**Jetty : 웹서버 + 서블릿 컨테이너(JSP+서블릿)**

WAS와 웹서버를 분리하여 정적인 데이터(html)은 웹서버에서 처리, 동적인 데이터(서블릿)은 WAS에서 처리

* WAS와 WebServer를 분리하지 않는 경우

모든 컨텐츠를 한곳에 집중시켜 웹서버와 WAS의 역할을 동시에 수행, 스위치를 통한 로드 밸러싱, 사용자가 적을 경우 효율적

* WAS와 WebServer를 분리한 경우

웹서버와 WAS의 기능적 분류를 통해 효과적인 분산을 유도, 정적인 데이터는 웹서버에서 처리, 동적인 데이터는 WAS가 처리

* WAS 여러개와 WebServer를 분리한 경우

WAS단을 프리젠테이션 로직과 비즈니스 로직으로 구분하여 구성, 특정 logic의 부하에 따라 적절한 대응할 수 있지만 설계단계 유지보수 단계가 복잡

**서블릿 : 클라이언트의 요청을 처리하고 그 결과를 다시 클라이언트에게 전송 ex)로그인**

특징

* 클라이언트의 요청에 대해 동적으로 작동하는 웹 어플리케이션 컴포넌트
* Html을 사용하여 요청에 응답한다.
* Java Thread를 이용하여 동작한다.
* MVC 패턴에서 Controller로 이용된다.
* HTTP 프로토콜 서비스를 지원하는 javax.servlet.http.HttpServlet 클래스를 상속받는다.
* UDP보다 속도가 느리다.
* Html 변경시 servlet을 재컴파일해야 한다.(jsp 수정시 서버 재가동해야함)

동작방식

1. 사용자(클라이언트)가 URL을 클릭하면 HTTP Request를 Servlet Container로 전송한다.
2. HTTP Request를 전송받은 Servlet Container는 HttpServletRequest, HttpServletResponse 두 객체를 생성한다.
3. web.xml은 사용자가 요청한 URL을 분석하여 어느 서블릿에 요청을 한 것인지 찾는다.
4. 해당 서블릿에서 service 메소드를 호출한 후 클라이언트의 POST, GET 여부에 따라 doGet() 또는 doPost()를 호출한다.
5. doGet(), doPost() 메소드는 동적 페이지를 생성한 후 HttpServletResponse 객체에 응답을 보낸다.
6. 응답이 끝나면 HttpServletRequest, HttpServletResponse 두 객체를 소멸시킨다.

**서블릿 컨테이너 : 서블릿을 관리해주는 컨테이너 ex)톰캣**

역할

* 웹서버와의 통신 지원

서블릿 컨테이너는 서블릿과 웹서버가 손쉽게 통신할 수 있게 해준다. 일반적으로 우리는 소켓을 만들고 listen, accept 등을 해야하지만 서블릿 컨테이너는 이러한 기능을 API로 제공하여 복잡한 과정을 생략할 수 있게 해준다. 그래서 개발자가 서블릿에 구현해야 할 비즈니스 로직에 대해서만 초점을 두게끔 도와준다.

* 서블릿 생명주기 관리

서블릿 컨테이너는 서블릿의 탄생과 죽음을 관리한다. 서블릿 클래스를 로딩하여 인스턴스화하고, 초기화 메소드를 호출하고, 요청이 들어오면 적절한 서블릿 메소드를 호출한다. 또한 서블릿이 생명을 다 한 순간에는 적절하게 가비지 컬렉션을 진행하여 편의를 제공합니다.

* 멀티쓰레드 지원 및 관리

서블릿 컨테이너는 요청이 올 때마다 새로운 자바 쓰레드를 하나 생성하는데 HTTP 서비스 메소드를 실행하고 나면 쓰레드는 자동으로 죽게된다. 원래는 쓰레드를 관리해야 하지만 서버가 다중 쓰레드를 생성 및 운영해주기 때문에 쓰레드의 안정성에 대해 걱정하지 않아도 된다.

* 선언적인 보안 관리
* 서블릿 컨테이너를 사용하면 개발자는 보안에 관련된 내용을 서블릿 또는 자바 클래스에 구현해놓지 않아도 된다. 일반적으로 보안관리는 XML 배포 서술자에 기록하므로 보안에 대해 수정할 일이 생겨도 자바 소스코드를 수정하여 다시 컴파일하지 않아도 보안관리가 가능하다.

**JSP : JAVA 코드가 들어가있는 HTML 코드 ex)PHP**

서블릿은 자바 소스코드 속에 HTML코드가 들어가는 형태인데, JSP는 이와 반대로 HTML 소스코드 속에 자바 소스코드가 들어가는 구조이다.

HTML속에 자바코드는 <% 소스코드 %> 또는 <%= 소스코드 =%>형태로 들어가며 자바 소스코드로 작성된 부분은 웹 브라우저로 보내는 것이 아니라 웹 서버에서 실행된다.

로직과 디자인이 한 파일내에 섞여있어 유지보수가 어렵다.

**Http 통신 : Client의 요청(Request)이 있을때만 서버가 응답(Response)하여 해당 정보를 전송하고 곧바로 연결을 종료하는 방식**

* 클라이언트가 요청을 보내는 경우에만 서버가 응답하는 단방향 통신이다.
* 서버로부터 응답을 받은 후에는 연결이 바로 종료된다.
* 모바일 어플리케이션 개발에 주로 사용되며 비용 및 유지보수 방면에서 좋다.

**Socket 통신 : Server와 Client가 특정 포트를 통해 실시간으로 양방향 통신을 하는 방식**

* 서버와 클라이언트가 계속 연결을 유지하는 양방향 통신이다.
* 실시간으로 데이터를 주고받을 수 있다.
* 실시간 동영상 스트리밍이나 온라인 게임등과 같은 경우에 사용된다.