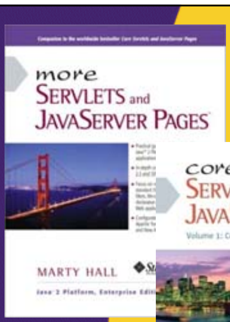
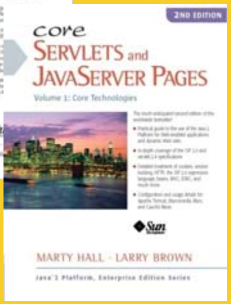
PrimeFaces: Number Input

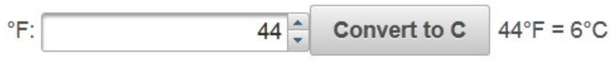
Originals of slides and source code for examples: http://www.coreservlets.com/JSF-Tutorial/primefaces/
Also see the JSF 2 tutorial – http://www.coreservlets.com/JSF-Tutorial/jsf2/
and customized JSF2 and PrimeFaces training courses – http://courses.coreservlets.com/jsf-training.html





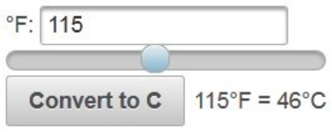
Topics in This Section

- **p:spinner**
  - For collecting int or double with a textfield that has up/down arrows.



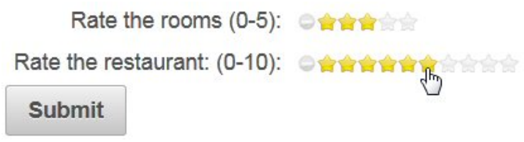
°F: 44 Convert to C 44°F = 6°C

- **p:slider**
  - For collecting int with a slider.



°F: 115 Convert to C 115°F = 46°C

- **p:rating**
  - For collecting int in a narrow range, for use as a rating.

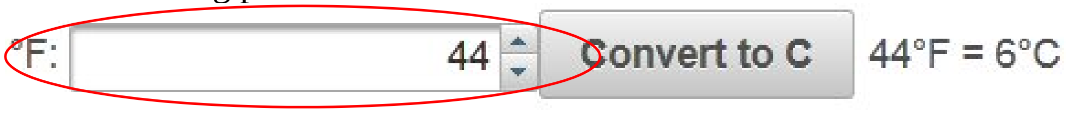


Rate the rooms (0-5):
Rate the restaurant: (0-10):

p:spinner

p:spinner: Overview

- **Appearance and behavior**
  - Textfield with arrow-like buttons for incrementing or decrementing values. User can also type number directly in the textfield.
- **Purpose: for collecting numbers from user**
  - Value is automatically converted to number. UI prevents user from entering an illegal value. Can collect either whole number or floating point number.



- **Spinners vs. sliders: usage**
  - Spinners are better for narrow ranges where you need user to choose a precise value. Sliders are better for wide ranges where similar values (e.g., 117 vs. 118) are treated about the same.

p:spinner: Summary of Most Important Attributes

- **<p:spinner value="…" …/>**
  - value
    - Usually refers to bean property of type double or Double. Can also be int (or Integer) if you also set the stepFactor to be a whole number (or use default stepFactor of 1).
  - min, max
    - Smallest/largest values allowed. Be sure initial value of bean property is in that range (usually min).
  - stepFactor
    - How much the value changes when user clicks on up or down buttons. Default is 1.
  - prefix
    - Value to display at front of number (like a currency symbol). Does not become part of the value.
      - There is also "suffix", but it is less used

Example 1: Whole Number Input

- **Input**
  - Temperature in Fahrenheit
  - UI will collect whole number in range from 32 to 212. Set initial bean value to 32 so it is in this range. No stepFactor given, so default of 1 is automatically used.
- **Output**
  - Temperature in Celsius (rounded to nearest whole number)
- **Command button**
  - No action given, so same page automatically redisplayed. Conversion (Celsius temp) shown on same page as input.

