# JSF 2

24/04/2017

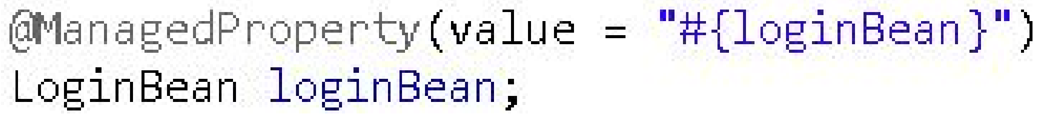
Walid YAICH
walid.yaich@esprit.tn
Bureau E204

## Plan

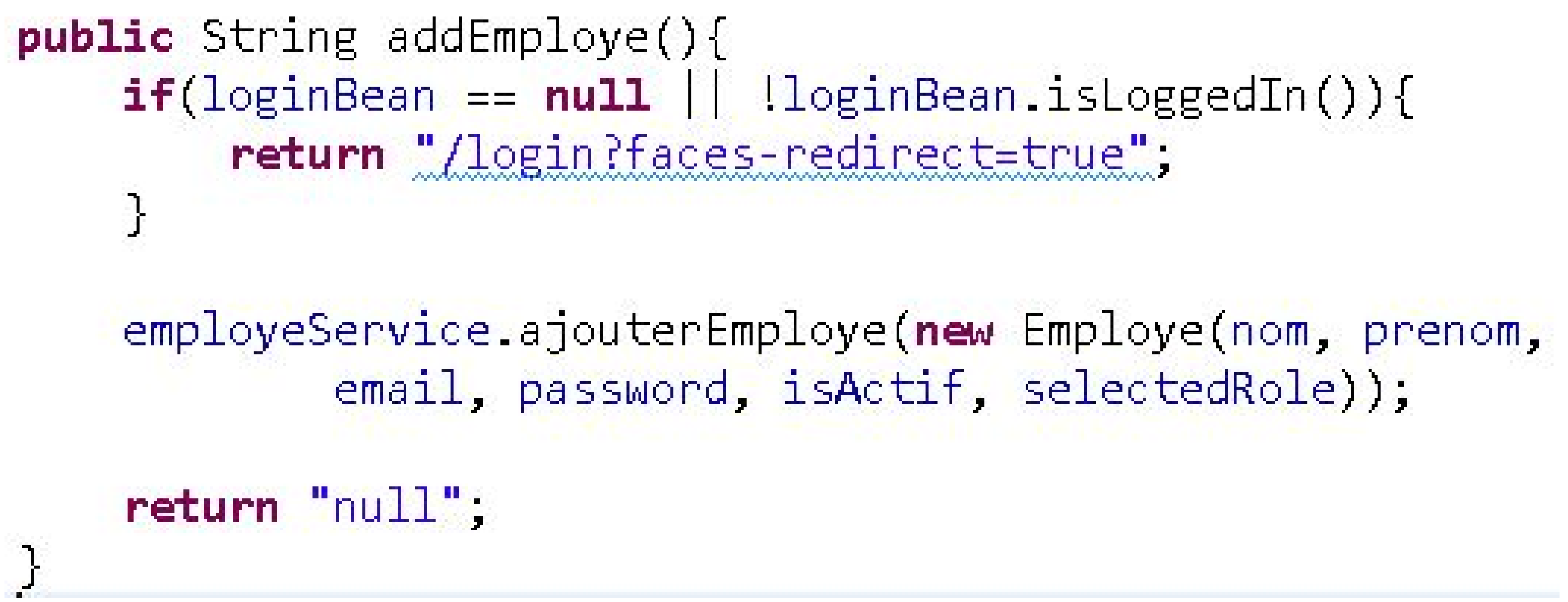
- Sécuriser une méthode
- Sécuriser une page - Filter
- Validator
- Custom validator
- Standard Converter (implicit and explicit)
- Custom converter
- JSF lifecycle

# Sécuriser une méthode

Faire un logout, aller directement sur http://localhost:18080/imputation-web/pages/admin/welcome.jsf
Ajouter un employé ⇒ **bug !**

