فصل اول

شروع کار با صفحات تحت سرور جاوا

Jsp یک تکنولوژی تحت جاوا است که برای ساده‌سازی پردازش درخواست‌های تحت وب برروی سرور اجرا می‌شود. بسیاری از وب‌سایت‌هایی که در طول روز از آنها دیدن می‌کنید ممکن است از *jsp* برای قالب‌بندی ونمایش اطلاعات به شما استفاده کنند.

در این فصل مشخص می‌کنیم که *jsp* چیست وعملکرد آن چگونه بوده ودلیل اهمیت آن چیست.

در این فصل همچنین سیر تکامل پردازش درخواست‌ها با استفاده از منطق سرورهای تحت جاوا نیز ارائه شده است، که *Jsp* نقشی حیاتی در این تکامل بازی می‌کند. این نقش که *Jsp* چطور به پردازش درخواست‌های وب کمک می‌کند نیز در ادامه بحث خواهد شد. این بخش به عنوان یک بحث پایه در زمینه ساختن مفاهیم جدید *jsp* وهمچنین برای معرفی خصوصیات جدید *jsp* در فصل‌های بعد بکار رفته است.

تمامی فصل‌های این کتاب شامل مثال‌های عملی برنامه‌نویسی *jsp* هستند. با شروع از این فصل بسیار ابتدایی شما بلافاصله با برنامه‌نویسی *jsp* کار خواهید کرد. این فصل بصورت مفصل چگونگی پیاده‌سازی کد *jsp* در سیستم ویندوز یا در گروه‌های کاری *unix* یا *linux* را نشان می‌دهد.

خصوصا این فصل موارد زیر را شامل می‌شود:

۱- مرور تاریخ سیر تکاملی تکنولوژی وب که به *jsp* منتهی می‌شود.

۲- بحث در مورد دلایل لزوم استفاده از *jsp*.

۳- بیان نحوه عملکرد *jsp*

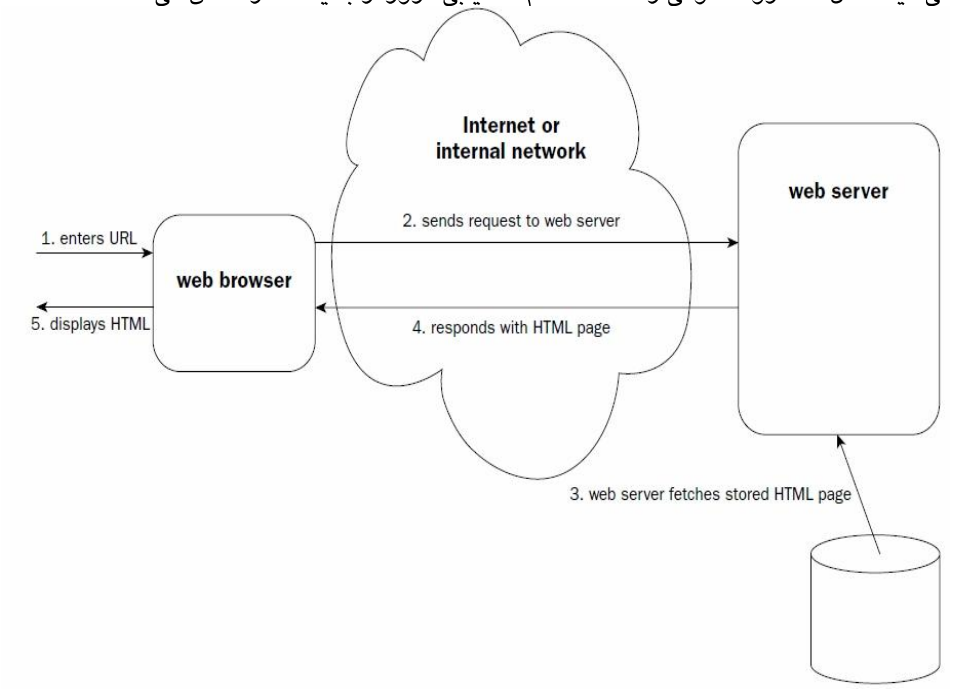
٤- نمایش لینک دانلود مثال‌های کد و مثال‌های پروژه *jsp*.

٥- نمایش لینک دانلود یک سرور برای اجرای کدهای *jsp* روی سیستم خانگی یا ایستگاه کاریمان.

