



Utilize Tools to Simplify
Creation of Fast and
Reliable Parallel Code.

Home » Enterprise Java » jsf » JSF Standard Converters Example

ABOUT VEERAMANI KALYANASUNDARAM



Veera is a Software Architect working in telecom domain with rich experience in Java Middleware Technologies. He is a OOAD practitioner and interested in Performance Engineering.



JSF Standard Converters Example

Posted by: Veeramani Kalyanasundaram in jsf March 16th, 2015

100% FREE

Veeam

Microsoft SQL Server

FREE tool for SQL Server® database recovery

[DOWNLOAD NOW](#)

Installed by 500.000 VMware | Hyper-V admins

In this example of JSF Standard Converters, we will show you how Java Server Faces standard converters work and also discuss the various options available to use standard converters.

When a request is sent from the browser to the application server, the form values are sent as String. To convert the String into Java Objects, we need to use converters. Similarly when the Java Object is passed back from the application server and gets converted into HTML, we need to convert it to String. JSF provides a set of standard converters as well as give option to create custom converters. Let's begin with setting up a JSF project and do all the necessary configuration to run the application.

Want to be a JSF Ninja?

Subscribe to our newsletter and download the JSF 2.0 Programming Cookbook right now!

In order to get you prepared for your JSF development needs, we have compiled numerous recipes to help you kick-start your projects. Besides reading them online you may download the eBook in PDF format!

Email address:

Your email address

Sign up

Our preferred environment is Eclipse. We are using Eclipse Luna SR1 with Maven Integration Plugin, JDK 8u25 (1.8.0_25) and Tomcat 8 application server. Having said that, we have tested the code against JDK 1.7 and Tomcat 7 as well.

Tip

You may skip project creation and jump directly to the **beginning of the example** below.

1. Create a new Maven Project

Go to File -> New->Other-> Maven Project

JIRA:
My team runs on Jira. Try it free for 30 days and see how Jira can help your team be more efficient, effective, and successful.

NEWSLETTER

170,124 insiders are already reading our weekly updates and complimentary whitepapers!

Join them now to gain access to the latest news in Java, JSF, and Angular.js as well as insights about Android, Groovy and other related technologies.

Email address:

