9장 스프링 프로젝트 시작하기

9.1 자바 엔터프라이즈 플랫폼과 스프링 애플리케이션

9.1.1 클라이언트와 백엔드 시스템

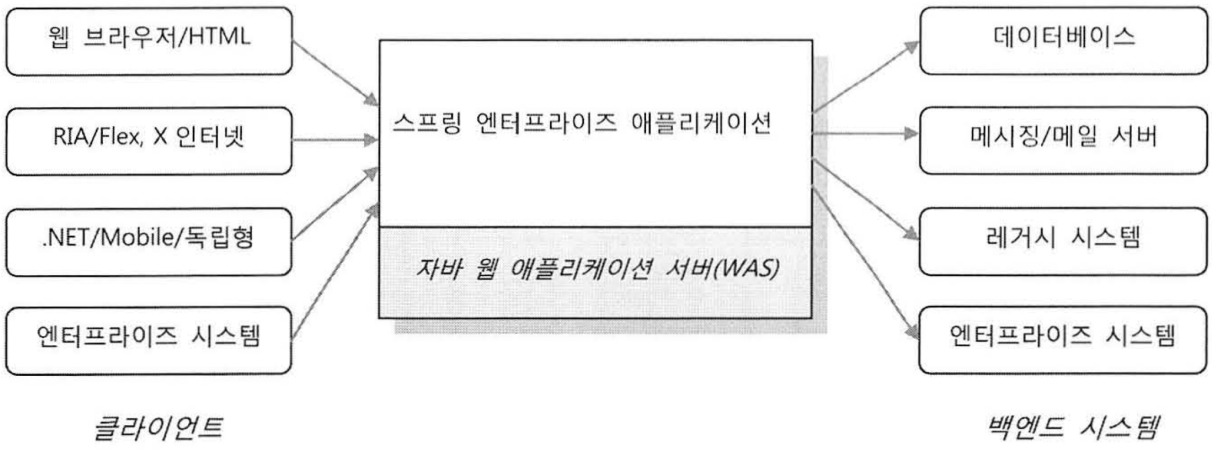

그림 9-1 스프링 엔터프라이즈 애플리케이션의 서비스와 협력 구조

9.1.2 애플리케이션 서버

스프링으로 만든 애플리케이션 자바 서버 환경에 배포하려면 JavaEE(J2EE)서버가 필요하다.

1. 경량급 WAS/서블릿 컨테이너

   스프링은 기본적으로 톰캣이나 제티같은 가벼운 서블릿 컨테이너만 있어도 충분하다. 엔터프라이즈 애플리케이션에 필요한 핵심기능을 모두 이용할 수 있다.

2. WAS

   미션 크리티컬한 시스템에서 요구되는 고도의 한정성이나 고성능 시스템에서 필수적인 안정적인 리소스 관리, 레거시 시스템의 연동이나 기존 EJB로 개발된 모듈을 함께 사용하는 등의 필요가 있다면 상용 또는 오픈소스 WAS를 쓸 수 있다. WAS는 상대적으로 관리 기능이나 모니터링 기능이 뛰어나 여러 대의 서버를 동시 운영할 때 유리한 점이 많다.

- 스프링소스 tcServer

  실제로 개발환경과 운영환경에서 가장 많이 사용되는 자바 서버는 웹 모듈만 지원하는 **서블릿 컨테이너인 아파치 톰캣**이다.

  기존 톰캣+고급서버 관리 기능, 배포기능 진단기능, 기술지원을 받을 수 있다.

9.1.3 스프링 애플리케이션의 배포 단위

- 독립 웹 모듈

  스프링은 보통 `war`로 패키징된 독립 웹 모듈로 배포된다. 단순하고 편리한 배포단위다.

- 엔터프라이즈 애플리케이션

  확장자가 `ear`인 엔터프라이즈 애플리케이션으로 패키징해서 배포할 경우 사용. 하나 이상의 웹모듈과 별도로 분리된 공유 가능한 스프링 컨텍스트를 엔터프라이즈 애플리케이션으로 묶어주는 방법

- 백그라운드 서비스 모듈

  `rar`로 패키징. 리소스 커넥터를 만들어 배포할 때 사용하는 방식으로 만약 스프링으로 만든 애플리케이션이 UI를 가질 필요는 없고 서버 내에 백그라운드 서비스처럼 동작할 필요가 있다면 rar모듈로 만들어 배포할 수 있다. J2EE 1.4이상의 표준을 따르는 WAS가 필요하다.

9.2 개발도구와 환경

9.2.1 JavaSE와 JavaEE

- JavaSE/JDK

  스프링3.0이후 JavaSE 5버젼.JDK5.0 이상을 필요로 한다.

- JavaEE/J2EE

  스프링 3.0 자바 엔터프라이즈 플랫폼엔 J2EE1.4 버젼이나 JavaEE5.0이 필요하다.

9.2.2 IDE

통합개발환경. 이클립스, 넷빈즈, 인텔리제이등이 있다.

9.2.3 SpringSource Tool Suite

이클립스 확장판으로 주요한 스프링 지원 플러그인과 관련 도구를 모아서 스프링 개발에 최적화되도록 만들어진 IDE다.

- SpringIDE 플러그인

  스프링 개발에 유용한 기능을 제공하는 플러그인의 모음이다.