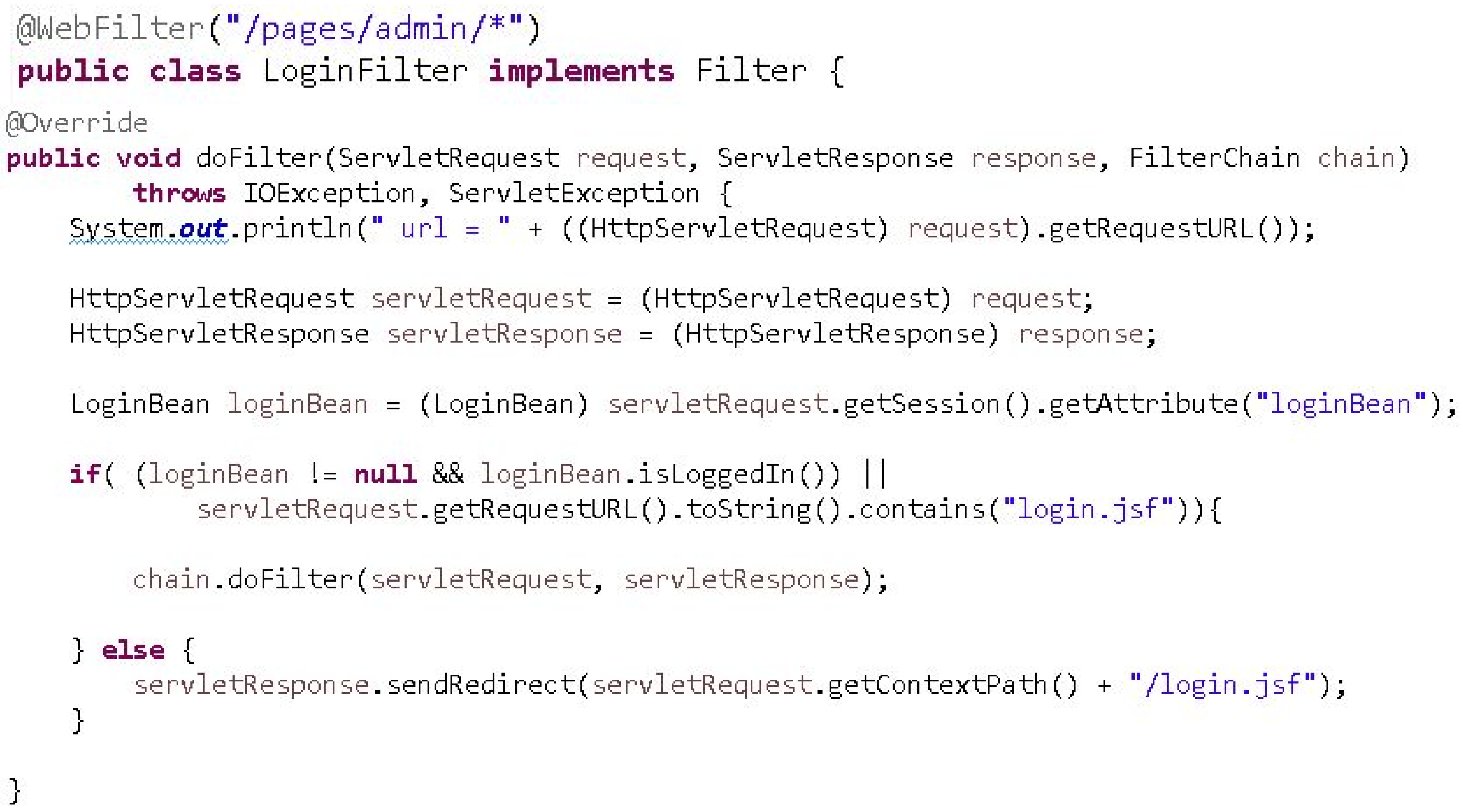


```
@ManagedProperty(value = "#{loginBean}")
LoginBean loginBean;
```



```
public String addEmploye(){
    if(loginBean == null || !loginBean.isLoggedIn()){
        return "/login?faces-redirect=true";
    }

    employeService.ajouterEmploye(new Employe(nom, prenom,
            email, password, isActif, selectedRole));

    return "null";
}
```

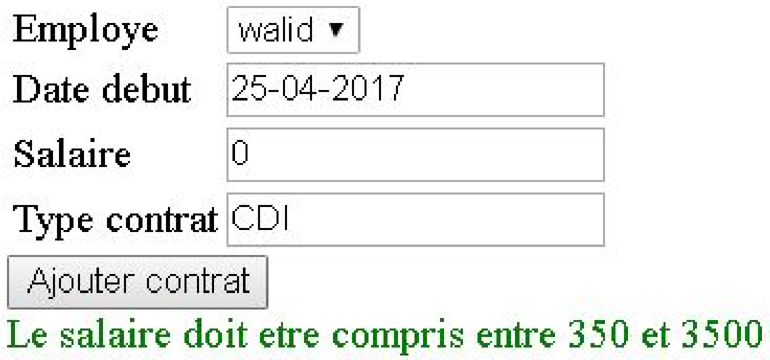
# Sécuriser une page - Filter



```
@WebFilter("/pages/admin/*")
public class LoginFilter implements Filter {

@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
    System.out.println(" url = " + ((HttpServletRequest) request).getRequestURL());

    HttpServletRequest servletRequest = (HttpServletRequest) request;
    HttpServletResponse servletResponse = (HttpServletResponse) response;

    LoginBean loginBean = (LoginBean) servletRequest.getSession().getAttribute("loginBean");

    if( (loginBean != null && loginBean.isLoggedIn()) ||
            servletRequest.getRequestURL().toString().contains("login.jsf")){

        chain.doFilter(servletRequest, servletResponse);

    } else {
        servletResponse.sendRedirect(servletRequest.getContextPath() + "/login.jsf");
    }

}
```

