



DOC XLS PPT PDF PNG XML
HTML EML RTF VSD BMP &
Barcode Images



Download a FREE Trial



[ANDROID ▾](#)
[CORE JAVA ▾](#)
[DESKTOP JAVA ▾](#)
[ENTERPRISE JAVA ▾](#)
[JAVA BASICS ▾](#)
[JVM LANGUAGES ▾](#)
[SOFTWARE DEVELOPMENT ▾](#)
[DEVOPS ▾](#)

Home » Enterprise Java » jsf » Param and Attribute Example with JSF 2.0

ABOUT THODORIS BAIS



Thodoris is an Oracle Certified Associate Java Programmer and currently works as a Junior Software Developer, for Intrasoft International S.A. He holds a diploma at Informatics & Telecommunications Engineering and is interested in continuous development.



Param and Attribute Example with JSF 2.0

Posted by: Thodoris Bais in jsf July 1st, 2014



Today we're gonna talk about parameter manipulation in JSF, using

param

and

attribute

tags.

1. Param Tag

What about parameters in JSF? In JSF, we can use the

<f:param>

tag, in order to pass a parameter to a component (or pass request parameters), but things here, are not so clear, as it's behavior depends on the component that it is attached.

The current example's goal is, to pass two parameters from a JSF page, to another. Let's see some introductory examples, before diving into the full example.

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

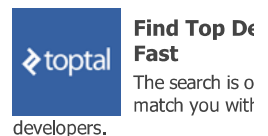
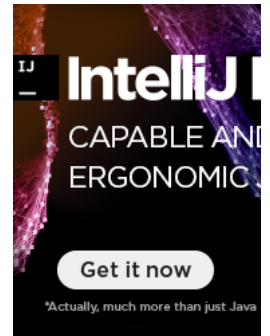
Sign up

Passing parameters to UI components

```
1 <h:outputFormat value="Hello {0}. It seems like you support {1} in this World Cup.">
2   <f:param value="Thodoris" />
3   <f:param value="Greece" />
4 </h:outputFormat>
```

Passing request parameters (attached to commandButton)

```
1 <h:commandButton id="submit"
2   value="Submit" action="#{user.outcome}">
```



NEWSLETTER

169,854 insiders are already weekly updates and complimentary whitepapers!

Join them now to gain

access to the latest news in as well as insights about Android Groovy and other related technologies.

Email address:

Your email address

Sign up

JOIN US



With **1,500** unique articles placed at your disposal, Constant lookout encouraged. So If you want unique and interesting content then check out our **JCG** partners projects.

```
3 <f:param name="country" value="Greece" />
4 </h:commandButton>
```

But how could we get these values in the back-end? How could we conform our usual bean to read them from the JSF page?

```
1 Map<String,String> params = FacesContext.getExternalContext().getRequestParameterMap();
2 String country = params.get("country");
```

The Full Example

Suppose that our super-clever web application has a page that prompts the user to insert his name (see Param Tag – Prompt Page); the application then, transfers his name to a welcome page, where a guess is made, also. In our case, it’s about the user’s favourite team in the World Cup 2014. Both parameters will be manipulated through the

```
param
```

tag.



So, regarding the technical part of our example:

- The fore-mentioned parameter manipulation will take place in the java bean
- The "index" page will contain one submit button, which will also send a parameter to the "welcome" page
- The "welcome" page will welcome the user in personal level, using the name that he provided in the "index" page; it will also display the parameter transferred from the button's click.

Here is the full example, demonstrating the usage for both of the two fore-mentioned mini-examples.

We 'll first take a look at our usual

```
UserBean
```

class:

```
01 package com.javacodegeeks.jsf.param;
02
03 import java.io.Serializable;
04 import java.util.Map;
05
06 import javax.faces.bean.ManagedBean;
07 import javax.faces.bean.SessionScoped;
08 import javax.faces.context.FacesContext;
09
10 @ManagedBean(name="user")
11 @SessionScoped
12 public class UserBean implements Serializable {
13
14     /**
15      *
16      */
17     private static final long serialVersionUID = 1L;
18
19     private String name;
20     private String country;
21
22     private String getCountryFromJSF(FacesContext context) {
23         Map<String, String> parameters = context.getExternalContext().getRequestParameterMap();
24
25         return parameters.get("country");
26     }
27
28     public String outcome() {
29         FacesContext context = FacesContext.getCurrentInstance();
30         this.country = getCountryFromJSF(context);
31
32         return "result";
33     }
34
35     public String getName() {
36         return name;
37     }
38
39     public void setName(String name) {
40         this.name = name;
41     }
42
43     public String getCountry() {
44         return country;
```

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



CAREER OPPORTUNITIES

- Jr. Software Developer**Techport**
Richmond, VA
Oct, 13
- Jr. Software Developer**Techport**
Duluth, GA
Oct, 05
- Java Developer**Transamerica**
Plano, TX
Oct, 05
- Sr. Java Developer - Struts/Hibe**i**
Trenton, NJ
Oct, 10
- Java Software Engineer - Entry L
REQUIRED**LexisNexis**
King of Prussia, PA
Oct, 08
- Java Software Engineer - Entry L
REQUIRED**Reed Elsevier**
King of Prussia, PA
Oct, 08
- Web Developer - Entry Level**Ver**
Alpharetta, GA
Sep, 26
- Java Developer**Blue Cross Blue**
NM, OK & TX
Chicago, IL
Oct, 04
- Web Developer - Entry Level**Ver**
Temple Terrace, FL
Sep, 26
- JAVA Developer**Advanced Tect**
Inc
Greenwood Village, CO
Oct, 06

1 2 ... 5909 »

Keyword ...

