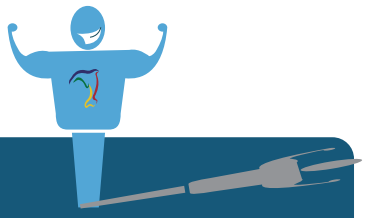
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

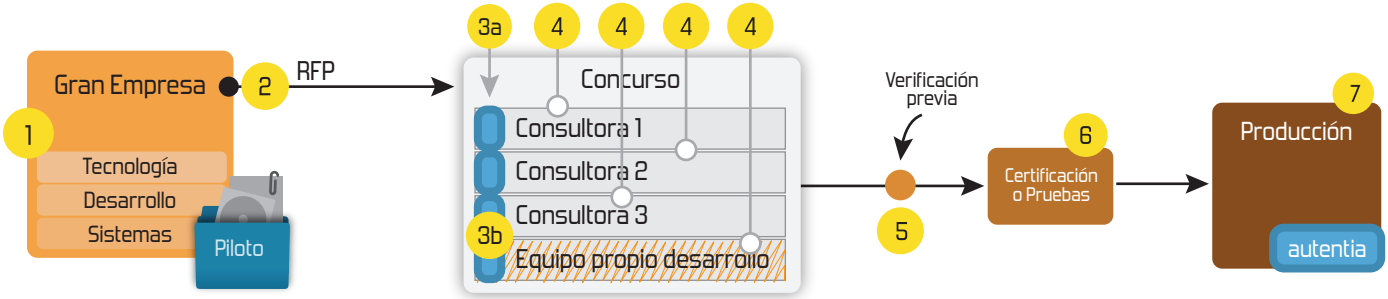
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

DESARROLLADO POR:
Jose Manuel Sánchez Suárez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Fecha de publicación del tutorial: 2011-02-08

Obtención de la fila seleccionada en un datatable con JSF2.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Invocación al método getRowData.
- 4. Enviando el identificador del objeto como parámetro.
- 5. Enviando el objeto como parámetro.
- 6. Con el soporte de Jboss Seam.
- 7. Referencias.
- 8. Conclusiones.

1. Introducción

En este tutorial vamos a analizar las opciones que tenemos disponibles en JSF2 para obtener el registro de la fila seleccionada en un dataTable.

La historia de usuario que queremos cumplir sería similar a esta: **Como Pablo quiero acceder, desde el listado de usuarios, a la edición de los mismos para modificar sus características de acceso**. Pablo es el administrador de usuarios en la aplicación de nuestro hipotético cliente.

Vamos a examinar todas las posibilidades de las que disponemos, que no son pocas, haciendo uso del estandar, y con el soporte de anotaciones de Jboss Seam. Existen más posibilidades si hacemos uso de librerías de componentes visuales de terceros como ICEfaces, RichFaces o Primefaces, pero dichas posibilidades las vamos a examinar en otro tutorial en el que hablaremos de la posibilidad de selección múltiple de filas de un dataTable.

Para el objetivo del tutorial, la selección de un registro de la fila puede suponer navegación a otra vista o no, en función de si el evento de selección responde a un listener o a un action.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 17' (2.93 GHz Intel Core 2 Duo, 4GB DDR3 SDRAM).

- Sistema Operativo: Mac OS X Snow Leopard 10.6.1
- JSF2 (Mojarra 2.0.4)
- Jboss Seam 2.1 y RichFaces 3.3
- Apache Tomcat 7.0.6

3. Invocación al método getRowData.

La primera opción que vamos a examinar es la posibilidad de parametrizar a nivel de dataTable y mediante el atributo binding, la referencia a un objeto del lado del managedBean. En dicho objeto, representado por la clase UIData, se mantiene una referencia del dataTable que se almacena en el árbol de componentes, que vive en el FacesContext durante toda la request.

De este modo, en la vista tendremos algo similar a lo siguiente:

```
<h:dataTable id="usersList"
        var="user"
    value="#{usersView.users}"
    rows="10"
    binding="#{usersView.usersDataTable}">
   <h:column>
       <f:facet name="header">
           <h:outputText id="number_label"
                         value="#{msg['Users.login']}"/>
       </f:facet>
       <h:commandLink actionListener="#{usersView.editUser}" immediate="true">
          <f:ajax render="@form"/>
            <h:outputText id="login"
                          value="#{user.login}"/>
           </h:commandLink>
   </h:column>
     ...
</h:dataTable>
```

Y en el managedBean el método que responde al evento obtiene el registro seleccionado de la siguiente forma:

```
@ManagedBean
@ViewScoped
public class UsersView {
    ...
    private UIData usersDataTable;

    public void setUsersDataTable(UIData usersDataTable) {
        this.usersDataTable = usersDataTable;
    }

    public UIData getUsersDataTable() {
        return usersDataTable;
    }
    ...
    public void editUser(ActionEvent event){
        log.trace("User selected " + ((User) usersDataTable.getRowData()) );
    }
    ...
}

```

Sería indiferente si trabajásemos con un action o con un actionListener, funciona con los dos tipos de evento.

4. Enviando el identificador del objeto como parámetro.

También podemos enviar como parámetro el identificador del usuario, de modo que se añada al mapa de parámetros de la request y obtenerlo desde el método correspondiente del managedBean.