# Validator

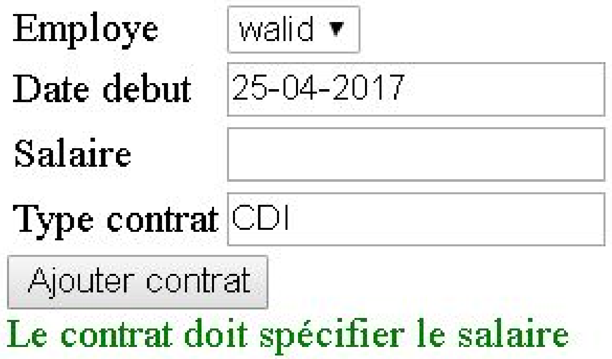
```
                    <h:message for="validateSalaire" style="color:green" />

<h:inputText id="validateSalaire"
             value="#{contratBean.salaire}"
             required="true"
             requiredMessage="Le contrat doit spécifier le salaire"
             validatorMessage="Le salaire doit etre compris entre 350 et 3500">
    <f:validateDoubleRange minimum="350" maximum="3500" for="validateSalaire"/>
</h:inputText>
```



| | |
|---|---|
| Employe | walid |
| Date debut | 25-04-2017 |
| Salaire | 0 |
| Type contrat | CDI |

Ajouter contrat

Le salaire doit etre compris entre 350 et 3500



| | |
|---|---|
| Employe | walid |
| Date debut | 25-04-2017 |
| Salaire | |
| Type contrat | CDI |

Ajouter contrat

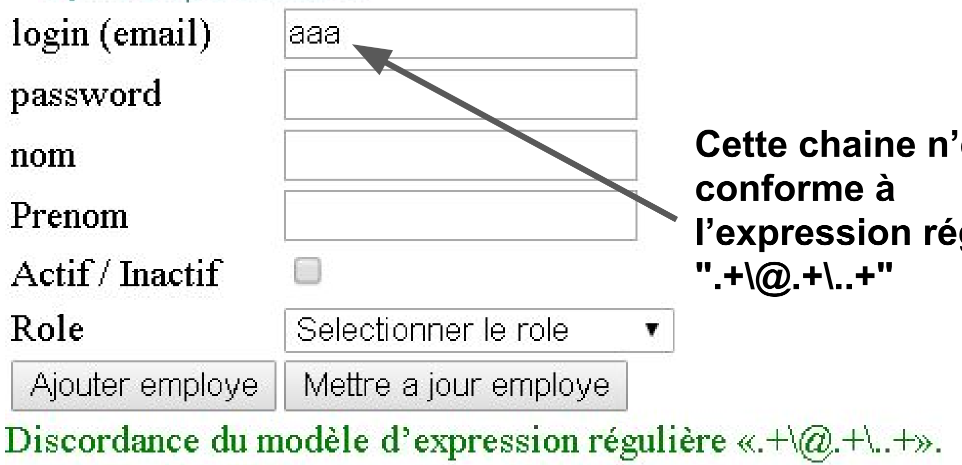
Le contrat doit spécifier le salaire

# Validator

```
<h:message for="validationEmail" style="color:green" />
```

Le message d'erreur généré par l'inputText va être affiché dans le <h:message/>.

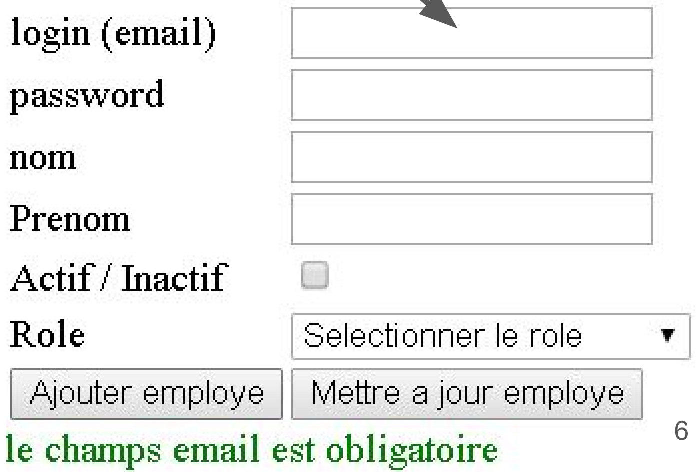
```
<h:inputText id="validationEmail"
             value="#{employeBean.email}"
             required="true"
             requiredMessage="le champs email est obligatoire">
    <f:validateRegex pattern=".+\@.+\..+" />
</h:inputText>
```



login (email) aaa
password
nom
Prenom
Actif / Inactif
Role Selectionner le role

Discordance du modèle d'expression régulière «.+\@.+\..+».