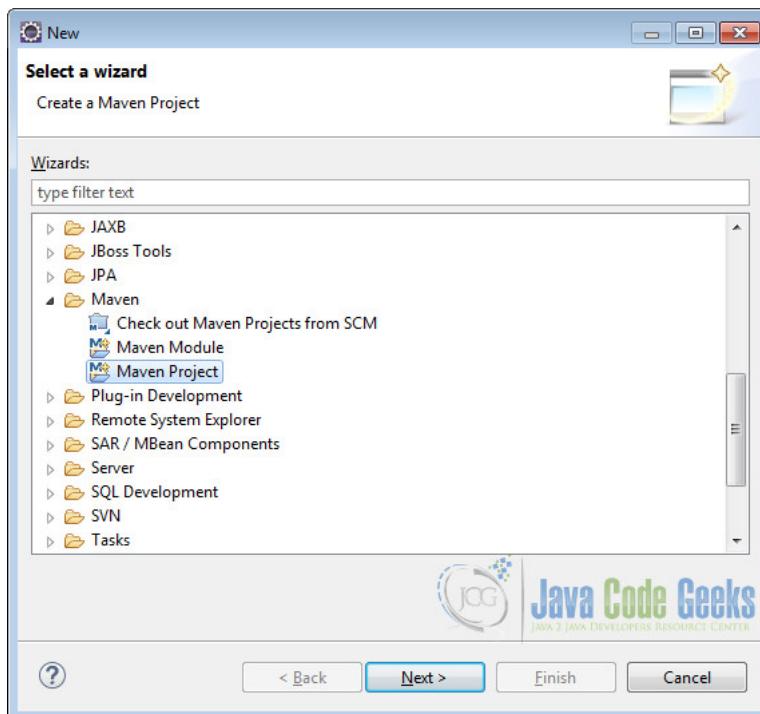
Your email address

Sign up

JOIN US

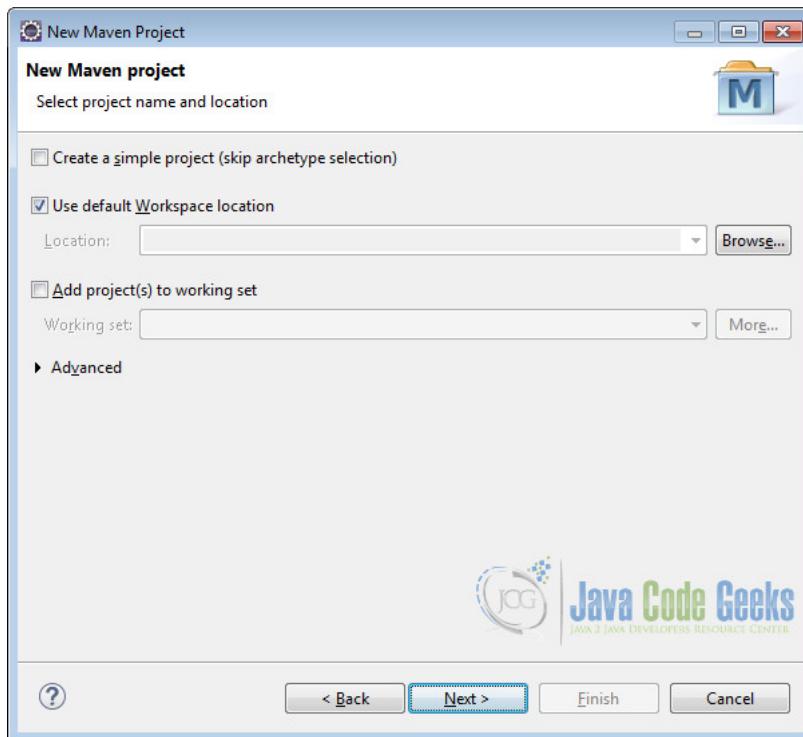


With **1,500** unique visitors per month, Java Code Geeks is one of the most popular Java developer websites. Our readers are mostly Java developers, architects, and managers who are looking for the latest news, tips, and tricks in Java, JSF, and Angular.js. We also cover topics like mobile development, cloud computing, big data, and more. Our goal is to provide valuable resources and tools to help you become a better developer. So if you're looking for unique and interesting content that you can't find elsewhere, check out our **JCG** partners program.



Maven Setup – Step 1

In the "Select project name and location" page of the wizard, make sure that "Create a simple project (skip archetype selection)" option is **unchecked**, hit "Next" to continue with default values.



Maven setup – step 2

Here choose "maven-archetype-webapp" and click on Next.

be a **guest writer** for Java Cod
your writing skills!

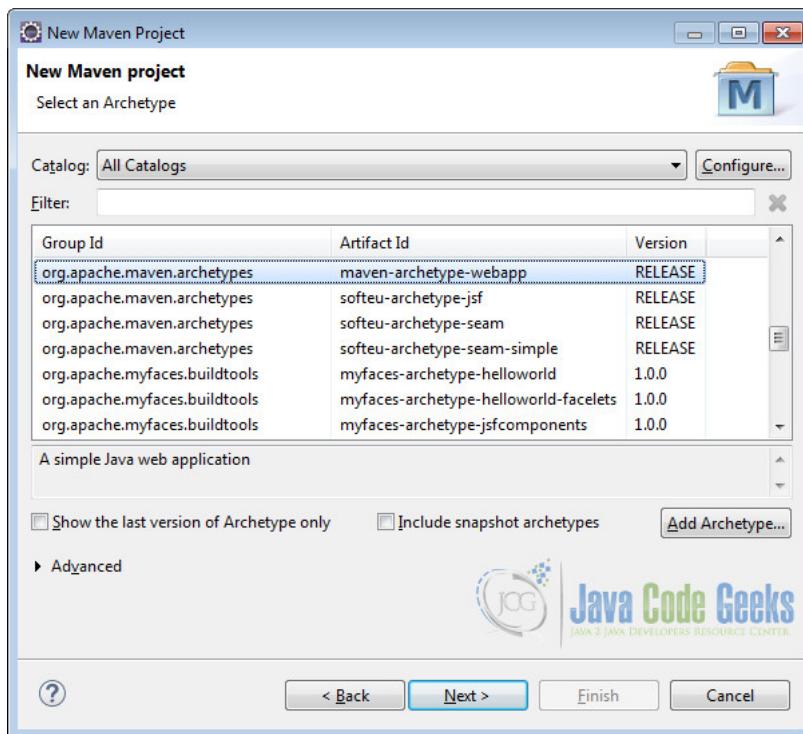
CAREER OPPORTUNITIES

- Sr Devops Engineer **Scout ET**
Eagan, MN
Oct, 14
- Sr. Java Developer (Contract) **Ali**
San Jose, CA
Oct, 13
- Java Developer **Carolina Hurricane**
Raleigh, NC
Oct, 06
- Senior Application Developer (re **Systems**
Reston, VA
Oct, 13
- Sr. Java Developer - Struts/Hibernate **Trenton, NJ**
Trenton, NJ
Oct, 10
- Sr Information Security Engineer **Airlines**
Fort Worth, TX
Oct, 14
- JAVA Developer **Advanced Technical Inc**
Greenwood Village, CO
Oct, 06
- JAVA (server side) Developer **Shutterstock Services Group**
Boston, MA
Sep, 14
- Java Developer **Transamerica**
Plano, TX
Oct, 05
- Senior Software Engineer - Angular **Simeio Solutions**
Atlanta, GA
Oct, 12

1 2 ... 5896 >

Keyword ...
Location ...
Country ...