Example 1: Bean

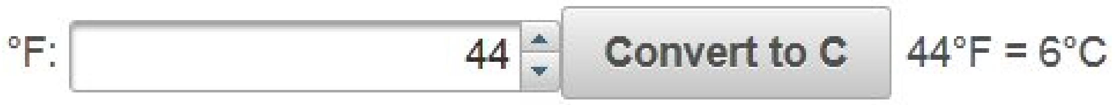
```
public class FahrenheitBean {
  private int f=32;

  public int getF() {
    return (f);
  }

  public void setF(int f) {
    this.f = Math.max(f, -460);  // -459.67 is absolute zero
  }

  public int getC() {
    return((int)((f - 32)*(5.0/9.0)));
  }
}
```

Since we use int, not double, the spinner must use a whole number for the stepFactor (default is 1).

Instead of using @ManagedBean, the bean name is given in faces-config.xml. This lets us use several different instances of the same class in the same page, but with different bean names. If you don't know how to declare beans in faces-config.xml, see lecture on page navigation and faces-config.xml in general JSF2 tutorial at coreservlets.com.

Example 1: Facelets and Results

```
...
<h:form>
   &deg;F:
   <p:spinner min="32" max="212" value="#{fBean1.f}"/>
   <p:commandButton value="Convert to C" ajax="false"/>
   #{fBean1.f}&deg;F = #{fBean1.c}&deg;C
</h:form>
...
```



Example 2: Floating Point Input

- **Input**
  - Temperature in Celsius
  - UI will collect double in range from 0 to 100. stepFactor set to 0.5 so that user can enter whole or half degrees.
- **Output**
  - Temperature in Fahrenheit
- **Command button**
  - No action given, so same page automatically redisplayed. Conversion shown on same page as input.

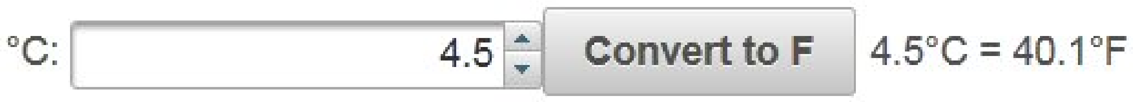
Example 2: Bean

```
@ManagedBean(name="cBean")
public class CelsiusBean {
  private double c;

  public double getC() {
    return(c);
  }

  public void setC(double c) {
    this.c = Math.max(c, -273.15); // -273.15 is abs. zero
  }

  public double getF() {
    return(c*9.0/5.0 + 32);
  }
}
```

The spinner has a stepFactor of 0.5, so we must use double (or Double) instead of int. Make sure initial bean property value is within specified range. Here, it is 0, which is OK.

Example 2: Facelets and Results

```
...
<h:form>
   &deg;C:
   <p:spinner min="0" max="100" stepFactor="0.5"
              value="#{cBean.c}"/>
   <p:commandButton value="Convert to F" ajax="false"/>
   #{cBean.c}&deg;C = #{cBean.f}&deg;F
</h:form>
...
```


°C: 4.5 Convert to F 4.5°C = 40.1°F

Example 3: Prefix

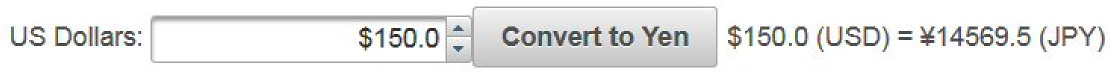
- **Input**
  - Number in US dollars. Display dollar sign ($) at front of number within spinner. The dollar sign is *not* sent as part of the value.
    - You could also use “suffix” to put something like “USD” at the end, but suffix is less common than prefix
- **Output**
  - Corresponding value in Japanese yen (¥)
- **Command button**
  - No action given, so same page automatically redisplayed. Conversion shown on same page as input.