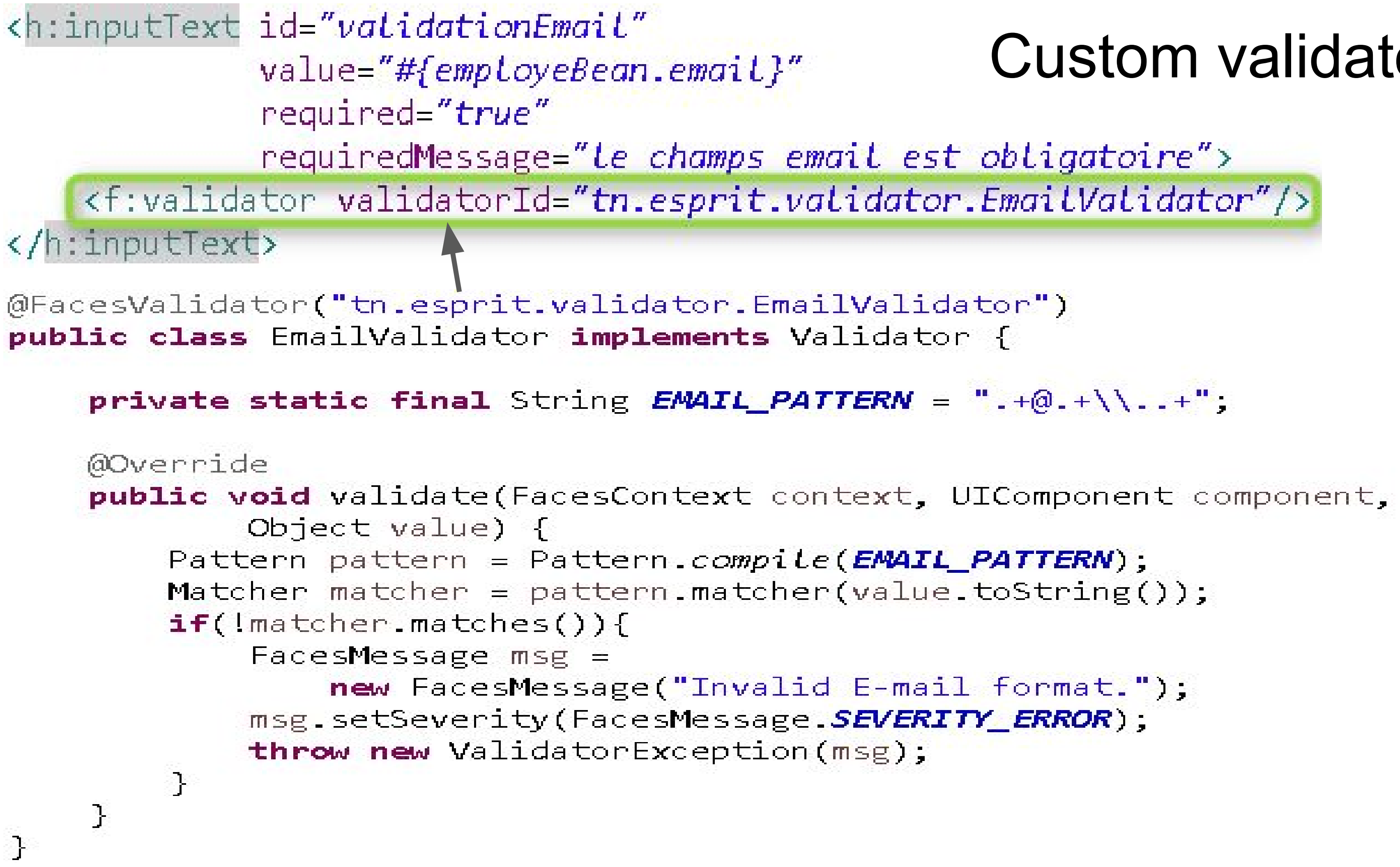
**Cette chaine n'est pas conforme à l'expression régulière ".+\@.+\..+"**