Filter Results
jobs by **Indeed**



Maven setup – step 3

In the "Enter an artifact id" page of the wizard, you can define the name and main package of your project. Set the "Group Id" variable to
"com.javacodegeeks.snippets.enterprise"

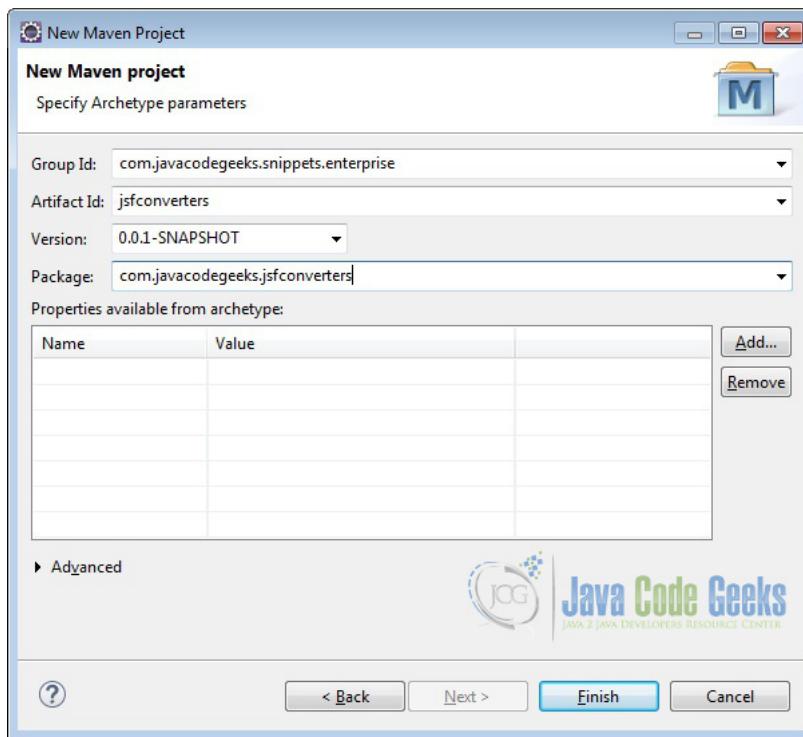
and the "Artifact Id" variable to

"jsfconverters"

. For package enter

"com.javacodegeeks.jsfconverters"

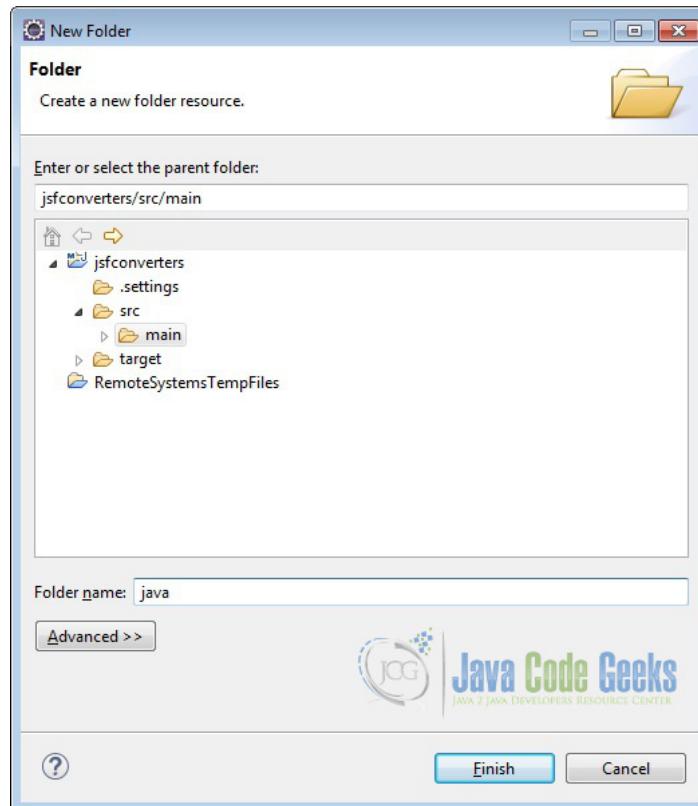
and hit "Finish" to exit the wizard and to create your project.