Example 3: Bean

```
@ManagedBean
public class CurrencyBean {
  private double dollars=100;

  public double getDollars() {
    return(dollars);
  }

  public void setDollars(double dollars) {
    this.dollars = dollars;
  }

 /** Dollar to Yen conversion taken from xe.com 9/2013. */

  public double getYen() {
    return(dollars * 97.13);
  }
}
```

Example 3: Facelets and Results

```
...
<h:form>
   US Dollars:
   <p:spinner prefix="$" value="#{currencyBean.dollars}"/>
   <p:commandButton value="Convert to Yen" ajax="false"/>
   $#{currencyBean.dollars} (USD) = &yen;#{currencyBean.yen} (JPY)
</h:form>
...
```


US Dollars: $150.0 $150.0 (USD) = ¥14569.5 (JPY)

Example 4: Ajax Updates

- **Input**
  - Temperature in Fahrenheit
  - UI will collect whole number in range from 32 to 212. Set initial bean value to 32 so it is in this range. No stepFactor given, so default of 1 is automatically used.
    - This is the same as example 1
- **Output**
  - Temperature in Celsius (rounded to nearest whole number). Shown automatically when spinner changes.
- **p:ajax – no command button**
  - Updates happen automatically when spinner changes.
    - p:ajax covered in more detail in Date Input section

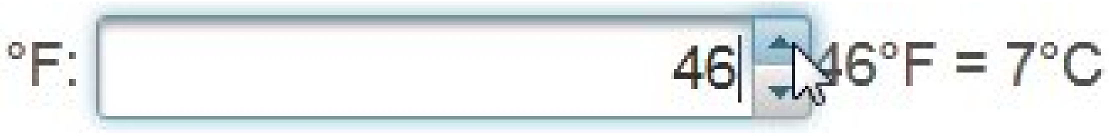
Example 4: Bean

```
public class FahrenheitBean {
  private int f=32;

  public int getF() {
    return (f);
  }

  public void setF(int f) {
    this.f = Math.max(f, -460);  // -459.67 is absolute zero
  }

  public int getC() {
    return((int)((f - 32)*(5.0/9.0)));
  }
}
```

This is the same bean as used in example 1, with no changes.

Example 4: Facelets and Results

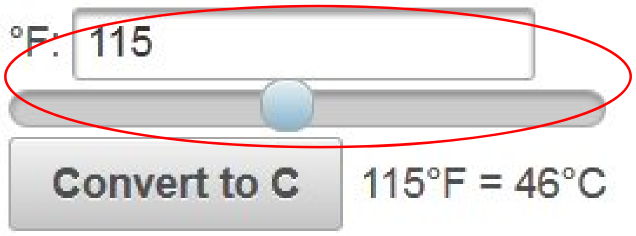
```
...
<h:form>
   &deg;F:
   <p:spinner min="32" max="212" value="#{fBean2.f}">
     <p:ajax update="f c"/>
   </p:spinner>
   <h:outputText value="#{fBean2.f}&deg;F" id="f"/>
   =
   <h:outputText value="#{fBean2.c}&deg;C" id="c"/>
</h:form>
...
```



p:slider

p:slider: Overview

- **Appearance and behavior**
  - Draggable slider with number displayed in associated textfield. User can also type number directly in the textfield.
    - UI prevents user from typing non-integers in textfield, but it does allow the user to enter values outside the range.
- **Purpose: for collecting numbers from user**
  - Value is automatically converted to number. UI prevents user from entering an illegal value. Can collect whole numbers only.



p:slider: Basics

- **<p:inputText ... id="foo"/>**
  - value
    - Must point to bean property of type int or Integer. Floating point numbers not allowed.
      - The slider does not point at a bean property at all. The slider refers to a textfield, and it is the textfield that has the value.
  - id
    - Identifies textfield to slider (slider is usually below textfield)
- **<p:slider for="foo".../>**
  - for
    - The component whose value is changed
  - minValue, maxValue
    - Note inconsistency with p:spinner, where they were called min and max. Also note that the UI does not prevent the user from manually entering value outside this range (unlike with spinner).

