Java EE

* Java EE is an open, standards-based development and deployment platform for creating distributed, transactional, reliable, secure, multitiered, web-based, server-centric, component-based enterprise applications
* Java EE Components
  + Java EE Clients
    - Application Clients,Applets (embedded in web clients)
  + Web Components
    - Servlets, JavaServer Pages (JSP), JavaServer Faces (JSF)
    - Enterprise JavaBeans (EJBM)
  + Java EE Containers
    - Client containers, web container, EJB container
  + Java EE Server

Java EE Web Application

* Collection of resources installed under a specific subset of the URL namespace of a web application server compliant with the Java EE Specification (e.g. Apache’s Tomcat, Apache’s Geronimo, Sun Microsystems‟ Glassfish, IBM‟s WebSphere, etc.)
* Resources
  + - * Static resources: web pages, images, stylesheets
      * Dynamic resources: servlets, JSP’s
      * Miscellaneous resources: business object classes (e.g. Java Beans, EJB), support libraries
    - XML-formatted descriptor and configuration files
      * Web.xml, application.xml, context.xml
    - Organized into a standard hierarchical structure and typically packaged and deployed as WAR or EAR files
  + Java EE API’s
    - Enterprise JavaBeans Technology
    - Java Servlet Technology
    - Java Server Pages
    - Java Server Pages Standard Tag Library
    - Java Server Faces
    - Java Msg Service API
    - Java Transaction API
    - JavaMail API
    - JavaBeans Activation Framework
    - Java API for XML Processing

Servlets

* + Java Object based on the Servlet API
    - Runs in a server application to answer client requests; technically, servlets are not tied to a specific client-server protocol, but they are most commonly used with HTTP and the term „servlet‟ is often used in the context of an “HTTP Servlet”
    - Web-tier components in the Java EE architecture
    - Runs in, and is managed by, a web-tier container called the ‘Servlet Container’
    - Mapped to URLs to which clients send requests
  + Typically asked with (among other things)
    - Processing and/or storing data submitted vial HTML forms
* Generating dynamic content

javax.servlet

* Servlet, GenericServlet
* ServletRequest, ServletResponse
* ServletConfig, ServletContext
* RequestDispatcher
  + javax.servlet.http
    - HttpServlet
    - HttpServletRequest
    - HttpServletResponse
    - HttpSession
    - Cookie
  + Servlet Processing
    - Client sends a request to a web server URL that is mapped to a servlet. Web server passes on the request to the servlet container
    - Servlet container checks if servlet is already loaded
    - If it is not yet loaded, servlet container loads the servlet class and instantiates the servlet, and calls its init method.
    - Servlet container invokes the servlet's service method, passing request and response objects as arguments
    - Servlet processes the request using the response object to create the response, which is returned by the servlet container to the web server, which in turn sends the response to the client
    - Subsequent request to the servlet will not require servlet re-instantiation, unless the servlet has been unloaded; before a servlet is unloaded, the servlet container invokes its destroy method.

* + - init(config)

Invoked once on the servlet by the servlet container when the servlet is instantiated; can be used by the servlet for one-time startup initialization

* + - service(request, response)

Invoked each time the servlet is called upon to process a request (typically on a separate thread for each call)

In HttpServlet, the default Service implementation maps the call to a specific doXXX() method (e.g. doGet, doPost) which is typically overridden to affect the servlet’s functionality x Destroy()

Invoked on the servlet by the servlet container when the servlet is to be unloaded (e.g. when the application is stopped or undeployed); can be used by the servlet for clean-up processing (e.g. resource deallocation)

* + - * Servlet Request Processing (HttpServletRequest)

Retrieving user-supplied request parameters

Retrieving request header values

* + - * Servlet Response Processing (HttpServletResponse)

Setting response status code

Setting response headers

Obtaining output object for sending the response

* Session Tracking(HttpSession)
  + - * Session tracking support is implemented either cookies or URL-rewriting
      * Obtaining session object from the current request

HttpSession session;

session = request.getSession(createNew);

* + - Obtaining session information (HttpSession)

getCreationTime(), getLastAccessedTime(), getMaxInactiveInternal(), getId(), isNew(), setMaxInactiveInterval(int val)

* + - * destroying sessions  invalidate()
      * URL-rewriting(HttpServletResponse)

encodeURL(String url), encodeRedirectURL(String url)

* Java Server Pages
  + - * Simply an HTML web page that contains additional bits of code that execute application logic to generate dynamic content.
      * Java Server Pages Actions (JSP tags) perform a variety of functions and extend the capabilities of JSP.
      * Java Server Pages Actions use XML-like syntax, and are used to manage JavaBeans component.
    - Directives are instructions that are processed by the JSP engine when the page is compiled to a servlet.
    - Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries
      * Motivation
        + It is typically a good idea to separate business logic from presentation concern

Allows modern web development teams to be divided up into programmers and web page authors / designers

Fosters component reuse (e.g. the same data object can be consumed by user agents of varying capabilities and needs)

Servlets can be very powerful for programming business logic, but are very awkward to use when generating static (i.e. template) content.

(X)HTML marked-up documents are very convenient for static content generation but cannot be used to program business logic (or generate dynamic content arising from data produced by the business logic).

* Features

Text-based document capable of generating both static and dynamic content (typically intermixed)

* + - * + Mark-up based document syntax (JSP-style or XML-style), combining (X) HTML elements as well as standard and custom JSP elements; thus, web page authors can feel right “at home” with the mark-up syntax.

Embedded Java Coding support via “scriptlets”  o Template text are converted into JSPWriter

* Components

Template (i.e. static) text

JSP elements

Directives

<%@ page ContentType=”text/html” pageEncoding=”UTF-8” %>

<%@ page import=”java.util.Random” %>

autoFlush

buffer

contentType

errorPage

extends

import

info

isELIgnored

isErrorPage

isThreadSafe

language

pageEncoding

session

<%@ taglib uri=<http://java.sun.com/jsp/jstl/core>prefix=”c” %>

Prefix

Taglib

Uri

Scripting Elements

Declarations

<%! Int a = 100; %>

<%! Int square(int n) { return n\*n ; } %>

Expressions

<% String s = new java.util.Date().toString(); %>

Scriplets

<% for (int I = 0; i < 10; i++) { out.println(i);} %>

Actions

Standard actions

<jsp:directive.include>, <jsp:directive.page>

<jsp:declarations>

<jsp:expressions>

<jsp:include>,<jsp:forward>

<jsp:text>

Custom Actions (JSTL)

JSTL, user-written custom tag libraries

Expression Language(EL)

$ { }

Implicit Scripting Objects

Request, response, out, pageContext

Session, pageContext, application

Config, page, exception

Implicit EL Objects

pageContext

pageScope

requestScope

sessionScope

applicationScope

param, paramValue

header, headerValue

cookie

initparam

Comment

<% --- this is a JSP comment ---- %>

* + - * + Model-View Controller Design Pattern

Model

Represents business objects (logic and state)

View

Presentation of the model in some appropriate way

Controller

Mediates application flow