Advanced Java

Trainer: Nilesh Ghule

JSP standard actions

* Ideal JSP page have No Java code (Scriptlets / Expressions) in it.
* To reduce / eliminate Java code from JSPs, there are a few options available:
  1. JSP standard actions
     - <jsp: ~~> tags.
  2. Java Beans
     - Java classes with business logic
  3. Third party tag libs
     - JSTL tags (by Sun).
  4. Custom tags.
  5. JSP Expression Lang.

* JSP standard actions.

1. `<jsp:forward page="next.jsp" />`
   - ↳ internally → rd.forward(–,–);
2. `<jsp:include page="next.jsp" />`
   - ↳ internally → rd.include(–,–);
3. `<jsp:param name="_" value="_" />`
   - ↳ child tag of fwd or include.
4. `<jsp:plugin type="applet" width="400" height="300" class="MyApplet.cls">`

   `</jsp:plugin>` → to load applets in JSPs.
5. `<jsp:fallback> Loading.... </jsp:fallback>`
   - child tag of jsp:plugin

Java Beans

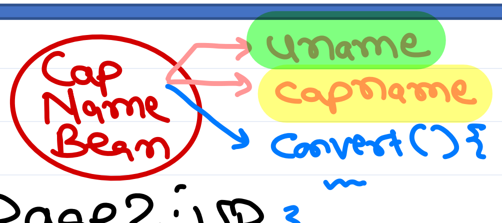
* Java beans are simple java classes

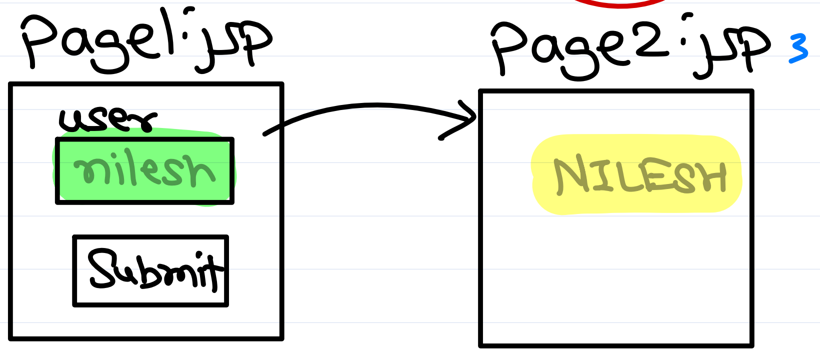
  Bean = POJO + one/more business logic methods.

* Encapsulates business logic & associated with JSP pages.

* Example:

```
class CapNameBean {
    private String uname;
    private String capname;
    public CapNameBean() {
        //ctor
    }
    //... getters/setters
    public void convert() {
        this.capname= this.uname
                      .toUpperCase();
    }
}
```





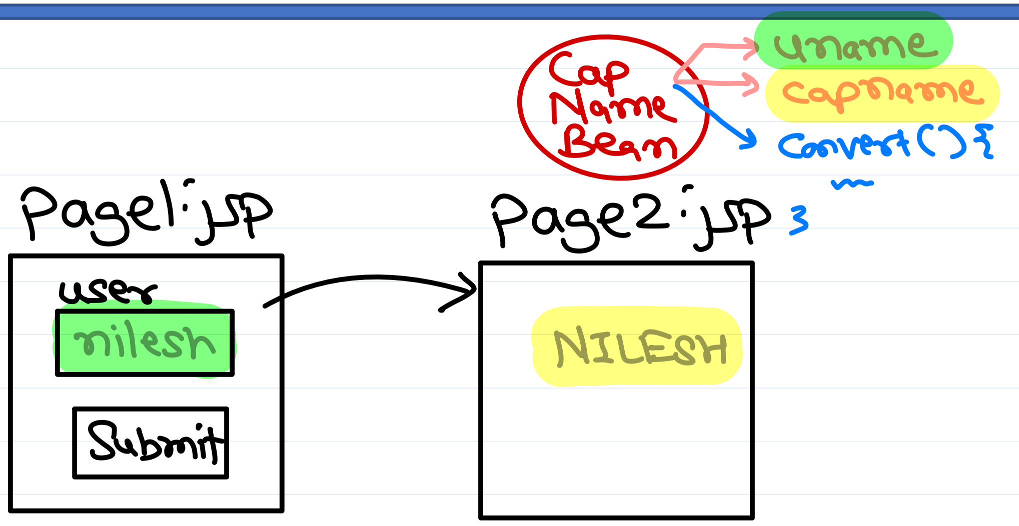
Page2.jsp (better way to use beans is using std actions).

```
<%
    CapNameBean cb=new CapNameBean();
    cb.setUname(request.getParameter("user"));
%>
<% cb.convert(); %>
Result: <%= cb.getCapname() %>
```

Java Beans

* Java beans are simple java classes
  Bean = POJO + one/more business logic methods.