p:slider: Controlling Width

- **Slider takes up width of containing element**
  - This is usually too big, unless containing element is a table cell or a div with a width given in CSS
- **h:panelGrid is a simple way to make tables**
  - You can also use p:panelGrid as seen in the section on popup windows, but borders are usually visible
- **So, put slider in a table cell via h:panelGrid**

```
<h:panelGrid>
  <h:panelGroup>
    ...<p:inputText id="blah" value="…"/>
  </h:panelGroup>
  <p:slider minValue="…" maxValue="…" for="blah"/>
  <h:panelGroup>...</h:panelGroup>
</h:panelGrid>
```

p:slider: More Attributes

- **<p:slider …>**
  – display
    - Component that is visually updated. If you want displayed value to be read only, you can use "for" to point at a hidden field, then have display point at an h:outputText that shows value.
      – See later example.
  – type (vertical or horizontal)
    - Default is horizontal
  – step
    - The smallest amount that the underlying value can change. Default is 1.

Example 1: Basics

- **Input**
  – Temperature in Fahrenheit
  – UI will collect whole number in range from 32 to 212. User can either drag slider or type value into textfield.
- **Output**
  – Temperature in Celsius (rounded to nearest whole number)
- **Command button**
  – No action given, so same page automatically redisplayed. Conversion (Celsius temp) shown on same page as input.

Example 1: Bean

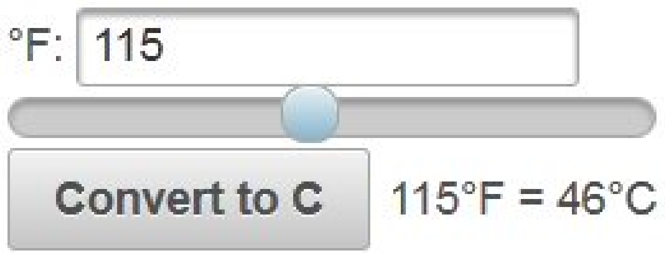
```
public class FahrenheitBean {
  private int f=32;

  public int getF() {
    return (f);
  }

  public void setF(int f) {
    this.f = Math.max(f, -460);  // -459.67 is absolute zero
  }

  public int getC() {
    return((int)((f - 32)*(5.0/9.0)));
  }
}
```

Same bean used in the spinner F to C example.

As mentioned in spinner section, instead of using @ManagedBean, the bean name is given in faces-config.xml. This lets us use several different instances of the same class in the same page, but with different bean names. If you don't know how to declare beans in faces-config.xml, see lecture on page navigation and faces-config.xml in general JSF2 tutorial at coreservlets.com.

Example 1: Facelets and Results

```
<h:form>
<h:panelGrid>
  <h:panelGroup>
    &deg;F:
    <p:inputText id="fInput1" value="#{fBean1.f}"/>
  </h:panelGroup>
  <p:slider minValue="32" maxValue="212" for="fInput1"/>
  <h:panelGroup>
    <p:commandButton value="Convert to C" ajax="false"/>
    #{fBean1.f}&deg;F = #{fBean1.c}&deg;C<p/>
  </h:panelGroup>
</h:panelGrid>
</h:form>
...
```



