**Exception:**

org.apache.jasper.JasperException: Unable to compile class for JSP

An error occurred at line: [1259] in the generated java file: [\...\...\EventPage\_jsp.java]

The code of method \_jspService(HttpServletRequest, HttpServeletResponse) is exceeding the limit 65535 bytes limit.

**Exception Overview:**

Because a JSP page compiles to a Java class, and there's a hard limit of 64k on the size of a Java class. This is a limitation of JVM specification.

So when JSP is converted to \_jspService method and if method has exceeded the 64kb limit you get "JSP file size limit exceeds..."

**Reason:**

* The major reason that JSPs get this big is because they are chock full of reams of Java scriptlet code.
* Static includes (<%@ include file="\_\_\_\_\_\_\_.jsp" %>)

**Recommended solutions:**

* The actual fix is to refactor the JSP to contain no Java code as modern standards dictate (JSTL (JSP Standard Tag Library)/EL/extract the java code from JSP to a java class and invokes when it is required as a method call).
* Replace static includes with dynamic includes for example <%@ include file="\_\_.jsp" %> to <jsp:include page="\_\_\_\_\_\_.jsp" />
* You could do break down each of the forms into its own JSP, and then include them using <jsp:include . . . >. This way, each JSP is its own servlet, which is called at runtime, and each should compile under the 65k limit.

**Note**: Include directive inserts the source of \_\_.jsp at translation time but the action tag inserts the response of \_\_.jsp at runtime

The results of including static and dynamic resources are quite different.

* If the resource is static, its content is inserted into the calling JSP file.
* If the resource is dynamic, the request is sent to the included resource, the included page is executed, and then the result is included in the response from the calling JSP page.

**Examples 1:**

The below scriplet code displaying context path

<%= request.getContextPath() %>

This can be replaced by the below JSP Expression Language (EL) code

${pageContext.request.contextPath}

**Examples 2:**

<% String name = request.getParameter(“name”); %>

<% out.println(name); %>

<% if(name.equals(“xxx”){

out.println(“Hai xxxx”);

%>

The above code can be replaced by

<c:set var=”name” value=”${param.name}”/>

<c:out value=”${name}” />

<c:if test=”${name eq ‘xxx’}” >

<c:out value=”Hai xxxx”>

</c:if>

**Example 3:**

<body>

<ul class="product-list">

<%

for (Product product : getProducts()) {

out.print("<li>" + product.getName() + "</li>");

}

%>

</ul>

</body>

The above java code can be replace with the following JSTL

<c:forEach var="product" items="${products}">

<li>${product.name}</li>

</c:forEach>