## Linkovi

<http://stackoverflow.com/questions/5850336/what-does-this-expression-language-pagecontext-request-contextpath-exactly-do>

<a href**="${pageContext.request.contextPath}/JSPAddress.jsp**">Profile</a>

pageContext is EL implicit object/ JSP implicit object. Provides acccess to request, response, session, servletContext objects.

**JSP implicit objects** are the Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.

**The request object** is an instance of a javax.servlet.http.HttpServletRequest object. Each time a client requests a page the JSP engine creates a new object to represent that request.

We can use request object to get the request parameters, cookies, request attributes, session, header information and other details about client request.

PageContext ima getRequest metodu

Request (HttpServletRequest) ima getContextPath() metodu

pisemo **pageContext.request.contextPath**

String getContextPath() - Returns the portion of the request URI that indicates the context of the request.

<a href**="${pageContext.request.contextPath}/JSPAddress.jsp**">Profile</a>

Thus this expression will get the current HttpServletRequest object and get the context path for the current request and append /JSPAddress.jsp to it to create a link (that will work even if the context-path this resource is accessed at changes).

The primary purpose of this expression would be to keep your links 'relative' to the application context and insulate them from changes to the application path.

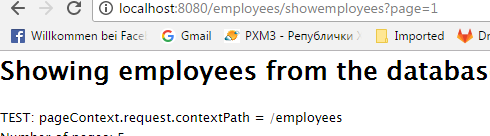
For example, if your JSP (named thisJSP.jsp) is accessed at http://myhost.com/**myWebApp**/thisJSP.jsp, the **context path** will be **myWebApp**. Thus, the link href generated will be /**myWebApp**/JSPAddress.jsp.

If someday, you decide to deploy the JSP on another server with the context-path of corpWebApp, the href generated for the link will automatically change to /corpWebApp/JSPAddress.jsp without any work on your part.

Ako pristupam preko http://localhost:8080/**myWebApp**/ JSPAddress.jsp the **context path** will be **myWebApp.** The link href generated will be /**myWebApp**/JSPAddress.jsp. Ako se context path promeni na npr. corpWebApp radice mi linkovi bez izmene u kodu, href ce generisati /**corpWebApp**/JSPAddress.jsp

Moja aplikacija daje sledece:

pageContext.request.contextPath = **/employees**



<img src="http://localhost:8080/employees/photo?id=${employee.id}" height="100" />

<a href="http://localhost:8080/employees/employee?id=${employee.id}">${employee.firstName} ${employee.lastName}</a>

**A mozda je bolje koristiti c:url koji ce automatski da doda contextPath?**

In particular, **<c:url>** provides three elements of functionality that are particularly important when constructing URLs for J2EE Web applications:

* Prepending the name of the current servlet context
* URL re-writing for session management
* URL encoding of request-parameter names and values

Listing shows the syntax for the <c:url> tag. The value attribute is used to specify a base URL, which the tag then transforms as necessary. If this base URL starts with a forward slash, then a servlet context name will be prepended. An explicit context name can be provided using the context attribute. If this attribute is omitted, then the name of the current servlet context will be used. This is particularly useful because servlet context names are decided during deployment, rather than during development. (If the base URL does not start with a forward slash, then it is assumed to be a relative URL, in which case the addition of a context name is unnecessary.)

<c:url value="expression" context="expression"

var="name" scope="scope">

<c:param name="expression" value="expression"/>

...

</c:url>

Finally, if any request parameters are specified through nested <c:param> tags, then their names and values will be appended to the generated URL using the standard notation for HTTP GET requests. In addition, URL encoding is performed: any characters present in either the names or values of these parameters that must be transformed in order to yield a valid URL will be translated appropriately. Generating a URL with request parameters

<c:url value="/content/search.jsp">

  <c:param name="keyword" value="${searchTerm}"/>

  <c:param name="month" value="02/2003"/>

</c:url>

primeri

<td><a href="<c:url value="/categories/edit/${category.id}"/>">Edit</a></td>

<td><a href="<c:url value="/categories/remove/${category.id}"/>">Remove</a></td>

Umesto:

<img src="http://localhost:8080/employees/photo?id=${employee.id}" height="100" />

<a href="http://localhost:8080/employees/employee?id=${employee.id}">

${employee.firstName} ${employee.lastName}

</a>

Stavi:

<img src = "**<c:url** value="/photo?id=${employee.id}"**/>**" height="100" />

<a href = "**<c:url** value="/employee?id=${employee.id}"**/>**" >

<c:out value="${employee.firstName}"/> <c:out value="${employee.lastName}"/>

</a>

Klik na prvog daje:

http://localhost:8080/employees/employee?id=1