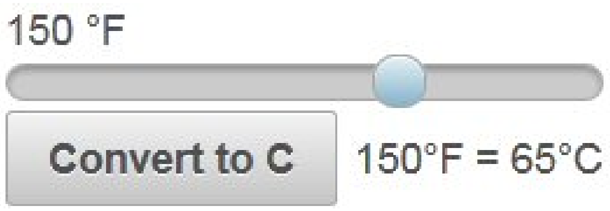
115°F = 46°C

Example 2: Read-Only Display

- **Input**
  - Temperature in Fahrenheit
  - UI will collect whole number in range from 32 to 212. User can only drag slider: there is no textfield to type into. “for” points at hidden field with value to update internally and “display” points at a JSF element (usually h:outputText) to visually update.
    - The “display” element is updated with pure client-side JavaScript; no Ajax requests are involved
- **Output**
  - Temperature in Celsius (rounded to nearest whole number).
- **Command button**
  - No action given, so same page automatically redisplayed. Conversion shown on same page as input.
- **Bean**
  - Same as in previous example, so not repeated here

Example 2: Facelets and Results

```
<h:form>
<h:panelGrid>
  <h:panelGroup>
    <h:inputHidden id="fInput2" value="#{fBean2.f}" />
    <h:outputText id="fDisplay2" value="#{fBean2.f}"/>
    &deg;F
  </h:panelGroup>
  <p:slider minValue="32" maxValue="212" for="fInput2"
            display="fDisplay2"/>
  <h:panelGroup>
    <p:commandButton value="Convert to C" ajax="false" />
    #{fBean2.f}&deg;F = #{fBean2.c}&deg;C<p/>
  </h:panelGroup>
</h:panelGrid>
</h:form>
...
```



150°F = 65°C

Example 3: Ajax Update

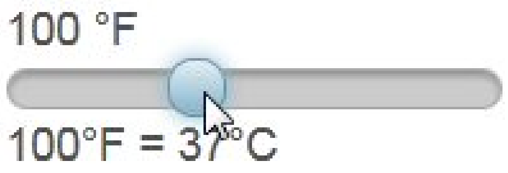
- **Input**
  – Temperature in Fahrenheit
  – UI will collect whole number in range from 32 to 212. Slider uses p:ajax with id of component to be updated.
    - Reminder: p:ajax uses “update” and “process” instead of “render” and “execute” as with f:ajax.
- **Output**
  – Temperature in Celsius (rounded to nearest int).
- **Command button**
  – None. Conversion done automatically when slider moves.
- **Bean**
  – Adds extra method to show conversion so that a single h:outputText can be used

Example 3: Bean

```
public class FahrenheitBean {
  private int f=32;

  public int getF() {
    return (f);
  }
  public void setF(int f) {
    this.f = Math.max(f, -460);  // -459.67 is absolute zero
  }
  public int getC() {
    return((int)((f - 32)*(5.0/9.0)));
  }

  public String getStatus() {
    return(String.format("%s&deg;F = %s&deg;C",
                         f, getC()));
  }
}
```

Example 3: Facelets and Results

```
<h:form>
<h:panelGrid width="200">
  <h:panelGroup>
    <h:inputHidden id="fInput3" value="#{fBean3.f}" />
    <h:outputText id="fDisplay3" value="#{fBean3.f}"/>
    &deg;F
  </h:panelGroup>
  <p:slider minValue="32" maxValue="212" for="fInput3"
            display="fDisplay3">
    <p:ajax process="fInput3" update="status"/>
  </p:slider>
  <h:outputText value="#{fBean3.status}" id="status"
                escape="false"/>
</h:panelGrid>
</h:form>
...
```