Maven setup – step 4

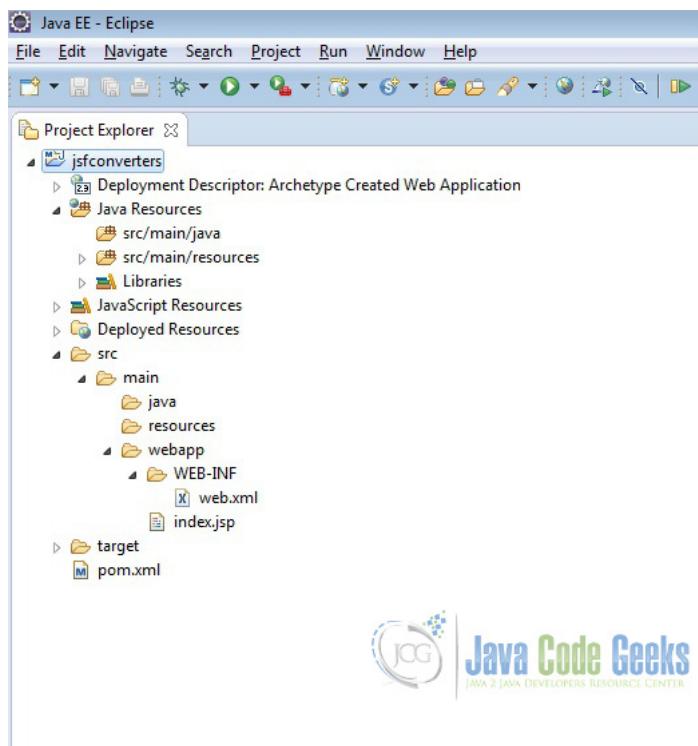
Now create a folder called java under

src/main



Maven setup – step 5

Refresh the project. Finally, the project structure will look like the below.

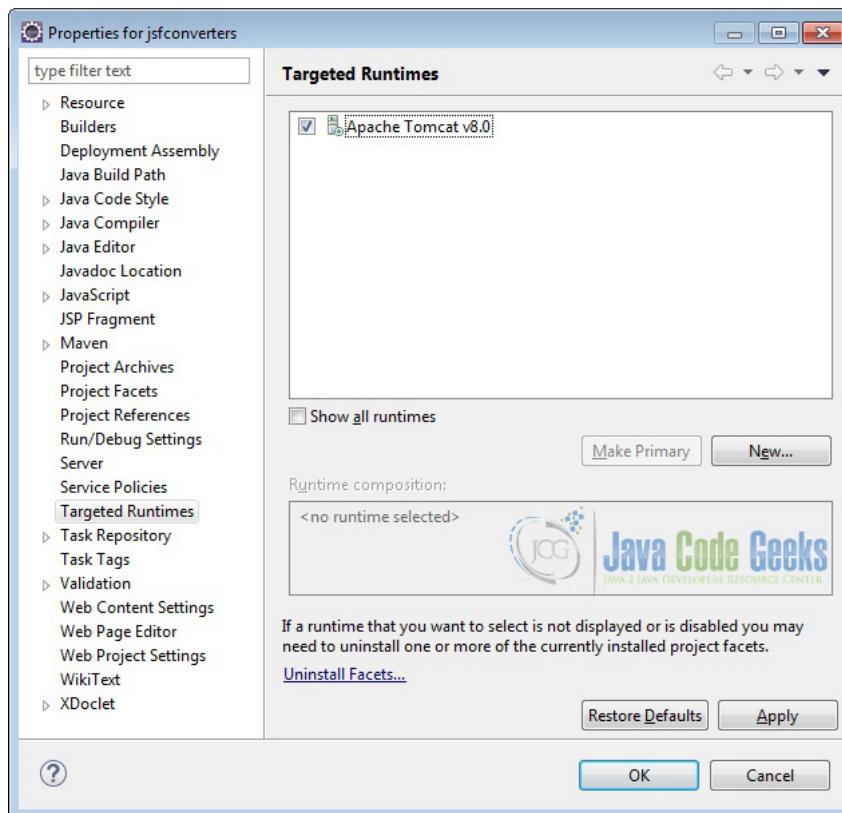


Maven setup – step 6

If you see any errors in the

`index.jsp`

, set target runtime for the project.



Maven setup – step 7

2. Modify POM to include JSF dependency

Add the dependencies in Maven's

pom.xml

file, by editing it at the "Pom.xml" page of the POM editor.

[pom.xml](#)

```

01 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
02   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
03   <modelVersion>4.0.0</modelVersion>
04   <groupId>com.javacodegeeks.snippets.enterprise</groupId>
05   <artifactId>jsfconverters</artifactId>
06   <packaging>war</packaging>
07   <version>0.0.1-SNAPSHOT</version>
08   <name>jsfconverters Maven Webapp</name>
09   <url>http://maven.apache.org</url>
10   <dependencies>
11     <dependency>
12       <groupId>junit</groupId>
13       <artifactId>junit</artifactId>
14       <version>3.8.1</version>
15       <scope>test</scope>
16     </dependency>
17     <dependency>
18       <groupId>com.sun.faces</groupId>
19       <artifactId>jsf-api</artifactId>
20       <version>2.2.9</version>
21     </dependency>
22     <dependency>
23       <groupId>com.sun.faces</groupId>
24       <artifactId>jsf-impl</artifactId>
25       <version>2.2.9</version>
26     </dependency>
27   </dependencies>
28   <build>
29     <finalName>jsfconverters</finalName>
30   </build>
31 </project>
```

3. Add Faces Servlet in web.xml

The

web.xml

file has to be modified to include the faces servlet configuration as below.

[web.xml](#)

```

01 <!DOCTYPE web-app PUBLIC
02   "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
03   "http://java.sun.com/dtd/web-app_2_3.dtd" >
```

```

05 <web-app>
06   <display-name>Archetype Created Web Application</display-name>
07   <servlet>
08     <servlet-name>Faces Servlet</servlet-name>
09     <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
10     <load-on-startup>1</load-on-startup>
11   </servlet>
12   <servlet-mapping>
13     <servlet-name>Faces Servlet</servlet-name>
14     <url-pattern>*.xhtml</url-pattern>
15   </servlet-mapping>
16 </web-app>

```

4. Standard Converters

JSF provides a set of standard converters. The following section shows all the available converters and state the purpose for each of the converter.