* Encapsulates business logic & associated with JSP pages.



* Example:

```
class CapNameBean {
    private String uname;
    private String capname;
    public CapNameBean() {
        //ctor
    }
    //... getters/setters
    public void convert() {
        this.capname= this.uname
                    .toUpperCase();
    }
}
```

Page2.jsp

```
<jsp:useBean id="cb" class="pkg.CapNameBean"/>
<jsp:setProperty name="cb" property="uname"
        ~~value="nilesh"~~ param="user"/>
<% cb.convert(); %>
Result: <jsp:getProperty name="cb"
                property="capname"/>
```

Java Beans

```
<jsp:useBean id="beanName"
      class="pkg.BeanClassName"
      scope="page|request|session|application" />
```

* JSP Bean Scopes:
  - default scope="page"
* jsp:useBean check if bean with given name is present in given scope.
  if present, same bean is reused.
  if not present, new bean object created & added in that scope.

* Scopes →

(low) page → pageContext attrs : cur req to cur page.
request → request attrs : cur req (fwd/inc)
session → session attrs : cur user all req to all pages.
(high) application → ServletCtx attrs : all user all req to all pages.

```
<jsp:useBean id="obj"
  class="pkg.BeanCls"
  scope="session" />
```

equivalent to

```
obj=session.getAttr("obj");
if(obj==null) {
    obj=new BeanClass();
    session.setAttr("obj", obj);
}
```

Java Beans

```
<jsp:setProperty name="bean" property="field" value="fixed" />
```

initializing field with fixed value

```
bean.setField(fixed);
```

```
<jsp:setProperty name="bean" property="field" param="pname" />
```

initializing field with req param value

```
bean.setField(request.getParameter("pname"));
```

```
<jsp:setProperty name="bean" property="*" />
```

initialize all fields (setters) whose names are matching with corresponding req params.

JSP EL

* Reduce expressions <%= ..... %>
* Syntax: ${expr}
  solved in req handling stage & result added in resp. body.

* Applications:

① arithmetic calculations.

e.g. ${2+3*4} → 14

e.g. ${23 mod 5} → 3

② access objs from diff scopes.

${pageScope.objName}

${requestScope.objName}

${sessionScope.objName}

${applicationScope.objName}

${objName} → auto search obj in all scopes - lowest to highest.

③ access obj getters

${objName.fieldName}

↳ objName.getFieldName()

④ access obj methods

${objName.method()}

⑤ implicit EL objects.

${cookie.cname}

${initParam.pname}

${param.pname}

${paramValues.pname} — req params

req headers

${header.hname}

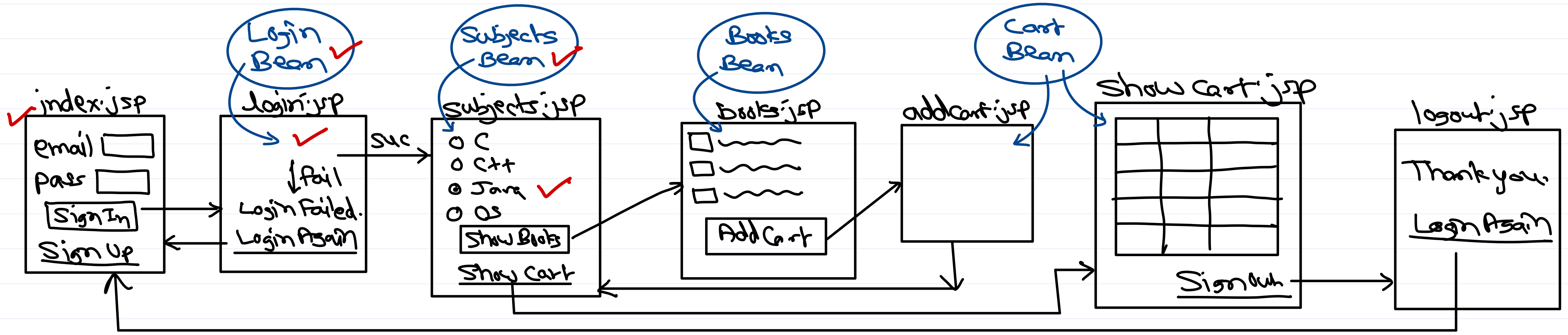
${headerValues.hname}

${pageContext,..}

JSTL

* Popular tag lib by Sun Microsystem.
* Five components :
  1. core → programming, redirection, url, ...
  2. fmt → format date-time & numbers
  3. functions → util fns for strings, ...
  4. sql → execute sql queries on db.
  5. xml → xml generation & parsing, ...

BookShop using JSP — Customer side



Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>