JSP, Servlet and JSF

JSP (JavaServer Pages)

JSP is a **Java view technology** running on the server machine which allows you to write template text in (the client side languages like HTML, CSS, JavaScript and so on). JSP supports taglibs, which are backed by pieces of Java code that let you control the page flow or output dynamically. A well known taglib is JSTL. JSP also supports Expression Language, which can be used to access backend data (via attributes available in page, request, session and application scopes), mostly in combination with taglibs.

When a JSP is requested for the first time or when the webapp starts up, the servlet container will compile it into a class extending `HttpServlet` and use it during the webapp's lifetime. You can find the generated source code in the server's work directory. In for example Tomcat, it's the `/work` directory. On a JSP request, the servlet container will execute the compiled JSP class and send the generated output (usually just HTML/CSS/JS) through the webserver over network to the client side, which in turn displays it in the web browser.

Servlets

Servlet is an **Java application programming interface (API)** running on the server machine, which intercepts requests made by the client and generates/sends a response. A well known example is the `HttpServlet` which provides methods to hook on HTTP requests using the popular HTTP methods such as `GET` and `POST`. You can configure `HttpServlet`s to listen on a certain HTTP URL pattern, which is configurable in `web.xml`, or more recently with Java EE 6, with `@WebServlet` annotation.

When a Servlet is first requested or during webapp startup, the servlet container will create an instance of it and keep it in memory during the webapp's lifetime. The same instance will be reused for every incoming request whose URL matches the servlet's URL pattern. You can access the request data by `HttpServletRequest` and handle the response by `HttpServletResponse`. Both objects are available as method arguments inside any of the overridden methods of `HttpServlet`, such as `doGet()` and `doPost()`.

JSF (JavaServer Faces)

JSF is a **component based MVC framework** which is built on top of the Servlet API, and provides components via taglibs which can be used in JSP or any other Java based view technology such as Facelets. Facelets is much more suited to JSF than JSP. It namely provides great templating capabilities such as composite components, while JSP basically only offers the `<jsp:include>` for templating, so that you're forced to create custom components with raw Java code (which is a bit opaque and a lot of tedious work in JSF)

when you want to replace a repeated group of components with a single component. Since JSF 2.0, JSP has been deprecated as view technology in favor of Facelets.

As being a MVC (Model-View-Controller) framework, JSF provides the `FacesServlet` as the sole request-response *Controller*. It takes all the standard and tedious HTTP request/response work from your hands, such as gathering user input, validating/converting them, putting them in model objects, invoking actions and rendering the response. This way you end up with basically a JSP or Facelets (XHTML) page for *View* and a Javabean class as *Model*. The JSF components are been used to bind the view with the model (such as your ASP.NET web control does) and the `FacesServlet` uses the *JSF component tree* to do all the work.

Related questions

- What is the main-stream Java alternative to ASP.NET / PHP?
- Java EE web development, what skills do I need?
- How do servlets work? Instantiation, session variables and multithreading
- What is a Javabean and where are they used?
- How to avoid Java code in JSP files?
- What components are MVC in JSF MVC framework?
- What is the need of JSF. When UI can be achieved from css html javascript jQuery?