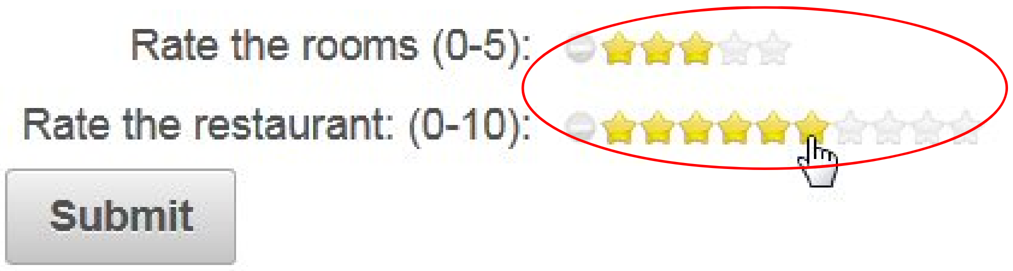


```
100 °F
100°F = 37°C
```

p:rating

p:rating: Overview

- **Appearance and behavior**
  - Horizontal row of stars that can be clicked (for choosing rating) or displayed read-only (for displaying rating)
- **Purpose: for collecting ratings from user**
  - Value is automatically converted to int from 0 to max number of stars (typically 5 or 10). Used for very specific purpose of getting an int corresponding to a rating.



p:rating: Summary of Most Important Attributes

- **<p:rating value="…" …/>**
  - value
    - Must point to bean property of type int or Integer. Floating point numbers not allowed.
  - stars (*integer* [default 5])
    - The number of stars to show. If value is less than the number of stars, stars will be in light gray. If value is greater than the number of stars, highest value will be shown and no error message given.
  - readonly (true or false [default])
    - Can user change the rating? A readonly display is used for the results page.
      - You must put p:rating inside h:form, even when using readonly.
      - Note that this attribute is “readonly”, not “readOnly”. PrimeFaces is inconsistent on use of camel case.

Bean (Used in All Examples)

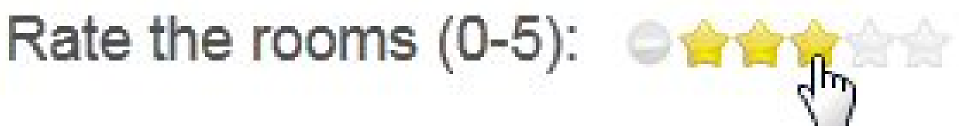
```
@ManagedBean
public class RatingBean {
  private int roomRating, restaurantRating;

  public int getRoomRating() {
    return(roomRating);
  }

  public void setRoomRating(int roomRating) {
    this.roomRating = roomRating;
  }

  public int getRestaurantRating() {
    return(restaurantRating);
  }

  public void setRestaurantRating(int restaurantRating) {
    this.restaurantRating = restaurantRating;
  }

  public String processRatings() {
    return("show-ratings");
  }
}
```

Example 1: Basic Usage

- **Idea**
  - Gather an int for a rating.
    - Range 0-5.
- **Facelets code**

```
Rate the rooms (0-5):
<p:rating value="#{ratingBean.roomRating}"/>
```

- **Result**

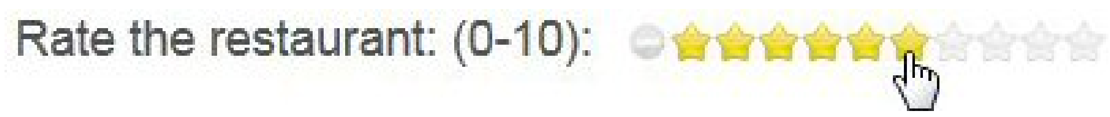

Rate the rooms (0-5):

Example 2: stars

- **Idea**
  - Gather an int for a rating.
    - Range 0-10.
- **Facelets code**

```
Rate the restaurant: (0-10):
<p:rating value="#{ratingBean.restaurantRating}"
          stars="10"/>
```

- **Result**

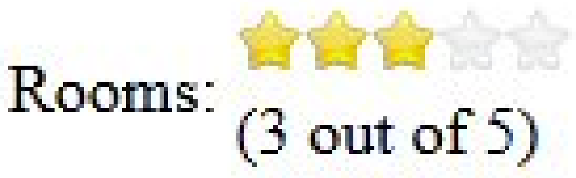

Rate the restaurant: (0-10):

Example 3: readonly

- **Idea**
  - Display an int from a rating. No user interaction.
    - Range 0-5.
- **Facelets code**