- **BigDecimalConverter** – For Conversion between String and

```
java.math.BigDecimal
```

- **BigIntegerConverter** – For Conversion between String and

```
java.math.BigInteger
```

- **BooleanConverter** – For Conversion between String and

```
java.lang.Boolean
```

- **ByteConverter** – For Conversion between String and

```
java.lang.Byte
```

- **CharacterConverter** – For Conversion between String and

```
java.lang.Character
```

- **DateTimeConverter** – For Conversion between String and

```
java.util.Date
```

- **DoubleConverter** – For Conversion between String and

```
java.lang.Double
```

- **FloatConverter** – For Conversion between String and

```
java.lang.Float
```

- **IntegerConverter** – For Conversion between String and

```
java.lang.Integer
```

- **LongConverter** – For Conversion between String and

```
java.lang.Long
```

- **NumberConverter** – For Conversion between String and

```
java.lang.Number
```

- **ShortConverter** – For Conversion between String and

```
java.lang.Short
```

The **DateTimeConverter** and **NumberConverter** have their own tags and provide attributes for data conversion. We will discuss about the two converters in detail later.

5. How to use Converters

JSF provides three ways to use converters. We can use any of the method depending on the type of converter.

5.1 Using converter attribute

We can add the

```
converter
```

attribute to the UI component using the fully qualified class name.

```
1 <h:inputText id="age" converter="javax.faces.Integer" />
```

5.2 Using f:converter tag

We can include the

```
f:converter
```

tag within a component. The

```
converterID
```

attribute point to reference the converter's name.

```
1 <h:inputText id="age">
2   <f:converter converterID="javax.faces.Integer" />
3 </h:inputText>
```

5.3 Using converter tags

We can use the standard converter tags provided in the JSF.

```
1 <h:outputText value="#{userBean.height}">
2   <f:convertNumber maxFractionDigits="2" />
3 </h:outputText>
```

Or by using custom converter

```
1 <h:outputText value="#{userBean.ssn}">
2   <my:ssnConverter country="US" />
3 </h:outputText>
```

6. Implicit Converter

JSF provides standard converters that automatically perform conversion for basic Java types. We will see it working by creating an example.

First we will create a package called

```
com.javacodegeeks.jsfconverters
```

under the folder

```
src/main/java
```

. Now, we need to create a managed bean called

```
UserBean
```

. The

```
@ManagedBean
```

annotation makes the POJO as managed bean. The

```
@SessionScoped
```

annotation on the bean makes the bean available to the entire user session.

UserBean.java

```
01 package com.javacodegeeks.jsfconverters;
02
03 import java.util.Date;
04 import javax.faces.bean.ManagedBean;
05 import javax.faces.bean.SessionScoped;
06
07 @ManagedBean(name="userBean", eager=true)
08 @SessionScoped
09 public class UserBean {
10
11     private String firstName;
12     private String lastName;
13     private int age;
14     private Date dateOfBirth;
15     private Double height;
16
17     public String getFirstName() {
18         return firstName;
19     }
20
21     public void setFirstName(String firstName) {
22         this.firstName = firstName;
23     }
24
25     public String getLastName() {
26         return lastName;
27     }
28
29     public void setLastName(String lastName) {
30         this.lastName = lastName;
31     }
32
33     public int getAge() {
34         return age;
35     }
36
37     public void setAge(int age) {
38         this.age = age;
39     }
40
41     public Date getDateOfBirth() {
42         return dateOfBirth;
43     }
44
45     public void setDateOfBirth(Date dateOfBirth) {
46         this.dateOfBirth = dateOfBirth;
47     }
48 }
```

```

49 public Double getHeight() {
50     return height;
51 }
52
53     public void setHeight(Double height) {
54         this.height = height;
55     }
56
57 }
58

```

Now, we will create a view called

adduser.xhtml

under

/src/main/webapp

. We have used

`h:outputLabel`

to display the label and

`h:inputText`

to get the user input. We will submit the form using the component

`h:commandButton`

. The action tag of

`h:commandButton`

is set to "viewuser". We will use the implicit navigation feature of JSF for navigating to the "viewuser" page.