٦- مشخص کردن نحوه تنظیم سرور *tomcat* منبع باز برای اجرای کد *jsp*.

ایجاد برنامه‌های کاربردی برای اینترنت

قبل از جستجو برای سروری که از *jsp* پشتیبانی کند. به این مساله فکر کنید که چه اتفاقاتی درسطح زیرین رخ می‌دهد زمانی که شما از مرورگرتان برای دسترسی به یک سایت استفاده می‌کنید. اگر شما از یکی از مرورگرهای مهم وب مثل (*netscap* و *microsoft internet explorer* و *firefox* و یا *opera*) استفاده می‌کنید، شکل ۱-۱ روند متوالی رخدادها هنگام دستیابی مرورگر به یک *url* را نشان می‌دهد.



شکل ۱-۱: روند دسترسی مرورگر به یک *url*

مراحل زیر به ترتیب عددی با اعداد شکل ۱-۱ مرتبط هستند:

۱- *url* یک صفحه وب را در مرورگرتان وارد کنید. این *url* به مرورگر می‌گوید به یک ماشین مشخص در اینترنت متصل شود.

۲- سپس مرورگر درخواست را برای ماشین مشخص در اینترنت می‌فرستد. ماشین تعیین شده سپس یک بخشی از نرم‌افزار که به آن *webserver* می‌گویند را اجرا می‌کند وب. سرور درخواست را دریافت و بررسی می‌کند. تعدادی از وب سرورهای معروف عبارتند از: *apache, microsoft internet netscape enterprise server , sun java system web server(formely sun one),oracle, information service (iis) http server and zeus web server.*

۳- با توجه به درخواست دریافت شده، وب سرور یک صفحه وب در قالب *html* را ازبین صفحات ذخیره شده بازیابی می‌کند.

٤- صفحه حاصل در مرحله سوم به عنوان پاسخ برای مرورگر ارسال می‌شود.

٥- مرورگر پس از دریافت صفحه وب درخواستی ان را به کاربر نمایش می‌دهد.

البته، صفحه وب می‌تواند شامل اجزا گرافیکی از قبیل فایل‌های *gif* (مرورگر باید درخواست‌های اضافی برای اینگونه فایل‌ها به سرور وب ارسال کند.) باشد. همچنین ابر لینک‌هایی که کاربر می‌تواند بر روی آنها کلیک کند.

Html فرمت استانداردی است که صفحات وب با آن کدنویسی می‌شوند. فایل‌های *html* از نوع فایل‌های متنی‌اند که قابل ویرایش در هر ویرایشگر متنی هستند. یک صفحه *html* از بخش‌هایی به نام تگ از جمله تگ عنوان و تگ بدنه، همچنین اجزا قالب‌بندی برای قسمت طراحی صفحه مانند پاراگراف‌ها و جداول تشکیل شده است. تمامی مرورگرها زبان *html* را پشتیبانی می‌کنند و صفحات را با توجه به تگ‌های قالب‌بندی نمایش می‌دهند. برای کسب اطلاعات بیشتر در مورد *html* می‌توانید کتاب *beginning web programing html,xhtml,css(wrox press-isbn 0-7645-7078-1)* را مطالعه کنید.

در پردازش قبلی، مرورگر از طریق اینترنت با وب سرور مکالمه می‌کند. این محاوره از طریق یک پروتکل استاندارد شبکه انجام می‌شود. که این پروتکل خاص*http* است(پروتکل انتقال *hypertext*). *http* در لایه‌ی بالای*Tcp/Ip* ساخته شده، و شامل مجموعه پروتکلی‌هایی است که تمامی کامپیوترها را در اینترنت به همدیگر متصل می‌کند.

محدودیت‌های مدل وب سرور پایه

تابع اصلی وب سرور آن را محدود کرده تا به یک تعداد معینی از صفحات ایستا سرویس دهد. محتوی هر صفحه بهمان شکل باقی می‌ماند. روش ساده‌ای برای نمایش اطلاعاتی که تغییر می‌کنند، وجود ندارد. مانند اطلاعات آب و هوای امروز یا آخرین اخبار و یا لیست محصولاتی که فروشگاه (*online*) ارائه می‌دهد. برای نمایش اطلاعات جدید مجموعه‌ی جدیدی از صفحات ایستا لازم است که ایجاد شوند.

