

ICEmobile : Input Form Tutorial

This page last changed on Jan 15, 2013 by [tyler.johnson](#).

Using <mobi> components to create an input form

This tutorial will demonstrate the use of various ICEmobile input and button components. Our goal is to create a page that will allow us to submit a first and last name, email, phone, date of birth, as well as a receive alerts preference.

Form Input & server side output	
First Name	<input type="text" value="first name"/>
Last Name	<input type="text" value="last name"/>
Email	<input type="text" value="www.email.com"/>
Phone	<input type="text" value="555-2323"/>
Date of Birth	<input type="text"/>
Receive updates?	<input checked="" type="checkbox"/> OFF
Submit Form	
First Name:	
Last Name:	
Email:	
Phone:	
Date of Birth:	
Receive alerts:	false

The source code for this sample can be found [here](#).

Add <mobi:fieldsetGroup> layout component

We will layout our form using the <mobi:fieldsetGroup> and <mobi:fieldsetRow> components. Paste the following into the <h:body> of your ICEmobile page:

```
<h:form>
  <mobi:fieldsetGroup>
    <mobi:fieldsetRow>
      </mobi:fieldsetRow>
    </mobi:fieldsetGroup>
  </h:form>
```

Add <mobi:inputText> for first and last name, email and phone

Our first two fields will be for the user's first and last name, followed by an <h:outputLabel> to render the label and a <mobi:inputText> for the entered value. We have set the type of the input text to 'text' and our placeholder will contain a default value for that field. We will also set **singleSubmit** to true which adds the ability to submit only one component to execute in the JSF lifecycle. You can find a listing of all component attributes in the ICEmobile TLD [here](#). Please add the following inside the <mobi:fieldsetGroup> tag:

```
<mobi:fieldsetRow>
  <h:outputLabel for="firstName" value="First Name" />
  <mobi:inputText id="firstName" value="#{bean.firstName}" type="text" placeholder="first name"
    title="First Name" singleSubmit="true"/>
</mobi:fieldsetRow>

<mobi:fieldsetRow>
```

```

<h:outputLabel for="lastName" value="Last Name" />
<mobi:inputText id="lastName" value="#{bean.lastName}" type="text" placeholder="last name"
    title="Last Name" singleSubmit="true"/>
</mobi:fieldsetRow>

```

Our next two inputText fields will contain the user's email and phone number and will be identical to our first and last name fields except that we will change the type attribute of the inputText. Please paste the following below our last name fieldsetRow:

```

<mobi:fieldsetRow>
    <h:outputLabel for="email" value="Email" />
    <mobi:inputText id="email" value="#{bean.email}" type="email" placeholder="email@email.com"
        title="email" singleSubmit="true"/>
</mobi:fieldsetRow>

<mobi:fieldsetRow>
    <h:outputLabel for="phone" value="phone" />
    <mobi:inputText id="phone" value="#{bean.phone}" type="phone" placeholder="555-2323"
        title="phone" singleSubmit="true"/>
</mobi:fieldsetRow>

```

Add **<mobi:dateSpinner>** for date of birth

We will add a field for the user's date of birth using the mobi:dateSpinner. The dateSpinner component displays a popup that allows a user to select a date value. The component must be value bound to a Date object and support converters and timezone assignment. Add the following below our phone fieldsetRow:

```

<mobi:fieldsetRow>
    <h:outputLabel for="date" value="Date of Birth" />
    <mobi:dateSpinner id="date" value="#{bean.dateOfBirth}" title="date" singleSubmit="true"/>
</mobi:fieldsetRow>

```

 Please use the [<mobi:timeSpinner>](#) for time input

Add **<mobi:flipswitch>** for alerts

A flipswitch will be used to toggle if the user would like to receive update notices. The flipswitch component displays a command button that toggles between on and off states. This UI element is common on the iOS platform for toggling on/off settings. Add the following below our date of birth fieldsetRow:

```

<mobi:fieldsetRow>
    <h:outputLabel for="alert" value="Receive updates?" />
    <mobi:flipswitch value="#{bean.alerts}" id="alert" singleSubmit="true" labelOn="ON" labelOff="OFF" />
</mobi:fieldsetRow>

```

Add <mobi:commandButton> for form submit

The <mobi:commandButton> will be used to submit our form to the server. Keep in mind that since we've set singleSubmit to true on our input field components, the user entered values have already been set in the server side bean. This command button will execute an actionListener that will output our field values to the server console. Four types of buttons are allowed default, important, back and attention. Each render with a different style. We will also add an submitNotification component that will block the UI until the actionListener has completed. Please add the following code below our flipSwitch fieldsetRow:

```
<mobi:commandButton actionListener="#{bean.submitForm}" buttonType="default" value="Submit Form"
    submitNotification="submitNotification"/>

<mobi:submitNotification id="submitNotification">
    <h:outputText value="Processing.."/>
</mobi:submitNotification>
```

Add server side output to page

In order to demonstrate singleSubmit and the dynamic updating of page values, we will add the following output below the <mobi:fieldsetGroup> that we added in the above steps:

```
<mobi:fieldsetGroup>
    <mobi:fieldsetRow>First Name: #{bean.firstName}</mobi:fieldsetRow>
    <mobi:fieldsetRow>Last Name: #{bean.lastName}</mobi:fieldsetRow>
    <mobi:fieldsetRow>Email: #{bean.email}</mobi:fieldsetRow>
    <mobi:fieldsetRow>Phone: #{bean.phone}</mobi:fieldsetRow>
    <mobi:fieldsetRow>Date of Birth:
        <h:outputText value="#{bean.dateOfBirth}">
            <f:convertDateTime pattern="d-M-yyyy" />
        </h:outputText>
    </mobi:fieldsetRow>
    <mobi:fieldsetRow>Receive alerts: #{bean.alerts}</mobi:fieldsetRow>
</mobi:fieldsetGroup>
```

Add Managed Bean code

Create a class named FormBean.java and paste the following:

```
@ManagedBean
@ViewScoped
public class FormBean {

    private String firstName;
    private String lastName;
    private String email;
    private String phone;
    private Date dateOfBirth;
    private boolean alerts;
```

```
//TODO: GENERATE GETTERS and SETTERS
```

The final step is to add the actionListener that will fire when our <mobi:commandButton> is clicked. Please paste the following into Bean.java:

```
public void submit(ActionEvent e) {  
    System.out.println("First Name: " + firstName);  
    System.out.println("Last Name: " + lastName);  
    System.out.println("Email: " + email);  
    System.out.println("Phone: " + phone);  
    System.out.println("Date of Birth: " + dateOfBirth);  
    System.out.println("Receive alerts: " + alerts);  
}
```

Run the application

We now have an input form with single submit enabled. When this attribute is set to true, only the specified component will decode, validate, update its bean model state, and trigger events, but a full render phase will occur. Here is what our form will look like:

The form consists of five input fields and one checkbox. The first four fields are standard text inputs with placeholder values. The fifth field is a date input with a calendar icon. The checkbox is currently off. A large blue button at the bottom is labeled 'Submit Form'.

When you tab through the form fields and enter data, the server side values are shown below the input components:

First Name:
Last Name:
Email:
Phone:
Date of Birth:
Receive alerts: false

Source

[icemobile-form-tutorial.zip](#)