You have built a collection of custom tags for your web application. The TLD file is located in the file: /WEB-INF/myTags.xml. You refer to these tags in your JSPs using the symbolic name: myTags. Which deployment descriptor element must you use to make this link between the symbolic name and the TLD file name?  
**A. <taglib>**  
<name>myTags</name>  
<location>/WEB-INF/myTags.xml</location>  
</taglib>  
**B. <tags>**  
<name>myTags</name>  
<location>/WEB-INF/myTags.xml</location>  
</tags>  
**C. <tags>**  
<tags-uri>myTags</tags-uri>  
<tags-location>/WEB-INF/myTags.xml</tags-location>  
</tags>  
**D. <taglib>**  
<taglib-uri>myTags</taglib-uri>  
<taglib-location>/WEB-INF/myTags.xml</taglib-location>  
</taglib>

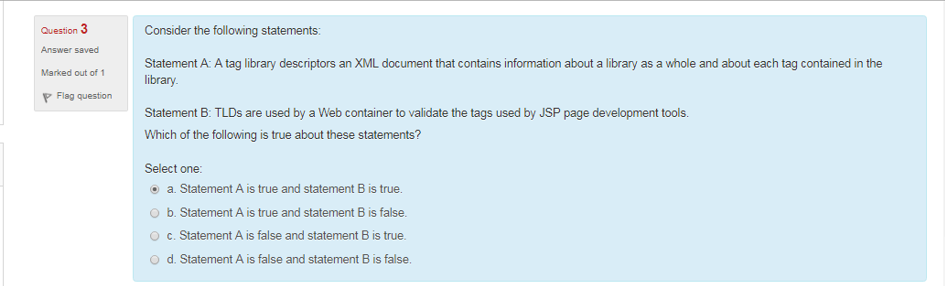
[HIDE ANSWERS](https://vceguide.com/which-deployment-descriptor-element-must-you-use-to-make-this-link-between-the-symbolic-name-and-the-tld-file-name/)

**Correct Answer: D**  
Explanation/Reference:  
*Deployment descriptor to link the tag name and the actual file name.*

A developer wants to make a name attribute available to all servlets associated with a particular user, across multiple requests from that user, from the same browser instance.  
Which two provide this capability from within a tag handler? (Choose two)  
**A. pageContext.setAttribute(“name”, theValue);**  
**B. pageContext.setAttribute(“name”, getSession() );**  
**C. pageContext.getRequest().setAttribute(“name”, theValue);**  
**D. pageContext.getSession().setAttribute(“name”, theValue);**  
**E. pageContext.setAttribute(“name”, theValue, PageContext.PAGE\_SCOPE);**  
**F. pageContext.setAttribute(“name”, theValue, PageContext.SESSION\_SCOPE);**

[HIDE ANSWERS](https://vceguide.com/which-two-provide-this-capability-from-within-a-tag-handler-2/)

**Correct Answer: DF**  
Explanation/Reference:

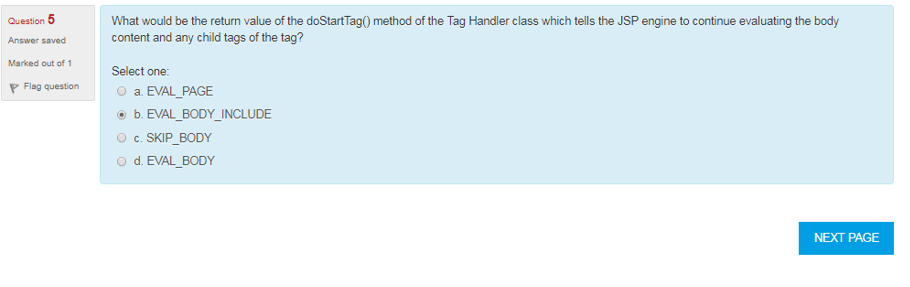


A custom tag is defined to take three attributes. Which two correctly invoke the tag within a JSP page? (Choose two.)  
**A. <prefix:myTag a="foo" b="bar" c="baz" />**  
**B. <prefix:myTag attributes={"foo","bar","baz"} />**  
**C. <prefix:myTag jsp:attribute a="foo" b="bar" c="baz" />**  
**D. <prefix:myTag>**  
<jsp:attribute a:foo b:bar c:baz />  
</prefix:myTag>  
**E. <prefix:myTag>**  
<jsp:attribute ${"foo", "bar", "baz"} />  
</prefix:myTag>  
**F. <prefix:myTag>**  
<jsp:attribute a="foo" b="bar" c="baz"/>  
</prefix:myTag>  
**G. <prefix:myTag>**  
<jsp:attribute name="a">foo</jsp:attribute>  
<jsp:attribute name="b">bar</jsp:attribute>  
<jsp:attribute name="c">baz</jsp:attribute>  
</prefix:myTag>

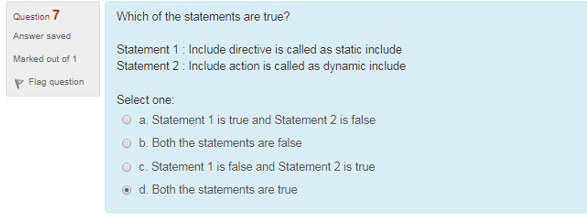
[HIDE ANSWERS](https://vceguide.com/which-two-correctly-invoke-the-tag-within-a-jsp-page-2/)

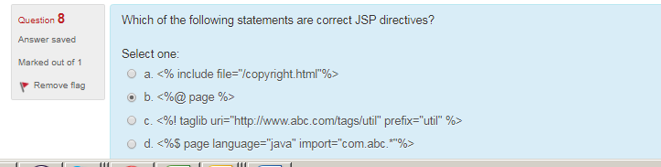
**Correct Answer: AG**  
Explanation/Reference:  
*Other options attribute attribute is not set correctly.*

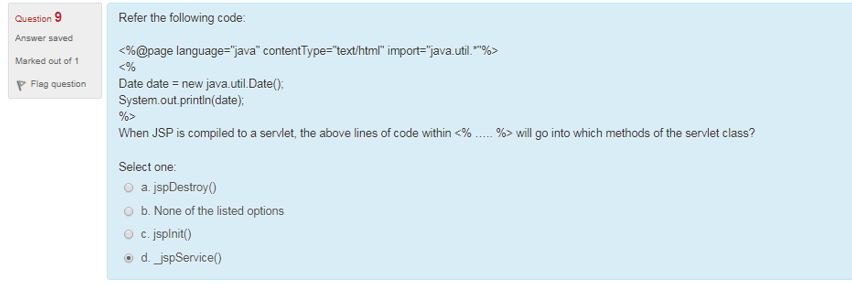
Called after the tag has been initialized, when the JSP engine encounters the opening of a tag at run time. Its return value should be one of two constants defined in the Tag interface: EVAL\_BODY\_INCLUDE, which instructs the JSP engine to evaluate both the tag's body and any child tags it has, or SKIP\_BODY, which instructs the JSP engine to ignore the body.



Include directive is called as static include







|  |  |
| --- | --- |
| **Comment** | <%-- comment --%> |
| **Directive** | <%@ directive %> |
| **Declaration** | <%! declarations %> |
| **Scriptlet** | <% scriplets %> |
| **Expression** | <%= expression %> |