ایجاد صفحات ایستای جدید برای هر تغییر دقیقه‌ای اطلاعات لایه‌های زیرین، ملال آور و خسته‌کننده است. بدیهی است، اگر راهی برای سرور وجود داشته باشد تا بتواند بخش‌هایی از صفحه *html* را به صورت خودکار ایجاد کند، زمان و انرژی بسیاری را می‌توان ذخیره کرد. و عملکردی اینگونه نیاز به ایجاد مکرر صفحات ایستای جدید در قبال تغییر اطلاعات را منتفی می‌کند. این تولید محتوی باید به صورت پویا هنگام پردازش درخواست‌ها صورت گیرد. به عنوان مثال سرور می‌تواند بخشی از صفحه *html* را تولید کند که تاریخ و زمان جاری را نمایش می‌دهد.

مهندسین نرم‌افزار اینترنت برای فراهم کردن توانایی تولید پویا سریعا تمامی توجه شان را به *cgi*ها معطوف کردند.

تولید *html* پویا از طریق *CGI*

CGI روشی جهت اجرای یک برنامه در سمت سرور را فراهم می‌کند. که این برنامه ممکن است بر روی ماشینی باشد که در حال اجرای وب سرور بوده یا ماشینی که به آن متصل است. خروجی برنامه *cgi*

صفحه *html*ی است که به مرورگر وب جهت نمایش فرستاده می‌شود. شکل ۲-۱ عملیات اصلی *cgi* را شرح می‌دهد.

۱- ابتدا به مرورگر دستور داده می‌شود تا به یک *url*دسترسی یابد. می‌توانید *URL* را به صورت دستی وارد کنید. زمانی که شما یک فرم *online*را پر می‌کنید یا بر لینکی از صفحه‌ای که در حال نمایش است کلیک می‌کنید در واقع یک *url* دستیابی می‌شود. به عنوان مثال:

url: http: //www. wrox. com/begingjsp/cg1test. cgi ممکن است اینگونه باشد.

این *url* به مرورگر می‌گوید تا به ماشین مشخصی در اینترنت که نام آن *www. wrox. com* متصل شود.

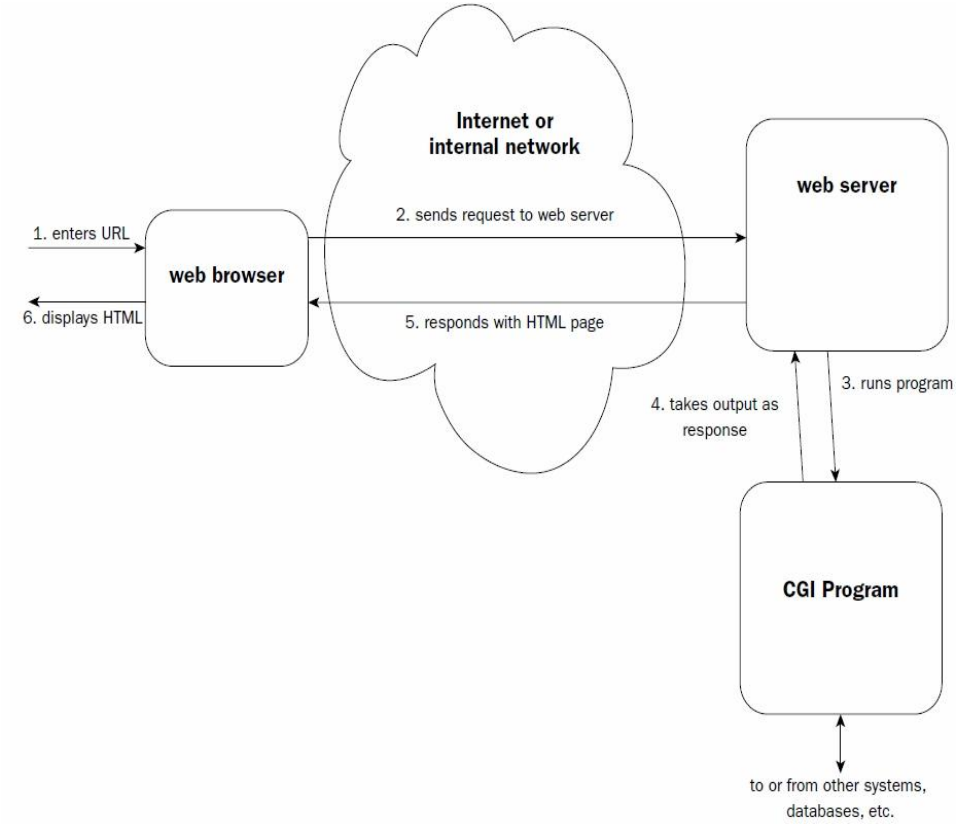
۲- سپس مرورگر درخواست را برای ماشین مشخص شده در اینترنت می‌فرستد. که این عمل همانند حالت بدون *cgi* در شکل ۱-۱ است. *url* علاوه بر ماشین خاص مکان برنامه *cgi* معینی را نیز مشخص می‌کند. بخشی از *url* که مکان *cgi* را مشخص می‌کند عبارت *beginjsp/ch1test. cgi* می‌باشد. وب سرور *url* درخواست ورودی را بررسی کرده و آن را به برنامه *cgi* مشخص هدایت می‌کند. (*ch1test. cgi*).