adduser.xhtml

```

01 <?xml version="1.0" encoding="ISO-8859-1" ?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml"
04     xmlns:h="http://java.sun.com/jsf/html"
05     xmlns:f="http://java.sun.com/jsf/core">
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
08 <title>Add User</title>
09 </head>
10 <body>
11     <h:form>
12         <h2>Add User</h2>
13         <div>
14             <h:outputLabel for="firstName">First Name</h:outputLabel>
15             </div>
16             <div>
17                 <h:inputText id="firstName" label="First Name"
18                     value="#{userBean.firstName}"></h:inputText>
19             </div>
20             <div>
21                 <h:outputLabel for="lastName">Last Name</h:outputLabel>
22             </div>
23             <div>
24                 <h:inputText id="lastName" label="Last Name"
25                     value="#{userBean.lastName}"></h:inputText>
26             </div>
27             <div>
28                 <h:outputLabel for="age">Age</h:outputLabel>
29             </div>
30             <div>
31                 <h:inputText id="age" label="age" value="#{userBean.age}">
32                     </h:inputText>
33             </div>
34             <div>&nbsp;</div>
35             <div>&nbsp;</div>
36             <div>
37                 <h:commandButton value="Submit" action="viewuser"></h:commandButton>
38             </div>
39     </h:form>
40 </body>
41 </html>

```

We will create another view called

viewuser.xhtml

under

/src/main/webapp

to display the values entered by the user.

viewuser.xhtml

```

01 <?xml version="1.0" encoding="ISO-8859-1" ?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml"
04     xmlns:h="http://java.sun.com/jsf/html"
05     xmlns:f="http://java.sun.com/jsf/core">
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />

```

```

08 <title>View User</title>
09 </head>
10 <body>
11   <h:form>
12     <h2>View User</h2>
13     <h4>
14       <h:outputText value="#{userBean.firstName}"></h:outputText>
15       <br />
16       <h:outputText value="#{userBean.lastName}"></h:outputText>
17       <br />
18       <h:outputText value="#{userBean.age}"></h:outputText>
19     </h4>
20   </h:form>
21 </body>
22 </html>

```

Now we can create the deployment package using Run as → Maven clean and then Run as → Maven install. This will create a war file in the target folder. The

war

file produced must be placed in

webapps

folder of tomcat. Now we can start the server.

Open the following URL in the browser.

<http://localhost:8080/jsfconverters/adduser.xhtml>



Add User -1

Enter the values for First Name, Last Name and age. Now, click on submit button. JSF will use the implicit navigation and display the viewuser.xhtml

. Here the value for age is converted to int by the standard converter automatically.



View User – 1

To validate the implicit converter, try to enter **some characters** in the age field and click on the submit. You will see error in **Tomcat server window**.

7. DateTimeConverter

The JSF DateTimeConverter provides the following attributes to convert and format the Date.

- **dateStyle** – Predefined formatting style which determines how the date component of a date string is to be formatted and parsed.
- **locale** – Locale to be used during formatting.
- **pattern** – Custom formatting pattern can be used using this attribute.
- **timeStyle** – Predefined formatting style which determines how the time component of a date string is to be formatted and parsed.
- **timeZone** – Time zone in which to interpret any time information in the date String.
- **type** – Specifies date or time or both to be used during formatting.

Now, we modify the

adduser.xhtml

to accept the Date of Birth as user input.

adduser.xhtml

```

01 <?xml version="1.0" encoding="ISO-8859-1" ?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml"
04   xmlns:h="http://java.sun.com/jsf/html"
05   xmlns:f="http://java.sun.com/jsf/core">
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
08 <title>Add User</title>
09 </head>
10 <body>
11   <h:form>
12     <h2>Add User</h2>
13     <div>
14       <h:outputLabel for="firstName">First Name</h:outputLabel>
15     </div>
16     <div>
17       <h:inputText id="firstName" label="First Name"
18         value="#{userBean.firstName}"></h:inputText>
19     </div>
20     <div>
21       <h:outputLabel for="lastName">Last Name</h:outputLabel>
22     </div>
23     <div>
24       <h:inputText id="lastName" label="Last Name"
25         value="#{userBean.lastName}"></h:inputText>
26     </div>
27     <div>
28       <h:outputLabel for="age">Age</h:outputLabel>
29     </div>
30     <div>
31       <h:inputText id="age" label="age" value="#{userBean.age}">
32     </h:inputText>
33   </div>
34   <div>
35     <h:outputLabel for="dateOfBirth">Date of Birth (dd-mm-yyyy)</h:outputLabel>
36   </div>
37   <div>
38     <h:inputText id="dateOfBirth" label="Date of Birth" value="#{userBean.dateOfBirth}">
39     <f:convertDateTime pattern="dd-mm-yyyy" />
40   </h:inputText>
41 </div>
42 <div>&nbsp;</div>
43 <div>&nbsp;</div>
44 <div>
45   <h:commandButton value="Submit" action="viewuser"></h:commandButton>
46 </div>
47 </h:form>
48 </body>
49 </html>
```

Now modify the

viewuser.xhtml

to display the date of birth in a different format using the

f:convertDateTime

tag and its

pattern

attribute.

