Como hacer una sesión en JSF de manera simple:

¿Qué es jsf?

JavaServer Faces (**JSF**) es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. **JSF** usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL.

JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos librerías de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.
- Beans administrados.

La especificación de JSF fue desarrollada por la Java Community Process como *JSR 127*, que definía JSF 1.0 y 1.1, y *JSR 252* que definía JSF 1.2.

¿Qué es una sesion?

En informática, en particular en redes informáticas, una sesión es la duración de una conexión empleando una capa de sesión de un protocolo de red, o la duración de una conexión entre un usuario (el agente) y un servidor, generalmente involucrando el intercambio de múltiples paquetes de datos entre la computadora del usuario y el servidor. Una sesión es típicamente implementada como una capa en un protocolo de red (por ejemplo, telnet y FTP).

En los casos de los protocolos de transporte en donde no se implementa una capa de sesión formal (por ejemplo, UDP), o en donde las sesiones en la capa de sesión son generalmente de una vida corta (por ejemplo, HTTP), las sesiones pueden ser mantenidas por un programa de más alto nivel, usando algún método. Por ejemplo, un intercambio HTTP entre un navegador y un servidor remoto, puede incluir una cookie, que permite mantener una "sesión", con su identificador propio, datos del usuario, sus preferencias, etc.

El login es la opción y acción (logging in) de iniciar una sesión, generalmente empleando un nombre de usuario y contraseña.

USO DE SESIONES

las sesiones se usan como modo de determinar quien esta dentro de tu Web, y esto porque el protocolo HTTP no te dice nada acerca de quien o quienes navegan por tu site.
Con las sesiones puedes validar un usuario y guárdalas en variables que luego puedes ir pasándola por las paginas de tu Web, de esta forma sabrás si el usuario esta autorizado a realizar determinadas consultas o lo que quieras exponer como información.
Normalmente estas variables se guardan en coockies, que fueron inventadas por Netscape para guardar información del estado de navegación (por las carencias obvias del http).
Otra forma es pasarlo por los métodos get o post en el URL (esto si el cliente no acepta coockies).
Cabe recalcar que el asunto de las coockies tiene su riesgo porque pueden guardar información confidencial como tarjetas de crédito, datos personales, etc. y los hackers siempre buscan sacar información de las coockies para atacar un servidor y apropiarse de datos confidenciales.

¿Cómo hacerlo con JSF?

Al contrario de lo que mucha gente supone esto es mas sencillo que en muchos otros lenguajes, ya que solo consiste en preguntar por la existencia de un dato en el bean de entidad.

Antes de ver el ejemplo se ponen las tags de jsf para poder trabajar.

La tabla muestra las core tags de jsf, que es donde uno tendrá que poner cualquier elemento del Framework, la descripción está en ingles.

Tag	Description
view	Creates the top-level view
subview	Creates a subview of a view
facet	Adds a facet to a component
attribute	Adds an attribute (key/value) to a component
param	Adds a parameter to a component
actionListener	Adds an action listener to a component
valueChangeListener	Adds a valuechange listener to a component
converter	Adds an arbitrary converter to a component
convertDateTime	Adds a datetime converter to a component
convertNumber	Adds a number converter to a component
validator	Adds a validator to a component
validateDoubleRange	Validates a double range for a component's value
validateLength	Validates the length of a component's value
validateLongRange	Validates a long range for a component's value
loadBundle	Loads a resource bundle, stores properties as a Map
selectitems	Specifies items for a select one or select many component
selectitem	Specifies an item for a select one or select many component
verbatim	Adds markup to a JSF page

Se añaden las HTML tags, que son las que se utilizan dentro de un core tag, también su descripción está en ingles, a pesar de ellos, su comprensión de sencilla.

Tag	Description
form	HTML form
inputText	Single-line text input control
inputTextarea	Multiline text input control
inputSecret	Password input control

Tag	Description
inputHidden	Hidden field
outputLabel	Label for another component for accessibility
outputLink	HTML anchor
outputFormat	Like outputText, but formats compound messages
outputText	Single-line text output
commandButton	Button: submit, reset, or pushbutton
commandLink	Link that acts like a pushbutton
message	Displays the most recent message for a component
messages	Displays all messages
graphicImage	Displays an image
selectOneListbox	Single-select listbox
selectOneMenu	Single-select menu
selectOneRadio	Set of radio buttons
selectBooleanCheckbox	Checkbox
selectManyCheckbox	Set of checkboxes
selectManyListbox	Multiselect listbox
selectManyMenu	Multiselect menu

panelGrid	HTML table
panelGroup	Two or more components that are laid out as one
dataTable	A feature-rich table control
column	Column in a dataTable

Ahora que conocemos todas las etiquetas necesarias para trabajar en JSF, podemos trabajar.