۳- برنامه *cgi* در قسمت سرور اجرا می‌شود.

٤- خروجی برنامه *cgi* توسط سرور ضبط می‌شود.

٥- خروجی برنامه*cgi* از طریق پروتکل *http* به مرورگر وب در خواست کننده انتقال داده می‌شود.

۶- مرورگر وب کلاینت در نهایت نتایج خروجی برنامه *cgi* را در قالب یک صفحه وب نمایش می‌دهد.



شکل۲-۱: عملیات پایه *CGI*

برنامه *cgi* را می‌توان توسط هر زبان برنامه‌نویسی کامپایلری نوشت. (که معتبرترین زبان‌های *CGI* عبارتند از *prel*، *shellscript* و همچنین از زبان‌های *C,C++, java* هم می‌توان استفاده کرد).

خروجی تولید شده توسط برنامه *cgi* محدود به مجموعه مشخصی از صفحات ایستا نیست. بعلاوه برنامه *cgi* در فرایند تولید خروجی می‌تواند از منابع اضافی دیگری نیز استفاده کند. به عنوان مثال برنامه *cgi*که در حال نمایش اطلاعات کمیت محصولی که در اینترنت برای فروش گذاشته شده می‌تواند به بانک اطلاعاتی که شامل اطلاعات موجودی انبار است دسترسی یابد.

cgi نقاط ضعف مهمی دارد. و این کمبودها، در ادامه همین بخش بیان شده‌اند، زمانی مشهود هستند که یک برنامه *cgi* توسط چند کاربر درخواست می‌شود.

نقاط ضعف *cgi*

ضعف‌های پایه برنامه‌های *cgi* عبارتند از:

۱- سربار مربوط به شروع یک پردازش سیستم عامل برای هر درخواست ورودی.

۲- سربار مربوط به بارگذاری واجرای یک برنامه برای هر درخواست ورودی.
۳- نیاز به کدنویسی خسته کننده وتکراری جهت هدایت پرتکل شبکه و رمزگشایی درخواست.

دو عمل اول مقدار زیادی از سیکل پردازنده و حافظه را مصرف می‌کنند. زیرا هر دو عمل باید برای هر درخواست ورودی اجرا شوند. در صورتی که درخواست‌ها در فاصله زمانی کوتاهی وارد شوند ماشین سرور دچار بار اضافی می‌شود. زیرا برنامه‌های *cgi* از یکدیگر مستقل بوده و اغلب از زبان‌های برنامه‌سازی ناسازگاری منعکس می‌شوند. و همچنین امکان اشتراک‌گذاری کدی که در شبکه و رمز گشایی استفاده می‌شود، وجود ندارد.