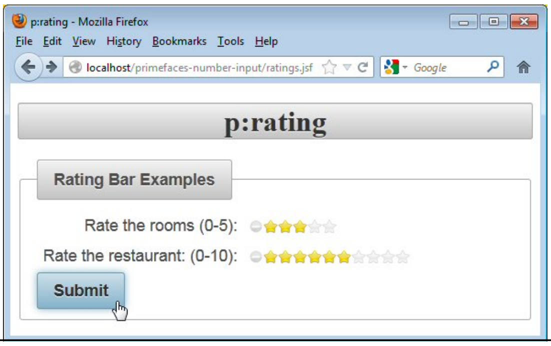
```
<p:rating value="#{ratingBean.roomRating}"
          readonly="true"/><br/>
(#{ratingBean.roomRating} out of 5)
```

- **Result**


Rooms: (3 out of 5)

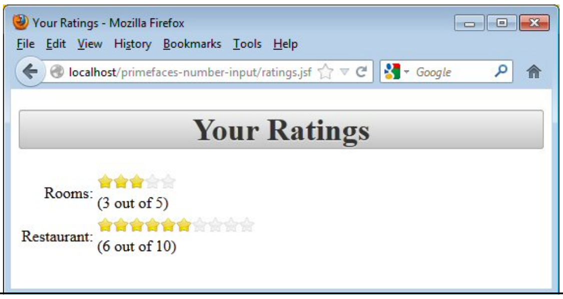
Combined Example: Input Page

```
<h:form>
<h:panelGrid columns="2" cellpadding="5" styleClass="formTable">
   Rate the rooms (0-5):
   <p:rating value="#{ratingBean.roomRating}"/>

   Rate the restaurant: (0-10):
   <p:rating value="#{ratingBean.restaurantRating stars="10"/>
</h:panelGrid>
<p:commandButton action="#{ratingBean.processRatings}"
                 value="Submit" ajax="false"/>
</h:form>
```


p:rating - Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost/primefaces-number-input/ratings.jsf
Google
p:rating
Rating Bar Examples
Rate the rooms (0-5):
Rate the restaurant: (0-10):
Submit

Combined Example: Results Page

```
<h:form>
<h:panelGrid columns="2" styleClass="formTable">
    Rooms:
    <h:panelGroup>
    <p:rating value="#{ratingBean.roomRating readonly="true"/><br/>
    (#{ratingBean.roomRating} out of 5)
    </h:panelGroup>
    Restaurant:
    <h:panelGroup>
    <p:rating value="#{ratingBean.restaurantRating}"
              stars="10" readonly="true"/><br/>
    (#{ratingBean.restaurantRating} out of 10)
    </h:panelGroup>
</h:panelGrid>
</h:form>
```


Your Ratings - Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost/primefaces-number-input/ratings.jsf
Google
Your Ratings
Rooms:
(3 out of 5)
Restaurant:
(6 out of 10)

Note that p:rating is inside h:form, even though there is no input.

Wrap-Up

Summary

- **p:spinner (widely used)**
  - <p:spinner value="#{someBean.number}"
    min="…" max="…" stepFactor="…"/>
    - Can collect int or double, depending on stepFactor
- **p:slider (widely used)**
  - <h:inputText value="#{someBean.number}" id="blah"/>
  - <p:slider for="blah" minValue="…" maxValue="…"/>
    - Can only collect int. You must control slider width.
- **p:rating (rarely used)**
  - <p:rating value="#{someBean.number}"
    stars="…" readonly="…"/>
    - Can only collect int

Questions?

More info:
http://www.coreservlets.com/JSF-Tutorial/jsf2/ – JSF 2.2 tutorial
http://www.coreservlets.com/JSF-Tutorial/primefaces/ – PrimeFaces tutorial
http://courses.coreservlets.com/jsf-training.html – Customized JSF and PrimeFaces training courses
http://coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training