  - 빈 클래스 이름 자동완성
  - 빈 설정 오류 검증
  - 프로젝트 생성, 설정파일 생성, 빈 등록 위저드
  - 빈 의존관계 그래프
  - AOP 적용대상 표시
- STS 플러그인

  스프링 개발과 설정파일 편집을 지원하는 SpringIDE에 더해서 스프링 애플리케이션의 서버 배치와 같은 추가 기능을 제공한다.
- 기타 플러그인
  - M2Eclipse: maven을 지원하는 이클립스 플러그인
  - AJDT: AspectJ AOP를 이용한 개발을 지원하는 편리한 툴
  - VMCI: VMWare서버 또는 워크스테이션과의 연동을 지원하는 플러그인
  - 이클립스 표준 플러그인: WTP, EMP, Mylyn, DSDP

9.2.4 라이브러리 관리와 빌드 툴

- 라이브러리 관리의 어려움

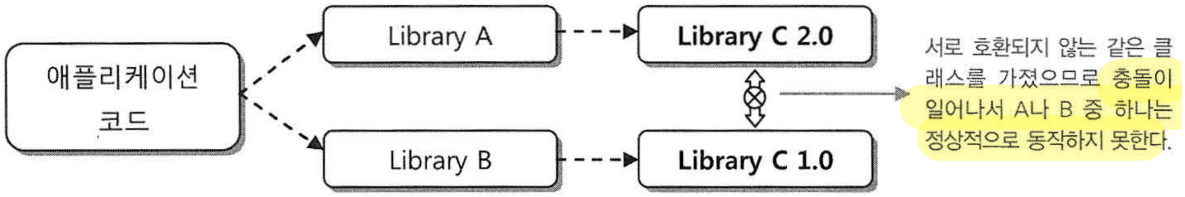

그림 9-11 버전이 다른 라이브러리의 문제

가장 간단한 해결방법은 재패키징이다.

자바엔 모듈이라는 개념이 없다. jar는 압축 패키징 방법일 뿐 구분 가능한 독립된 모듈이 아니다.

- 라이브러리 선정
  - 스프링 모듈

    사용할 기능과 기술 목록이 만들어졌으면 스프링 모듈부터 선정한다. 총 20개의 스프링 모듈이 있는데 일부는 공통적으로 사용되는 필수 모듈이고 그 외는 선택적으로 적용할 수 있다. 스프링 모듈의 의존관계를 참조해서 쓰면 된다.

  - 라이브러리

    오픈소스 라이브러리 또는 표준 API를 필요로 하기도 하고 상용제품의 라이브러리에 의존할 때도 있다. 때로는 각 라이브러리를 활용하는 방법에 따라서 다른 서드파티 라이브러리를 필요로 하는 경우가 있다.

- 빌드 툴과 라이브러리 관리

  Maven은 단순 빌드 툴을 넘어 개발 과정에서 필요한 빌드, 테스트, 배치, 문서화, 리포팅 등의 다양한 작업을 지원하는 종합 프로젝트 관리 툴의 성격을 띠고 있다. POM이라는 프로젝트 모델 정보를 이용한다 → 선언적이다. 전이적 의존 라이브러리 추적 기능이 있어 지정된 라이브러리에 필요한 여타 라이브러리까지 다운로드 해준다.

  선택 라이브러리는 전이적 의존 라이브러리 추적 기능의 적용을 받지 못한다. 참고는 할 수 있지만 명시적으로 POM에 선언해줘야 한다.

  조직이나 팀이 관여하는 프로젝트 사이에 변하지 않고 공통으로 사용하는 기술 목록을 만든다. 그 공통 기술을 적용할 때 필요한 스프링 모듈과 라이브러리를 선정한다. com.mycompany.common-deps같은 이름으로 라이브러리 목록만 담긴 간단한 POM파일을 만든다. 라이브러리 목록을 정의해놓은것이 목적이다.

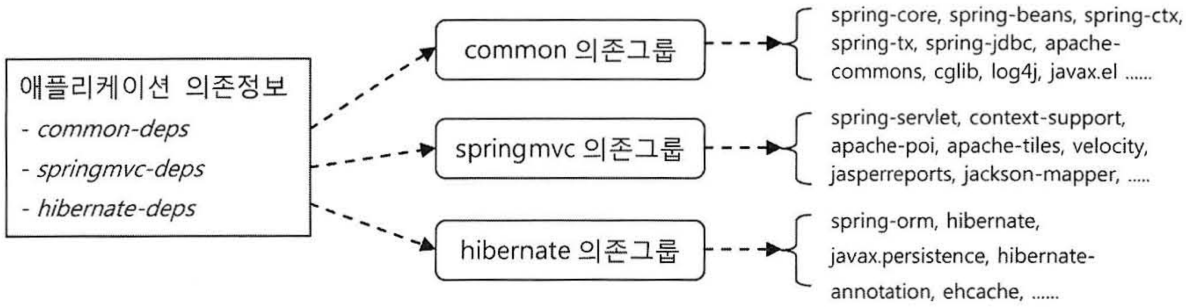

그림 9-12 의존 라이브러리 그룹의 활용

- 스프링 모듈의 두 가지 이름과 리포지토리

  Maven은 그룹 아이디와 아티팩트 아이디, 버젼 세가지로 라이브러리를 정의하는데 그 중 아티팩트 아이디와 버젼을 조합해서 파일 이름으로 사용한다.

  → spring-core-3.0.7.RELEASE.jar

  스프링의 모든 모듈은 OSGi호환 모듈로 만들어져있다. OSGi플랫폼에서 사용되지 않더라도 OSGi스타일의 모듈이름을 사용하도록 권장한다. OSGi 호환 이름을 갖는 스프링 모듈을 사용할 때는 Maven표준 리포지토리 대신 스프링소스가 제공하는 엔터프라이즈 번들 리포지토리를 사용해야한다.

  → org.springframework.core-3.0.7.RELEASE.jar