*Cgi*هایی که بر پایه جاوا هستند برای هدایت درخواست‌های *cgi* مناسب نیستند زیرا اساسا جاوا یک زبان مفسری است. به همین دلیل، یک برنامه بسیار حجیم که ماشین مجازی جاوا نام دارد باید برای هدایت درخواست ورودی اجرا شود. فرایند اجرای یک پردازش سیستم، وسپس اجرای *jvm* در خلال این پردازش، و سپس اجرای کد *cgi* جاوا در داخل *JVM* تنها برای پردازش یک درخواست، هم از نظر سیکل‌های محاسباتی و هم از نظر منابع بسیار پر هزینه است. بدتر از آن اینکه تکرار تمامی این فرایند برای هر درخواست ورودی الزامی است. هنگامی که با زمان مورد نیاز برای پردازش درخواست و تولید خروجی مقایسه می‌شود. آنگاه این بار اضافی می‌تواند قابل توجه باشد. و اگر سرور قرار باشد درخواست‌های ورودی زیادی را هدایت کند، آنگاه سربار می‌تواند سیستم را مختل کند.

اصلاح *cgi*های جاوا: سرولت‌ها

در صورتی که بتوان سربار را حذف کرد آنگاه *cgi*های جاوا قابل اصلاح خواهند بود. اگر روشی وجود داشته باشد تا بتوان تمام درخواست‌های ورودی را تنها با یک باراجرای یک پردازش سیستم عامل و *jvm* مورد پردازش قرار داد، سربار را می‌توان حذف کرد.

از آنجایی که پلتفرم جاوا می‌تواند به صورت پویا در حین اجرا کلاس‌های جدید را بارگذاری کند، از این قابلیت جاوا می‌توان برای بارگذاری کد جدید جاوا جهت هدایت درخواست‌های ورودی استفاده کرد. بعبارت دیگر، یک پردازش سمت سرور یک بار اجرا شده و همراه با *jvm* یک بار بارگذاری می‌شود. اما کلاس‌های اضافی توسط *jvm* برای پردازش درخواست‌های ورودی بارگذاری می‌شوند. این عملیات به صورت قابل توجهی موثر است، در این سناریو. مراحل زیر قابل مشاهده است:

۱- سربار مربوط به شروع یک پردازش سیستم برای هر درخواست ورودی حذف شده است.
۲- سربار مربوط به بارگذاری *jvm* برای هر درخواست ورودی حذف شده است.
۳- کلاس‌های جاوا توسط *jvm* برای پردازش درخواست‌ها بارگذاری می‌شوند. و اگر بیش از یک درخواست پردازش مشابهی نیاز داشته باشد، کلاسی که قبلاً بارگذاری شده این کار را انجام می‌دهد. که این عمل سربار بارگذاری کلاس برای همه به جز اولین درخواست را حذف می‌کند.
٤- کدی که پروتکل شبکه و رمزگشایی درخواست‌های ورودی را هدایت می‌کرد. می‌تواند توسط کلاس‌های جاوای پردازشگر درخواست که به صورت پویا بارگذاری می‌شوند مشترک باشد.

وب سرور جاوا از سان میکرو سیستم که در اواخر ۱۹۹۰ منتشر شد دقیقا در این مسیر فعالیت می‌کند. این وب سرور که به زبان برنامه‌سازی جاوا نوشته شده، یک *jvm* را برای هدایت و کنترل پیام‌های ورودی راه‌اندازی می‌کند. همچنین درعین حال کلاس‌هایی از جاوا که برای پردازش درخواست‌های خاصی نیاز هستند را بارگذاری می‌کند.

برای اطمینان از اینکه کلاس‌های جاوای هدایت‌کننده‌ی درخواست‌ها به صورت مرحله‌ای و یکی پس از دیگری عمل نکنند، و یا با یکدیگر موجب توقف کامل وب سرور نشوند. یک استاندارد کدنویسی تولید شد، که کلاس‌ها باید از آن تبعیت کنند، به این استاندارد *API* سرولت جاوا می‌گویند. به کلاس‌هایی که بصورت پویا برای عملیات پردازش بارگذاری می‌شوند سرولت(*servlet*) می‌گویند.

قسمتی از کد که عملیات بارگذاری، تخلیه بار، بارگذاری مجدد و اجرای سرولت‌ها را مدیریت می‌کند به آن سرولت کانتینر گفته می‌شود. در صورتی که حافظه سیستم کم باشد یا تعدادی از سرولت‌ها برای مدت طولانی استفاده نشوند باید *unload* شوند. اگر کد سرولت از آخرین زمانی که از آن استفاده شده تغییر کرده باشد آنگاه لازم است تا سرولت مجددا بارگذاری شود.

روش‌های زیادی برای پیکربندی وب سرور و سرولت کانتینر وجود دارد. که بخش بعدی به این موضوع می‌پردازد.