login (email)
password
nom
Prenom
Actif / Inactif
Role Selectionner le role

le champs email est obligatoire

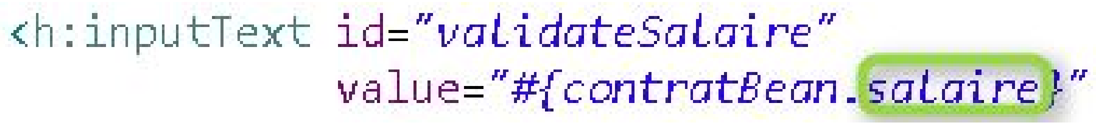
# Custom validator



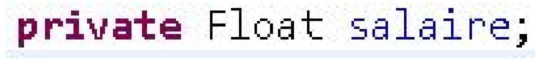
```
<h:inputText id="validationEmail"
             value="#{employeBean.email}"
             required="true"
             requiredMessage="le champs email est obligatoire">
    <f:validator validatorId="tn.esprit.validator.EmailValidator"/>
</h:inputText>
```

```
@FacesValidator("tn.esprit.validator.EmailValidator")
public class EmailValidator implements Validator {

    private static final String EMAIL_PATTERN = ".+@.+\\..+";

    @Override
    public void validate(FacesContext context, UIComponent component,
            Object value) {
        Pattern pattern = Pattern.compile(EMAIL_PATTERN);
        Matcher matcher = pattern.matcher(value.toString());
        if(!matcher.matches()){
            FacesMessage msg =
                new FacesMessage("Invalid E-mail format.");
            msg.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(msg);
        }
    }
}
```

# Standard Converter

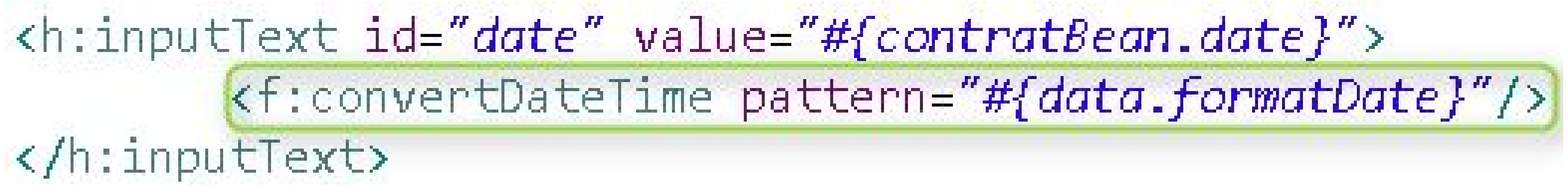


```
<h:inputText id="validateSalaire"
             value="#{contratBean.salaire}"
```



```
private Float salaire;
```

**Implicit converter** : FloatConverter – For Conversion between String and java.lang.Float



```
<h:inputText id="date" value="#{contratBean.date}">
          <f:convertDateTime pattern="#{data.formatDate}"/>
</h:inputText>
```

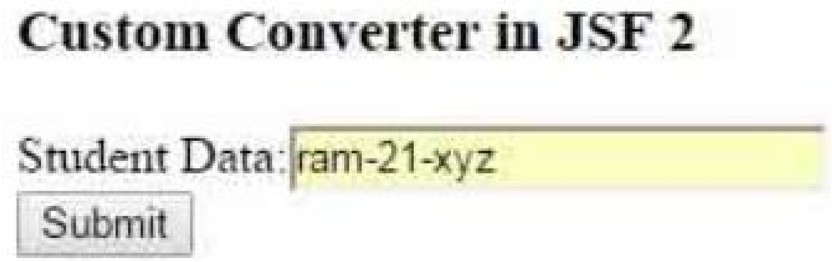
**Explicit converter** : DateTimeConverter – For Conversion between String and java.util.Date

## Standard Converters

- **BigDecimalConverter** – For Conversion between String and java.math.BigDecimal
- **BigIntegerConverter** – For Conversion between String and java.math.BigInteger
- **BooleanConverter** – For Conversion between String and java.lang.Boolean
- **ByteConverter** – For Conversion between String and java.lang.Byte
- **CharacterConverter** – For Conversion between String and java.lang.Character
- **DateTimeConverter** – For Conversion between String and java.util.Date
- **DoubleConverter** – For Conversion between String and java.lang.Double
- **FloatConverter** – For Conversion between String and java.lang.Float
- **IntegerConverter** – For Conversion between String and java.lang.Integer
- **LongConverter** – For Conversion between String and java.lang.Long
- **NumberConverter** – For Conversion between String and java.lang.Number
- **ShortConverter** – For Conversion between String and java.lang.Short