Veamos un ejemplo sencillo:

1- lo primero es tener algún formulario para login

```
<f:view>
    <h:form id="primero">
        <p>Login: <h:inputText value="#{UserBean.login}" id="login" required="true">



        </h:inputText>
<h:message for="login" style="color:red" />
</p>
        <p>Pass: <h:inputSecret  value="#{UserBean.pass}"id="pass" required="true"/>
            <h:message for="pass" style="color:red" /> </p>
        <h:commandButton  value="enviar" action="#{UserBean.ir}" />
    </h:form>
</f:view>
```

Las etiquetas de inputText tienen un valor algo extraño, el cual es, #{UserBean.login}
valor que hacer referencia a un atributo del bean de entidad llamado UserBean.

El commandButton lanza su acción a UserBean.ir en este caso los comandbutton hacen referencia a un String, dependiendo de esta se tomaran las reglas de navegación, cosa que analizaremos un poco mas adelante.

Ahora es el turno de revisar el UserBean , el cual es un managed bean con la única diferencia es que en su scope dice session en vez de request.

Este java class (managed bean) debe tener los atributos que se apuntan en el archivo JSP que lo llama, además un método con nombre "ir" que devuelva un String (si no se quiere ningún método podría eventualmente ser un String)

```
private String pass;
private String login;
```

```
/** Creates a new instance of UserBean */
public UserBean() {
}

public String getPass() {
    return pass;
}

public void setPass(String pass) {
    this.pass = pass;
}

public String getLogin() {
    return login;
}

public void setLogin(String login) {
    this.login = login;
}
```

Los métodos de get y set de cada atributo se crean con la función refactor de NetBeans.

A continuación se muestra un ejemplo de un método “ir”

```
    public String ir() {
      [redacted]
         String pss=this.pass;
         String lgin=this.login;
         String klaex="error";
         try {
             InitialContext ic = new InitialContext();
//en esta parte es donde ponemos el Nombre
//de JNDI para que traiga el datasource

             [redacted]
             Statement st = con.createStatement();
             System.out.println("Se ha realizado con exito la conexión a MySQL");
//el resultSet es el encargado de traer los datos de la consulta
             ResultSet rs = st.executeQuery([redacted]

             [redacted]
rs.next();
klaex=rs.getString("rut");
             this.setRut(klaex);
[redacted]
             this.setNombres(nom);
             this.setAp(apep);
             this.setAm(apem);
if(klaex.isEmpty())
    klaex= "error";
else klaex= "ir";

         } catch (SQLException ex) {return klaex;

         } catch (NamingException ex) {return klaex;

         } finally {
            try {
                con.close();

            } catch (SQLException ex) {

            }
         }
         return klaex;
    }
```

Solo tomar en cuenta que la variable “klaex” es finalmente un String que puede ser “error” o “ir”, con lo cual se definirá la regla de navegación.

Suponiendo que tenemos esto listo y queremos agregarle session al sitio, basta solo con agregar una línea encima de un JSF core tag, la cual preguntara por un atributo del managed bean

```
<body>
    <f:view>
        <h:form rendered="#{not empty UserBean.rut}">

                    <h:inputHidden value="#{UserBean.rut}" />
 <h:commandButton  value="enviar" action="#{deudbean.deuda}" />

                  <a href="http://thalos-online.blogspot.com/">
                    visita blog Thalos</a><br />

                    <br /><br />
                    <br /><br />
                </div>
                <div class="clearDiv"></div>
            </div>
        </div>


        <!-- Footer Page Content -->
        <div id="footer">
            <div id="footerContent">
                <div id="footerLeftColumn">
                    <br />Thalos.online 2008 <a href="http://thalos-online.blogspot.com/" title="Thalos"><br />
                </div>
                <div id="footerRightColumn">
                 </div>
                <div class="clearDiv"></div>
            </div>
        </div>

    </div>
</div>
 </h:form>
```

La línea de rendered es la que hace la sesión posible, ya que pregunta si el atributo rut tiene algún valor, en caso de tenerlo muestra el código contenido dentro del form.

```
 </h:form>


<h:form rendered="#{ empty UserBean.rut}">



DEBE LOGREARSE

                    <a href="http://thalos-online.blogspot.com/">
                    visita blog Thalos</a><br />

                    <br /><br />
                    <br /><br />
                </div>
                <div class="clearDiv"></div>
            </div>
        </div>


        <!-- Footer Page Content -->
        <div id="footer">
            <div id="footerContent">
                <div id="footerLeftColumn">
                    <br />Thalos.online 2008 <a href="http://thalos-online.blogspot.com/" title="Thalos"><br />
                </div>
                <div id="footerRightColumn">
                 </div>
                <div class="clearDiv"></div>
            </div>
        </div>

    </div>
</div>
</h:form>
  </f:view>
</body>
```

Debe agregarse una sentencia contraria el rendered anterior, o sea el caso en que rut este vacío, y colocar el código que se desea desplegar, cabe señalar que todo debe ir en un mismo view y en el mismo body de la pagina.