ - въведете вашият факултетен номер;
- При изпращане на данните да се



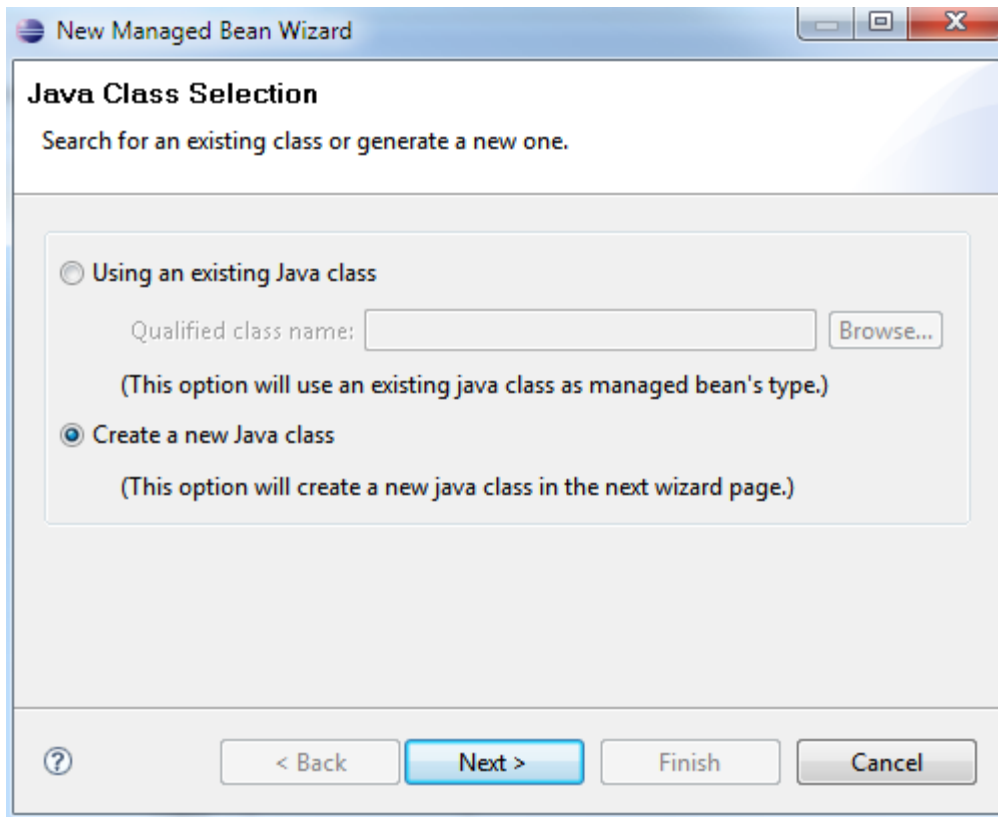
Eclipse JSF project

Managed beans

Step 1: Select type and click 'Add'



Step 2: Create managed bean



The screenshot shows a Java IDE window titled "New Managed Bean Wizard". The main heading is "Java Class Selection" with the instruction "Search for an existing class or generate a new one." There are two radio button options. The first option, "Using an existing Java class", is unselected and includes a text field for "Qualified class name:" and a "Browse..." button, with a note below stating "(This option will use an existing java class as managed bean's type.)". The second option, "Create a new Java class", is selected and includes a note below stating "(This option will create a new java class in the next wizard page.)". At the bottom, there is a help icon (?), and four buttons: "< Back", "Next >" (highlighted in blue), "Finish", and "Cancel".

New Managed Bean Wizard

Java Class Selection
Search for an existing class or generate a new one.

☐ Using an existing Java class

Qualified class name:

(This option will use an existing java class as managed bean's type.)

☒ Create a new Java class

(This option will create a new java class in the next wizard page.)

New Managed Bean Wizard

Java Class

Create a new Java class.

Source folder: DemoJSFBean/src Browse...

Package: models Browse...

☐ Enclosing type: Browse...

Name: Student

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

? < Back Next > Finish Cancel

Result:

The screenshot shows a web IDE with several tabs: DeployConfig.java, FedoraClientHelper.j, WaitCursorEventQueue, web.xml, and *faces-config.xml (selected). The main window is titled "ManagedBean" and contains two main sections.

Managed Bean Elements
The following managed beans are defined

- session
- request
- student
- application
- none

Buttons: Add, Remove

Managed Bean
This section describes general configuration of this managed bean

Managed Bean name*: student

Managed Bean class*: models.Student Browse...

Managed Bean scope*: request

Initialization
You can initialize the managed bean's properties or itself if it is a subclass of java.util.Map or java.util.List

Managed Bean class type: ☒ General class ☐ Map ☐ List

Name	Class	Value

Buttons: Add..., Edit..., Remove

Bottom navigation: Introduction | Overview | Navigation Rule | ManagedBean | Component | Others | Source