# Custom converter

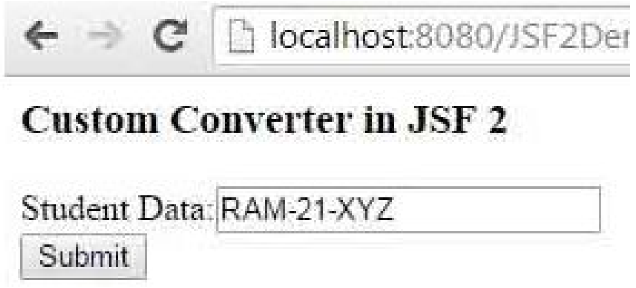
Appui sur Submit ⇒ appel a **getAsObject** du converter



**Custom Converter in JSF 2**

Student Data: ram-21-xyz

Demander la page ⇒ appel a **getAsString** du converter



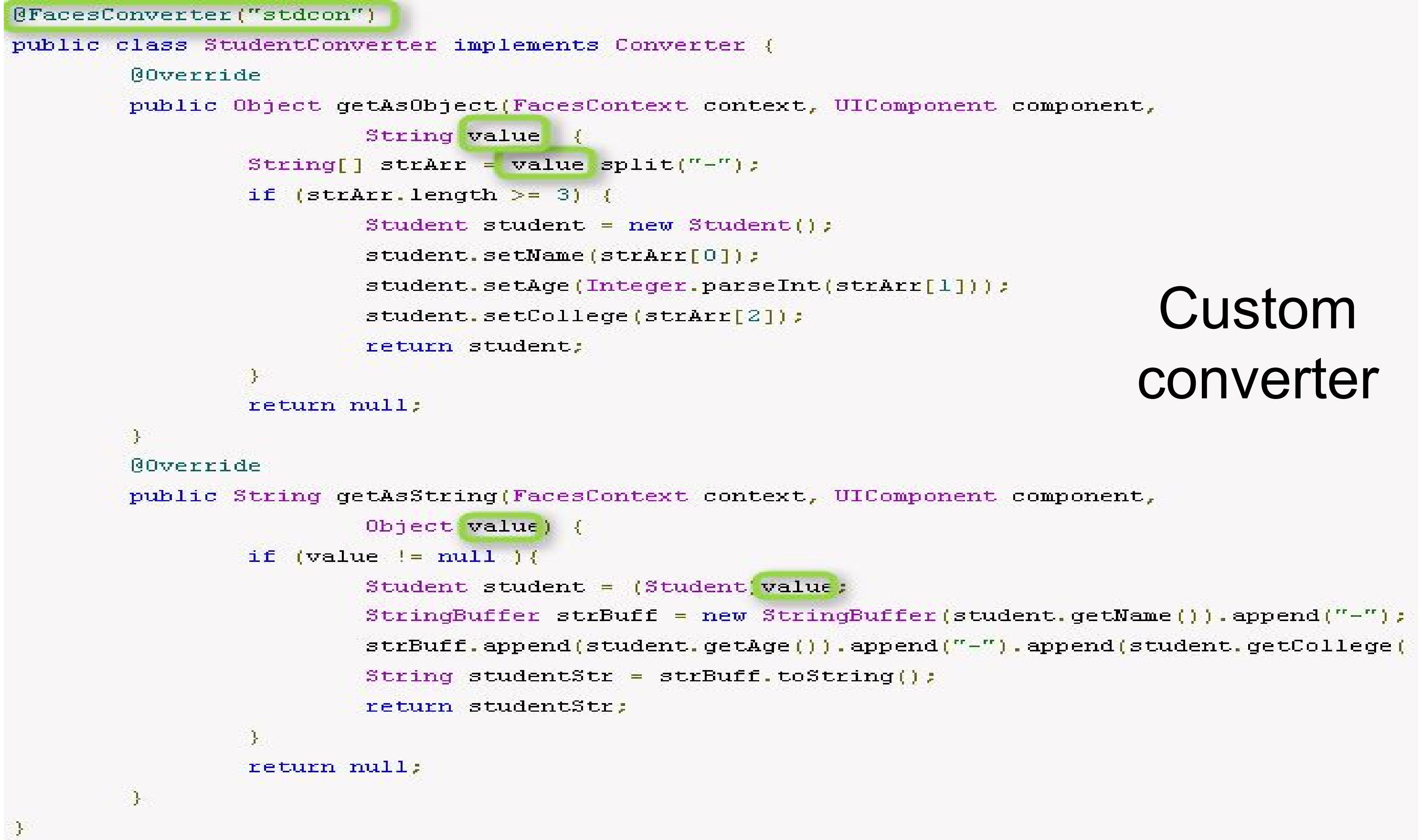
**Custom Converter in JSF 2**

Student Data: RAM-21-XYZ

```
<h:inputText value="#{studentForm.student}" converter="stdcon"/>
```