Location ...

Country ...

Filter Results

jobs by **indeed**

```

45     }
46
47     public void setCountry(String country) {
48         this.country = country;
49     }
50 }

```

Now, let's see the code structure of our JSF pages that interact with the "backing-bean" class:

index.xhtml

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
03 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
04 <html xmlns="http://www.w3.org/1999/xhtml"
05       xmlns:h="http://java.sun.com/jsf/html"
06       xmlns:f="http://java.sun.com/jsf/core"
07       xmlns:c="http://java.sun.com/jsp/jstl/core">
08
09 <h:head>
10 <title>JSF Param Example</title>
11 </h:head>
12 <h:body>
13 <h1>JSF 2.2 Param Example</h1>
14 <h:form id="simpleform">
15     Please insert your name:<h:inputText size="10"
16     value="#{user.name}" />
17     <br />
18     <h:commandButton id="submit" value="Submit" action="#{user.outcome}">
19         <f:param name="country" value="Greece" />
20     </h:commandButton>
21 </h:form>
22 </h:body>
23 </html>

```

result.xhtml

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
03 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
04 <html xmlns="http://www.w3.org/1999/xhtml"
05       xmlns:h="http://java.sun.com/jsf/html"
06       xmlns:f="http://java.sun.com/jsf/core"
07       xmlns:c="http://java.sun.com/jsp/jstl/core">
08
09 <h:head>
10 <title>JSF Param Example</title>
11 </h:head>
12 <h:body>
13 <h1>JSF 2.2 Param Example - Result</h1>
14 <h:outputFormat
15     value="Hello {0}.
16     It seems like you support {1} in this
17     World Cup.">
18     <f:param value="#{user.name}" />
19     <f:param value="#{user.country}" />
20 </h:outputFormat>
21 </h:body>
22 </html>

```

2. Attribute Tag

The purpose and usage of this tag are somehow similar to the

param

tag, but here, we have to deal with an actionListener. According to the tag's definition, it provides an option to pass an attribute's value to a component, or a parameter to a component via action listener.

What a better example to be experimented with the newly introduced tag, rather than a simple button? Yes, that is, let's keep it simple by investigating the tag's behavior through a button (a simple button is enough to fire the

actionListener

).

So, assuming that we have a web page with a button, we want this button to "carry" some parameters, when it is clicked (especially, a password, in our case).

In order to get familiar with the

attribute

, keep in mind that we can also use it to assign a value to our button:

```

1 <h:commandButton>
2     <f:attribute name="value" value="Submit" />
3 </h:commandButton>
4
5 // Similar to:
6 <h:commandButton value="Submit" />

```

Now, let's see how we can assign and retrieve a parameter from a component:
(The concept is exactly the same with the above example.

```

1 <h:commandButton actionListener="#{user.actionListenerSampleMethod}" >

```

GET THE JAVA SKILLS YOU NEED IN 2016.

Start Learning

...and that's our battery, carrying a password value, is accessed above (and it's also made clear by yourself, except from our request page (where we just want to display the password's value, so there isn't anything more complicated than

```
# {user.password}
```

), we have to implement the "interaction" method. That is, this method is written in the class that connects the two pages, in the back-end, as we already know from all the previous examples. Here it is:

```
01 package com.javacodegeeks.jsf.attribute;
02
03 import java.io.Serializable;
04
05 import javax.faces.bean.ManagedBean;
06 import javax.faces.bean.SessionScoped;
07 import javax.faces.event.ActionEvent;
08
09 @ManagedBean(name="user")
10 @SessionScoped
11 public class UserBean implements Serializable {
12
13     /**
14      *
15      */
16     private static final long serialVersionUID = 1L;
17
18     private String password;
19
20     // The action listener method.
21     public void actionListenerSampleMethod(ActionEvent event) {
22         password = (String)event.getComponent().getAttributes().get("password");
23     }
24
25     public String getPassword() {
26         return password;
27     }
28
29     public void setPassword(String password) {
30         this.password = password;
31     }
32 }
```

This was an example of Param and Attribute in JSF 2.0.

Tagged with: ECLIPSE

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:

Sign up

E ganhe dinheiro com o seu
negócio

APRENDA MAIS

GET THE JAVA SKILLS YOU NEED IN 2016.

[Start Learning](#)[Courses](#)[News](#)[Resources](#)[Tutorials](#)[Whitepapers](#)

THE CODE GEEKS NETWORK

[.NET Code Geeks](#)[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 solve File Not Found Exception](#)[java.lang.arrayindexoutofboundsexception – 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 developers. 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 on this site are not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