پیکربندی‌های وب سرور: جمع‌بندی سرولت کانتینرها

با توجه به وب سرور و سرولت کانتینر مورد استفاده، پیکربندی‌های متفاوتی امکان‌پذیر است.

تعدادی از معتبر ترین پیکربندی‌ها در زیر آورده شده‌اند:

۱- *jvm* یکسانی وب سرور را بانضمام سرولت‌ها اجرا می‌کند. در این حالت، وب سرور به زبان جاوا کدنویسی شده، همانند وب سرور جاوا که قبلاً به آن اشاره کردیم. اغلب به آن پیکربندی *stand alone* می گویند

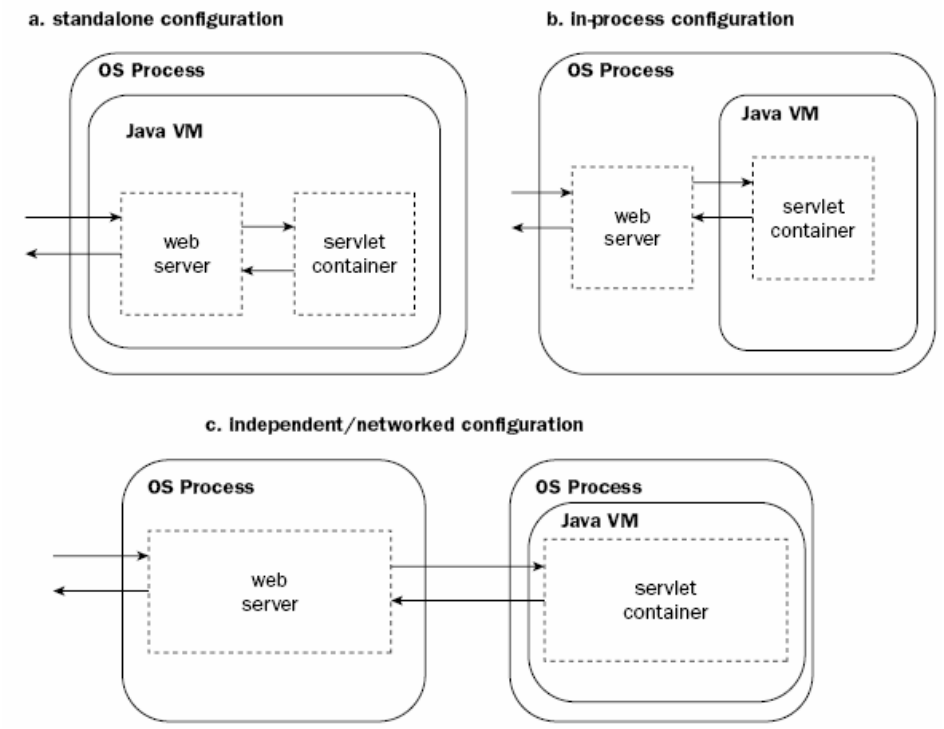
شکل۳-۱ *A* پیکربندی *stand alone* را شرح می‌دهد.

۲- در حالت دوم وب سرور به زبان برنامه‌سازی جاوا نوشته نشده، اما یک *jvm* را در داخل پردازش سیستم عامل یکسانی اجرا می‌کند. در این حالت اطلاعات مستقیماً از وب سرور جاوا به *jvm* که میزبان سرولت کانتینر است منتقل می‌شود. (بعضی از نسخه‌های وب سرور *apache* *iis*، مایکرسافت می‌توانند به این روش عمل کنند.) به این روش پیکربندی *in process* می‌گویند. شکل۳-۱ *B* این پیکربندی را شرح می‌دهد.

۳- وب سرور به زبان برنامه‌سازی جاوا نوشته نشده، و در یک پردازش سیستم مجزا از سرولت کانتینر اجرا می‌شود. در این حالت، وب سرور درخواست‌ها را به سرولت کانتینر با استفاده از یک شبکه محلی یا یک مکانیزم ارتباطی پردازش داخلی خاص سیستم عامل انتقال می‌دهد. که به این روش اغلب پیکربندی مستقل یا پیکربندی شبکه‌ای گویند. شکل۳-۱ *C* این پیکربندی را شرح می‌دهد.