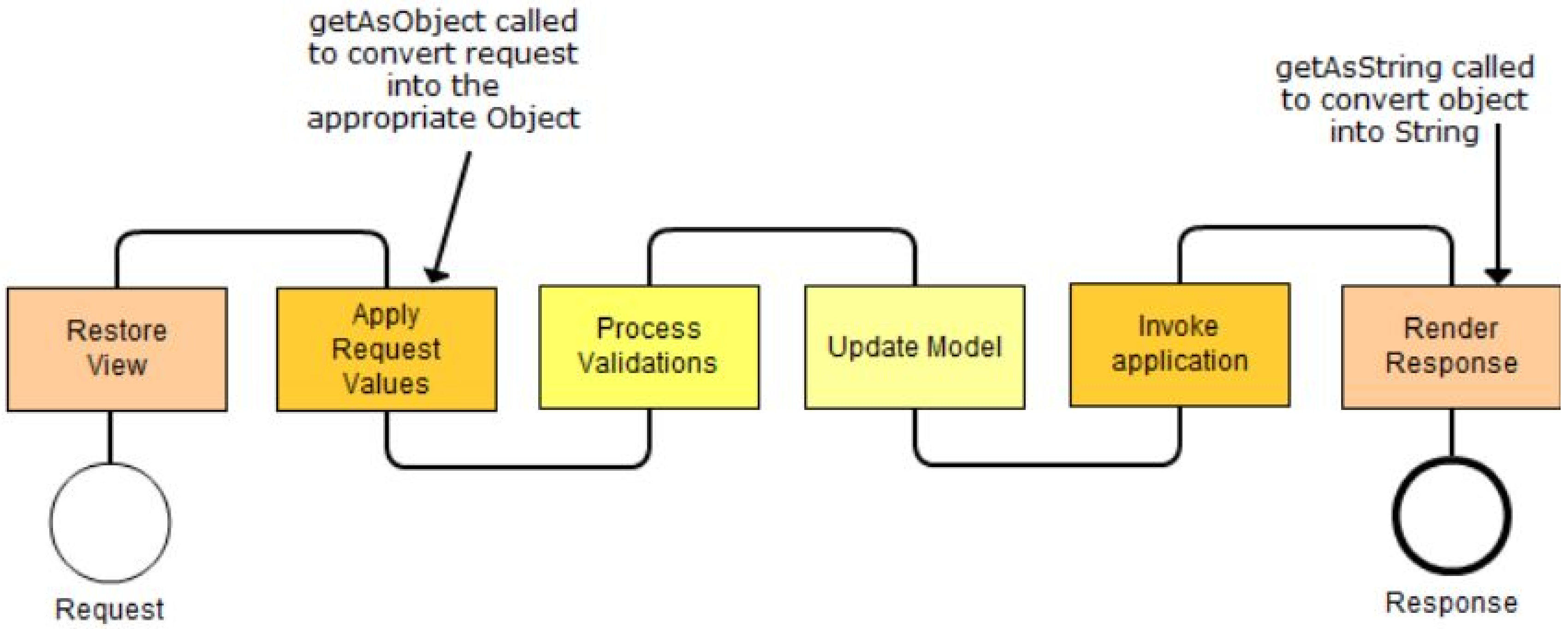
**C’est un object** **L’id du converter**

