

Nutech Computer Training Institute

1682 E. Gude Dr. #102, Rockville, MD. 20850, 8300 Arlington Blvd., #B1 Fairfax VA 22031
Web: nutechtraining.com Tel: 301-610-9300

Java Web Components Developer Objectives Overview

Section 1: The Servlet Technology Model

- For each of the HTTP Methods (such as GET, POST, HEAD, and so on) describe the purpose of the method and the technical characteristics of the HTTP Method protocol, list triggers that might cause a Client (usually a Web browser) to use the method; and identify the HttpServlet method that corresponds to the HTTP Method.
- Using the HttpServletRequest interface, write code to retrieve HTML form parameters from the request, retrieve HTTP request header information, or retrieve cookies from the request.
- Using the HttpServletResponse interface, write code to set an HTTP response header, set the content type of the response, acquire a text stream for the response, acquire a binary stream for the response, redirect an HTTP request to another URL, or add cookies to the response.
- Describe the purpose and event sequence of the servlet life cycle: (1) servlet class loading, (2) servlet instantiation, (3) call the init method, (4) call the service method, and (5) call destroy method.

Section 2: The Structure and Deployment of Web Applications

- Construct the file and directory structure of a Web Application that may contain (a) static content, (b) JSP pages, (c) servlet classes, (d) the deployment descriptor, (e) tag libraries, (d) JAR files, and (e) Java class files; and describe how to protect resource files from HTTP access.
- Describe the purpose and semantics for each of the following deployment descriptor elements: error-page, init-param, mime-mapping, servlet, servlet-class, servlet-mapping, servlet-name, and welcome-file.
- Construct the correct structure for each of the following deployment descriptor elements: error-page, init-param, mime-mapping, servlet, servlet-class, servlet-mapping, servlet-name, and welcome-file.
- Explain the purpose of a WAR file and describe the contents of a WAR file, how one may

be constructed.

Section 3: The Web Container Model

- For the ServletContext initialization parameters: write servlet code to access initialization parameters; and create the deployment descriptor elements for declaring initialization parameters.
- For the fundamental servlet attribute scopes (request, session, and context): write servlet code to add, retrieve, and remove attributes; given a usage scenario, identify the proper scope for an attribute; and identify multi-threading issues associated with each scope.
- Describe the Web container request processing model; write and configure a filter; create a request or response wrapper; and given a design problem, describe how to apply a filter or a wrapper.
- Describe the Web container life cycle event model for requests, sessions, and web applications; create and configure listener classes for each scope life cycle; create and configure scope attribute listener classes; and given a scenario, identify the proper attribute listener to use.
- Describe the RequestDispatcher mechanism; write servlet code to create a request dispatcher; write servlet code to forward or include the target resource; and identify and describe the additional request-scoped attributes provided by the container to the target resource.

Section 4: Session Management

- Write servlet code to store objects into a session object and retrieve objects from a session object.
- Given a scenario describe the APIs used to access the session object, explain when the session object was created, and describe the mechanisms used to destroy the session object, and when it was destroyed.
- Using session listeners, write code to respond to an event when an object is added to a session, and write code to respond to an event when a session object migrates from one VM to another.
- Given a scenario, describe which session management mechanism the Web container could employ, how cookies might be used to manage sessions, how URL rewriting might be used to manage sessions, and write servlet code to perform URL rewriting.

Section 5: Web Application Security

- Based on the servlet specification, compare and contrast the following security mechanisms: (a) authentication, (b) authorization, (c) data integrity, and (d) confidentiality.
- In the deployment descriptor, declare a security constraint, a Web resource, the transport

guarantee, the login configuration, and a security role.

- Compare and contrast the authentication types (BASIC, DIGEST, FORM, and CLIENT-CERT); describe how the type works; and given a scenario, select an appropriate type.

Section 6: The JavaServer Pages (JSP) Technology Model

- Identify, describe, or write the JSP code for the following elements: (a) template text, (b) scripting elements (comments, directives, declarations, scriptlets, and expressions), (c) standard and custom actions, and (d) expression language elements.
- Write JSP code that uses the directives: (a) 'page' (with attributes 'import', 'session', 'contentType', and 'isELIgnored'), (b) 'include', and (c) 'taglib'.
- Write a JSP Document (XML-based document) that uses the correct syntax.
- Describe the purpose and event sequence of the JSP page life cycle: (1) JSP page translation, (2) JSP page compilation, (3) load class, (4) create instance, (5) call the `jspInit` method, (6) call the `_jspService` method, and (7) call the `jspDestroy` method.
- Given a design goal, write JSP code using the appropriate implicit objects: (a) request, (b) response, (c) out, (d) session, (e) config, (f) application, (g) page, (h) pageContext, and (i) exception.
- Configure the deployment descriptor to declare one or more tag libraries, deactivate the evaluation language, and deactivate the scripting language. 6.7 Given a specific design goal for including a JSP segment in another page, write the JSP code that uses the most appropriate inclusion mechanism (the include directive or the `jsp:include` standard action).

Section 7: Building JSP Pages Using the Expression Language (EL)

- Given a scenario, write EL code that accesses the following implicit variables including `pageScope`, `requestScope`, `sessionScope`, and `applicationScope`, `param` and `paramValues`, `header` and `headerValues`, `cookie`, `initParam` and `pageContext`.
- Given a scenario, write EL code that uses the following operators: property access (the `.` operator), collection access (the `[]` operator).
- Given a scenario, write EL code that uses the following operators: arithmetic operators, relational operators, and logical operators.
- Given a scenario, write EL code that uses a function; write code for an EL function; and configure the EL function in a tag library descriptor.

Section 8: Building JSP Pages Using Standard Actions

- Given a design goal, create a code snippet using the following standard actions:

jsp:useBean (with attributes: 'id', 'scope', 'type', and 'class'), jsp:getProperty, and jsp:setProperty (with all attribute combinations).

- Given a design goal, create a code snippet using the following standard actions: jsp:include, jsp:forward, and jsp:param.

Section 9: Building JSP Pages Using Tag Libraries

- For a custom tag library or a library of Tag Files, create the 'taglib' directive for a JSP page.
- Given a design goal, create the custom tag structure in a JSP page to support that goal.
- Given a design goal, use an appropriate JSP Standard Tag Library (JSTL v1.1) tag from the "core" tag library.

Section 10: Building a Custom Tag Library

- Describe the semantics of the "Classic" custom tag event model when each event method (doStartTag, doAfterBody, and doEndTag) is executed, and explain what the return value for each event method means; and write a tag handler class.
- Using the PageContext API, write tag handler code to access the JSP implicit variables and access web application attributes.
- Given a scenario, write tag handler code to access the parent tag and an arbitrary tag ancestor.
- Describe the semantics of the "Simple" custom tag event model when the event method (doTag) is executed; write a tag handler class; and explain the constraints on the JSP content within the tag.
- Describe the semantics of the Tag File model; describe the web application structure for tag files; write a tag file; and explain the constraints on the JSP content in the body of the tag.

Section 11: J2EE Patterns

- Given a scenario description with a list of issues, select a pattern that would solve the issues. The list of patterns you must know are: Intercepting Filter, Model-View-Controller, Front Controller, Service Locator, Business Delegate, and Transfer Object.
- Match design patterns with statements describing potential benefits that accrue from the use of the pattern, for any of the following patterns: Intercepting Filter, Model-View-Controller, Front Controller, Service Locator, Business Delegate, and Transfer Object.