مزیت دو روش اول پیکربندی اشاره شده در لیست بالا این است که *jvm* همانند وب سرور همراه پردازش یکسانی از سیستم عامل اجرا می‌شود. این روش قابلیت انتقال مکرر اطلاعات در خواست و پردازش خروجی به یا از کد *cgi* را فراهم می‌کند. برعکس، اگر سرولت کانتینر یا یکی از سرولت‌های آن از کار بیفتد، آنگاه تمام وب سرور به این دلیل که آنها در یک پردازش هستند ممکن است از کار بیفتد.

در هنگام انتقال داده درخواستی بین وب سرور و سرولت کانتینر، پیکربندی سوم نا کارآمدتر می‌باشد. هر چند سرولت کانتینر فعالیتش می‌تواند متوقف شده ویا دوباره راه‌اندازی شود بدون اینکه تاثیری در عملکرد وب سرور داشته باشد. این حالت سیستم قوی تری برای هدایت درخواست‌های وب ایجاد می‌کند.



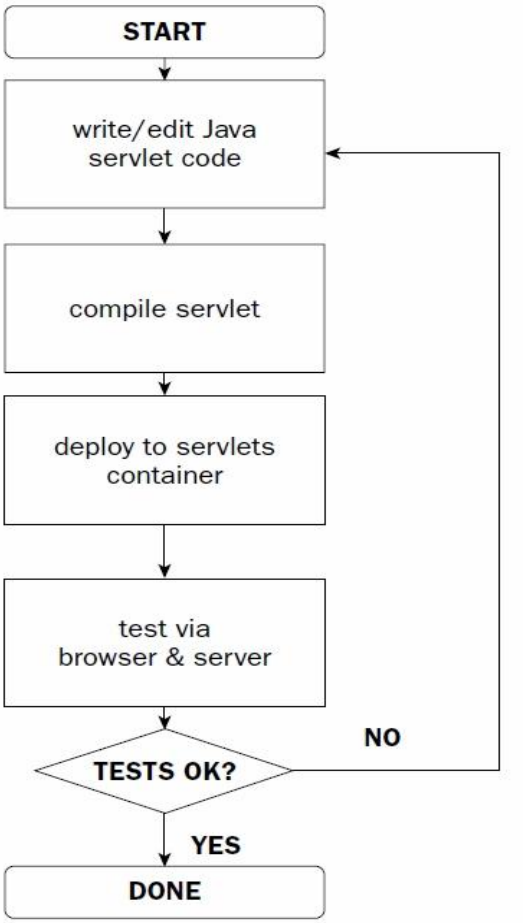
شکل ۳-۱: وب سرور و پیکربندی سرولت کانتینر

هریک از سه پیکربندی که شرح دادیم، ویژگی‌های عملکردی برتری در مقایسه با عملیات *cgi* پایه که قبلاً معرفی شد ارائه می‌دهند.

بهینه‌سازی فرایند توسعه سرولت: صفحات تحت سرور جاوا

سرولت‌ها می‌توانند روش مناسبی برای اجرای عملیات‌های *cgi* توسط زبان برنامه‌سازی جاوا باشند. کلاس‌های جاوای ارائه کننده‌ی سرولت در صورت نیاز به آنها به صورت پویا توسط سرولت کانتینر

بارگذاری می‌شوند، هرچند، هر سرولت می‌تواند از چندین کلاس کامپایل شده جاوا درست شده باشد. یعنی اینکه برنامه‌نویس ابتدا کد جاوا را می‌نویسد، سپس آن را کامپایل کرده، سپس کد را رجیستر می‌کند و از طریق سرولت کانتینر کد را اجرا می‌کند، و اگر نیازی به ویرایش سرولت باشد، انگاه کد منبع جاوا باید ویرایش شده، کامپایل شود و دوباره در سرولت کانتینر قرار گیرد. شکل ٤-۱ مراحل مختلف در فرایند توسعه سرولت را توضیح می‌دهد.



شکل٤-۱: فرایند توسعه سرولت.

به مکانیزم *cgi* پایه که درقسمت‌های قبلی ارائه شد رجوع کنید. بدیهی است که هدف اصلی از بیشتر سرولت‌ها ایجاد یک صفحه *html* است. ازینرو، شما ممکن است سورس کد جاوای سرولتی به شکل زیر ببینید: