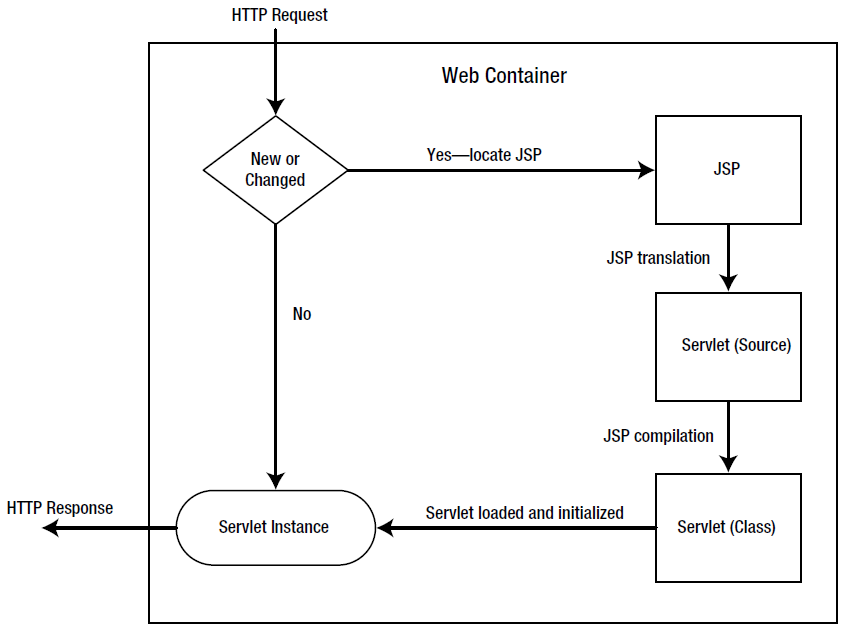
JSP life cycle



The life cycle of a JSP page can be split into four phases: **translation, initialization, execution,** and **finalization**.

Translation

* a request is first made for a JSP page
* first it checks if the JSP page is **new** or has **changed**
* if so, the **JSP engine** will examine the JSP file to check that it?s correctly formed and that the JSP syntax is correct
* if so, the JSP Engine will translate the JSP page into **its page implementation class**, which takes the form of a **standard Java servlet** (the servlet will extend **HttpJspBase** which extends HttpServlet)
* the page implementation class will **be compiled into a class file** by the JSP engine and will be ready for use

If it was not changed the JSP file since its last translation, the servlet instance created before is used.

Initialization

The JSP engine **loads** the generated class file and **creates an instance of the servlet**. It is called **jspInit()** method which has an identical behavior to the standard servlet init() method.

**jspInit() is automatically generated** during the translation phase, but **it's possible to override this method** in the JSP page by using a declaration. The method can be used for initializing application-level variables or resources, for example:

<%! AppVar appvar = null; %>  
<%!  
  public void jspInit() {  
    try {  
      appvar = initAppVar(...);  
    } catch (Exception e){  
      //handle exception  
    }  
  }  
%>

Execution

The initial request can be serviced. For each request, the web container calls in a separate thread the **\_jspService()** method of the implementation servlet.

The \_jspService() method provides all the functionality for handling a request and returning a response to the client. All **the scriptlets and expressions end up inside this method**, in the order in which they were declared inside the JSP page.

Notice that JSP declarations and directives **aren't included** inside this method because they apply to the entire page, not just to a single request, and therefore exist outside the method.

The \_jspService() method **may not be overridden** in the JSP page.

Finalization

It is called the **jspDestroy()** method which has the same behavior as destroy() method found in a standard servlet. This method **can be overridden** in the JSP page.

For example, to release the application resource you opened inside the jspInit() method, you would use the following:

<%!  
public void jspDestroy() {  
  try {  
    appVar.release();  
  } catch (Exception e){}  
  appVar = null;  
}  
%>