## Custom converter



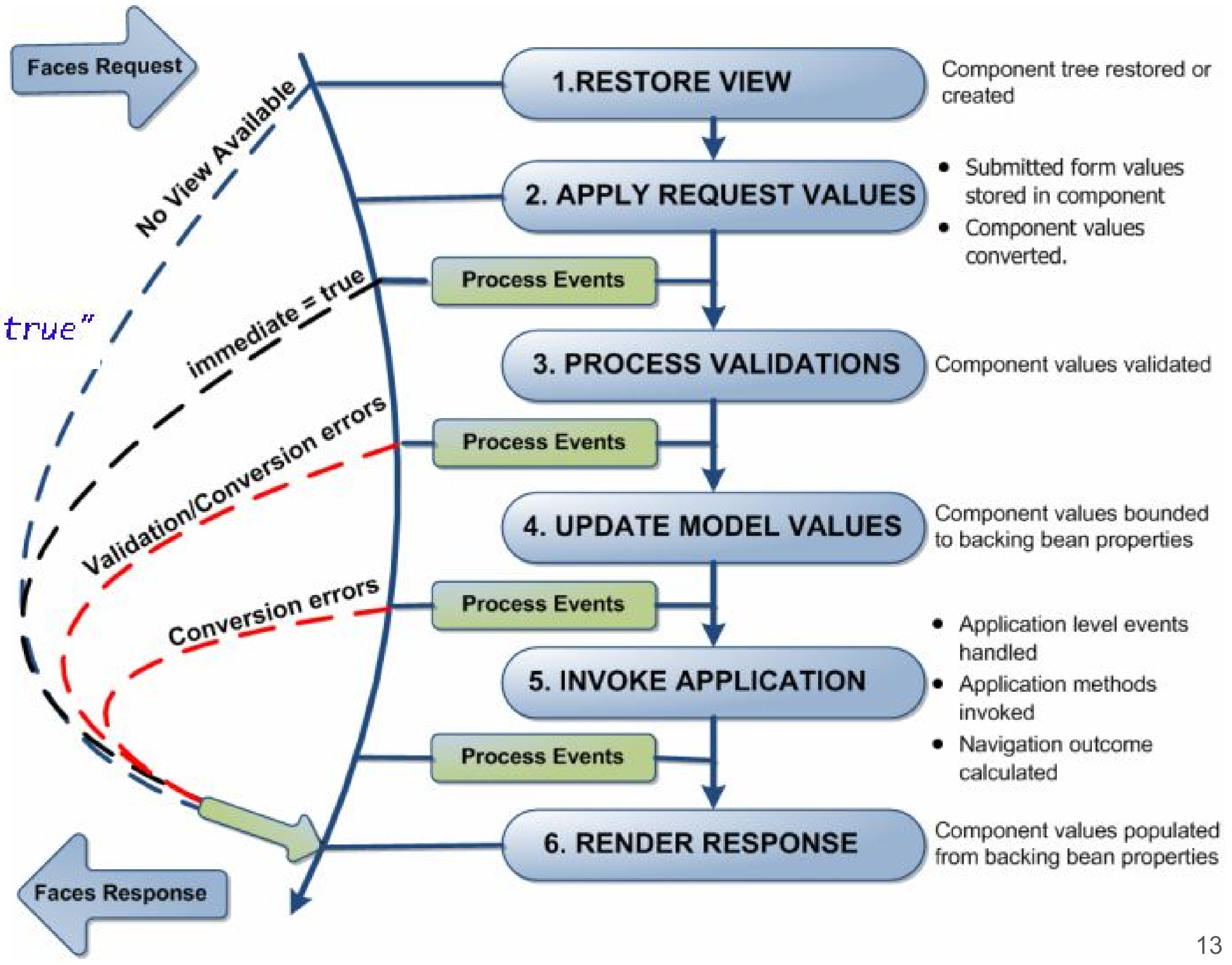
```
@FacesConverter("stdcon")
public class StudentConverter implements Converter {
        @Override
        public Object getAsObject(FacesContext context, UIComponent component,
                        String value) {
                String[] strArr = value.split("-");
                if (strArr.length >= 3) {
                        Student student = new Student();
                        student.setName(strArr[0]);
                        student.setAge(Integer.parseInt(strArr[1]));
                        student.setCollege(strArr[2]);
                        return student;
                }
                return null;
        }
        @Override
        public String getAsString(FacesContext context, UIComponent component,
                        Object value) {
                if (value != null ){
                        Student student = (Student)value;
                        StringBuffer strBuff = new StringBuffer(student.getName()).append("-");
                        strBuff.append(student.getAge()).append("-").append(student.getCollege(
                        String studentStr = strBuff.toString();
                        return studentStr;
                }
                return null;
        }
}
```

# Conversion in JSF Lifecycle


getAsObject called to convert request into the appropriate Object
getAsString called to convert object into String
Restore View
Apply Request Values
Process Validations
Update Model
Invoke application
Render Response
Request
Response

# JSF lifecycle

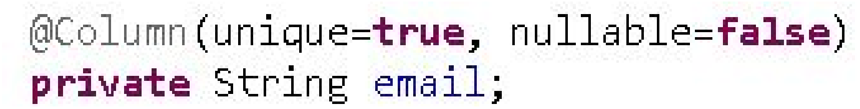
```
<h:commandButton immediate="true"
```



Faces Request
No View Available
immediate = true
Validation/Conversion errors
Conversion errors
Faces Response
1.RESTORE VIEW
Component tree restored or created
2. APPLY REQUEST VALUES
• Submitted form values stored in component
• Component values converted.
Process Events
3. PROCESS VALIDATIONS
Component values validated
Process Events
4. UPDATE MODEL VALUES
Component values bounded to backing bean properties
Process Events
5. INVOKE APPLICATION
• Application level events handled
• Application methods invoked
• Navigation outcome calculated
Process Events
6. RENDER RESPONSE
Component values populated from backing bean properties

# Correction de bugs

Enregistrer 2 utilisateurs avec le même login et même password
Essayer de se logger avec cet utilisateur dupliqué dans la base ⇒ **bug !**



```
@Column(unique=true, nullable=false)
private String email;
```

Dans la page “gerer les employes” on récupère tous les employés et pas les employés de l’entreprise en question (celle de l’admin) ⇒ **bug !**

Ajouter bouton pour reset de la form et de la variable **employeIdToBeUpdated**