viewuser.xhtml

```

01 <?xml version="1.0" encoding="ISO-8859-1" ?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml"
04   xmlns:h="http://java.sun.com/jsf/html"
05   xmlns:f="http://java.sun.com/jsf/core">
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
```

```

08 <title>View User</title>
09 </head>
10 <body>
11   <h:form>
12     <h2>View User</h2>
13     <h4>
14       <h:outputText value="#{userBean.firstName}"></h:outputText>
15       <br />
16       <h:outputText value="#{userBean.lastName}"></h:outputText>
17       <br />
18       <h:outputText value="#{userBean.age}"></h:outputText>
19       <br />
20       <h:outputText value="#{userBean.dateOfBirth}">
21         <f:convertDateTime pattern="dd-MMM-yy"></f:convertDateTime>
22       </h:outputText>
23     </h4>
24   </h:form>
25 </body>
26 </html>

```

Now again package using maven and copy the

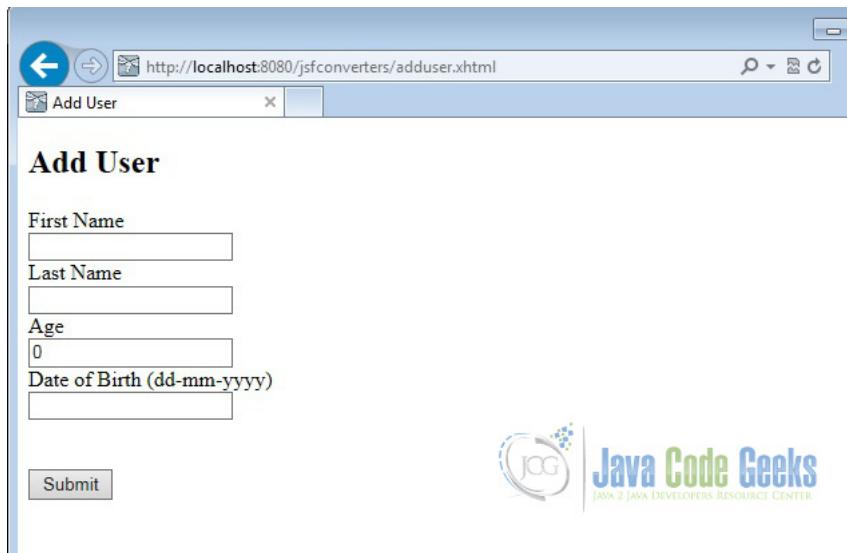
war

to the apache tomcat

webapps

folder. Open the following URL in the browser

<http://localhost:8080/jsfconverters/adduser.xhtml>



Add User -2

Enter the values for first name, last name, age and date of birth. Now, click on the submit button. The JSF will use the implicit navigation to forward the request to

viewuser.xhtml

. we will see the date of birth being displayed in the new format

dd-MMM-yy

defined using the pattern attribute.



View User -2

8. NumberConverter

The JSF NumberConverter provides the following attributes to convert and format the number.

- **currencyCode** – To apply the currency code.
- **currencySymbol** – To apply the currency symbol.
- **groupingUsed** – Flag specifying whether formatted output will contain grouping separators.
- **integerOnly** – Flag specifying whether only the integer part of the value will be formatted and parsed.
- **locale** – Locale whose predefined styles for numbers are used during formatting and parsing.
- **maxFractionDigits** – Maximum number of digits in the fractional portion.
- **maxIntegerDigits** – Maximum number of digits in the integer portion.
- **minFractionDigits** – Minimum number of digits in the fractional portion.
- **minIntegerDigits** – Minimum number of digits in the integer portion.
- **pattern** – To define the custom formatting pattern.
- **type** – Specifies one of number, currency and percent.

Now, we modify the adduser.xhtml to accept the height as user input.

adduser.xhtml

```

01 <?xml version="1.0" encoding="ISO-8859-1" ?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml"
04   xmlns:h="http://java.sun.com/jsf/html"
05   xmlns:f="http://java.sun.com/jsf/core">
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
08 <title>Add User</title>
09 </head>
10 <body>
11   <h:form>
12     <h2>Add User</h2>
13     <div>
14       <h:outputLabel for="firstName">First Name</h:outputLabel>
15     </div>
16     <div>
17       <h:inputText id="firstName" label="First Name"
18         value="#{userBean.firstName}"></h:inputText>
19     </div>
20     <div>
21       <h:outputLabel for="lastName">Last Name</h:outputLabel>
22     </div>
23     <div>
24       <h:inputText id="lastName" label="Last Name"
25         value="#{userBean.lastName}"></h:inputText>
26     </div>
27     <div>
28       <h:outputLabel for="age">Age</h:outputLabel>
29     </div>
30     <div>
31       <h:inputText id="age" label="age" value="#{userBean.age}">
32         </h:inputText>
33     </div>
34     <div>
35       <h:outputLabel for="dateOfBirth">Date of Birth (dd-mm-yyyy)</h:outputLabel>
36     </div>
37     <div>
38       <h:inputText id="dateOfBirth" label="Date of Birth"
39         value="#{userBean.dateOfBirth}">
40           <f:convertDateTime pattern="dd-mm-yyyy" />
41         </h:inputText>
42     </div>
43     <div>
44       <h:outputLabel for="height">Height</h:outputLabel>
45     </div>
46   </div>
```

```

47   <h:inputText id="height" label="Height" value="#{userBean.height}"></h:inputText>
48 </div>
49 <div>&nbsp;</div>
50 <div>&nbsp;</div>
51 <div>
52   <h:commandButton value="Submit" action="viewuser"></h:commandButton>
53 </div>
54 </h:form>
55 </body>
56 </html>
```

Now modify the

viewuser.xhtml

GET THE JAVA SKILLS YOU NEED IN 2016.