Para ello desde la vista usamos el componente no visual <f:param

```
<h:dataTable id="usersList"
        var="user"
    value="#{usersView.users}"
    rows="10"
    binding="#{usersView.usersDataTable}">
   <h:column>
       <f:facet name="header">
           <h:outputText id="number_label"
                         value="#{msg['Users.login']}"/>
       </f:facet>
       <h:commandLink actionListener="#{usersView.editUser}" immediate="true">
          <f:param name="userLogin" value="#{user.login}" />
          <f:ajax render="@form"/>
            <h:outputText id="login"
                          value="#{user.login}"/>
           </h:commandLink>
   </h:column>
     ...
</h:dataTable>
```

En el método que recibe el evento podemos obtener el parámetro como sigue:

```
@ManagedBean
@ViewScoped
```

```
public class UsersView {
  ...
  private UIData usersDataTable;
  ...
  public void editUser(ActionEvent event){
    final String userLogin =
FacesContext.getCurrentInstance().getExternalContext().getRequestParameterMap().get("userLogi
    selectedUser = FinancialService.getInstance().getByLogin(userLogin);
    log.trace("User selected " + selectedUser );
  }
  ...
}
```

Aquí el inconveniente es que tenemos que volver a obtener el objeto totalmente poblado porque lo que remitimos es su identificador y se trata de un tipo básico.

5. Enviando el objeto como parámetro.

Existe una alternativa a la de remitir el identificador que es la de remitir todo el objeto completo y, para ello tenemos dos posibilidades.

La primera es hacer uso del componente no visual <f:setPropertyActionListener en el que le podemos indicar el valor que tendrá un atributo del managedBean antes de que el evento de acción sea invocado.

```
<h:dataTable id="usersList"
        var="user"
    value="#{usersView.users}"
    rows="10"
    binding="#{usersView.usersDataTable}">
   <h:column>
       <f:facet name="header">
           <h:outputText id="number_label"
                         value="#{msg['Users.login']}"/>
       </f:facet>
       <h:commandLink actionListener="#{usersView.editUser}" immediate="true">
          <f:setPropertyActionListener value="#{user}"
                                target="#{usersView.selectedUser}" />
          <f:ajax render="@form"/>
            <h:outputText id="login"
                      value="#{user.login}"/>
           </h:commandLink>
   </h:column>
     ...
</h:dataTable>
```

Del lado del managedBean solo necesitaremos el atributo con sus métodos set & get.

```
@ManagedBean
@ViewScoped
public class UsersView {

    ...
    private User selectedUser;

    public void setSelectedUser(User selectedUser) {
        this.selectedUser = selectedUser;
    }

    public User getSelectedUser() {
        return selectedUser;
    }
    ...
}
```

La segunda opción, solo podremos hacer uso de ella, si tenemos disponible la versión 2.2 de EL, como en nuestro caso que usamos Tomcat 7.

Consiste en invocar directamente desde Expression Language a un método del managedBean pasándole un parámetro.

```
<h:dataTable id="usersList"
        var="user"
    value="#{usersView.users}"
    rows="10"
    binding="#{usersView.usersDataTable}">
   <h:column>
       <f:facet name="header">
           <h:outputText id="number_label"
                         value="#{msg['Users.login']}"/>
       </f:facet>
       <h:commandLink action="#{usersView.editUser(user)}" immediate="true">
          <f:ajax render="@form"/>
            <h:outputText id="login"
                      value="#{user.login}"/>
           </h:commandLink>
   </h:column>
     ...
</h:dataTable>
```

De este modo, en el managedBean, tendríamos simplemente el método con dicho parámetro de entrada.

```
@ManagedBean
@ViewScoped
public class UsersView {
    ...
  public void editUser(User user){
    selectedUser = user;
    log.trace("User selected " + selectedUser );
  }
    ...
}

```

Esta técnica es la que veníamos usando hasta ahora en los proyectos que tenían el soporte de Jboss Seam que sí permitía invocar a un método del managedBean pasándole parámetors de entrada.

6. Con el soporte de Jboss Seam.

Además de la que hemos comentado anteriormente existe una segunda posibilidad en Jboss Seam consistente en mantener en el managedBean un atributo anotado que tendrá una referencia al dataTable y un segundo atributo, también anotado, que mantiene una referencia a la fila seleccionada.

```
@Name("usersView")
@Scope(ScopeType.SESSION)
public class UsersView {

  @DataModel
  List<User> users;

  @DataModelSelection
  User selectedUser;

  ...
}
```

La vista no requiere de ninguna configuración adicional, desde cualquier método de evento del managedBean podemos acceder al atributo que estará poblado con la entidad de la fila seleccionada.

```
...
<rich:dataTable id="usersList"
          var="user"
      value="#{usersView.users}"
      rows="10">
...
```

7. Referencias.

- http://balusc.blogspot.com/2006/06/using-datatables.html

8. Conclusiones.

Hemos visto que no son pocas las posibilidades que tenemos en nuestra mano, hacer o haber hecho uso de una u otra dependerá de nuestro grado de acercamiento al api, esto es, del grado de madurez que tengamos con JSF.

Un saludo.

Jose

jmsanchez@autentia.com

COMENTARIOS