Start Learn

attribute of the

f:convertNumber

tag to achieve this.

```

01 <?xml version="1.0" encoding="ISO-8859-1" ?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml"
04   xmlns:h="http://java.sun.com/jsf/html"
05   xmlns:f="http://java.sun.com/jsf/core">
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
08 <title>View User</title>
09 </head>
10 <body>
11   <h:form>
12     <h2>View User</h2>
13     <h4>
14       <h:outputText value="#{userBean.firstName}"></h:outputText>
15       <br />
16       <h:outputText value="#{userBean.lastName}"></h:outputText>
17       <br />
18       <h:outputText value="#{userBean.age}"></h:outputText>
19       <br />
20       <h:outputText value="#{userBean.dateOfBirth}">
21         <f:convertDateTime pattern="dd-MMM-yy"></f:convertDateTime>
22       </h:outputText>
23       <br />
24       <h:outputText value="#{userBean.height}">
25         <f:convertNumber maxFractionDigits="1" />
26       </h:outputText>
27     </h4>
28   </h:form>
29 </body>
30 </html>
```

Now again package using maven and copy the

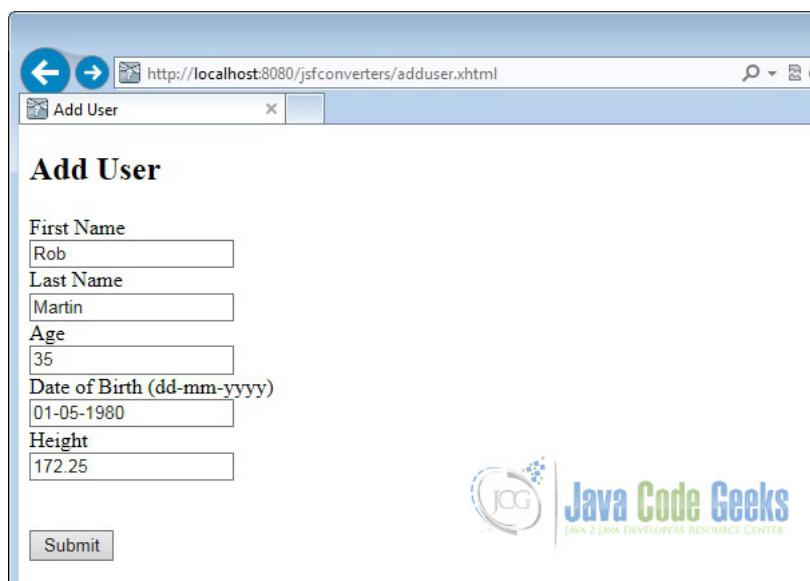
war

to the apache tomcat

webapps

folder. Open the following URL in the browser

<http://localhost:8080/jstutorial/jsfconverters/adduser.xhtml>



Add User -3

Enter the values for first name, last name, age, date of birth and height (two decimal digits). Now, click on the submit button. The JSF will use the implicit navigation to forward the request to

`viewuser.xhtml`

. We will see the height being displayed with single fraction digit irrespective of the user input.

Kob
Martin
35
01-Jan-80
172.2

View User -3

9. Download the Eclipse Project

This was an example of how to use Java Server Faces Standard Converters.

Download

You can download the full source code of this example here: [JSF Standard Converters](#)

Tagged with:

[JSF CONVERTERS](#)

Do you want to know how to develop your skillset to become a **Java Rockstar?**

Subscribe to our newsletter to start Rocking [right now!](#)
To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more

Email address:

Your email address

Sign up



[Courses](#)[News](#)[Resources](#)[Tutorials](#)[Whitepapers](#)**THE CODE GEEKS NETWORK****GET THE JAVA SKILLS YOU NEED IN 2016.**[Java Code Geeks](#)[System Code Geeks](#)[Web Code Geeks](#)[Android Alert Dialog Example](#)[Android OnClickListener Example](#)[How to convert Character to String and a String to Character Array in Java](#)[Java Inheritance example](#)[Java write to File Example](#)[java.io.FileNotFoundException – How to](#)[How to handle Array Index Out Of Bounds Exception](#)[java.lang.NoClassDefFoundError – How to solve No Class Def Found Error](#)[JSON Example With Jersey + Jackson](#)[Spring JdbcTemplate Example](#)

JCGs (Java Code Geeks) is an independent online community focused on ultimate Java to Java developers resource center; targeted at the technical team lead (senior developer), project manager and junior dev JCGs serve the Java, SOA, Agile and Telecom communities with daily news, domain experts, articles, tutorials, reviews, announcements, code snippets and source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples

[Start Learning](#)