**Presentation Layer 프레임워크**

http://hyeonstorage.tistory.com/72



**1. Struts1**

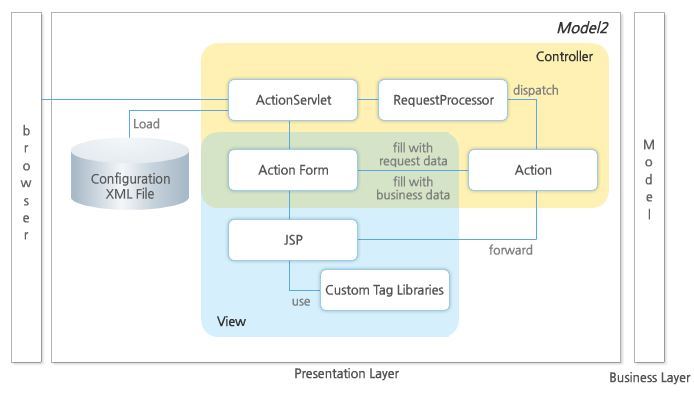
(1) 정의 : MVC 개념을 적용하여 대중적으로 사용된 Web 프레임워크

(2) 특징

- Command 패턴이 적용된 Action 클래스를 작성하여 사용자의 요청을 받아들임

- ActionForm을 비롯한 다양한 Form 클래스를 제공

- Tag Lib에 대한 지원과 Tiles 와의 연동이 가능하고 Message 처리에 대한 부분도 프레임워크 차원에서 지원해줌



**2. WebWork**

(1) 특징

- 간단한 Action Mapping

- 다양한 뷰 선택 : 액션이 실행되고 난 후 JSP, FreeMarker 등의 뷰 선택 가능

- 간단한 Parameter 처리(Value Stack)

> 액션 클래스에 setter 와 getter를 설정하면 메소드명과 일치하는 메소드를 호출해 값을 자동으로 셋팅해 줌

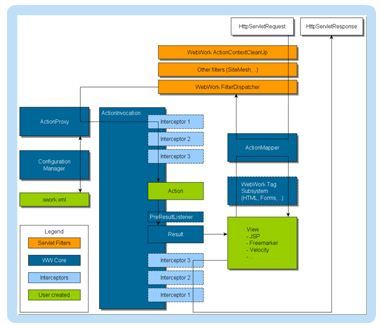
> 파라미터가 객체인 경우에 그 객체의 메소드도 접근 가능

> 뷰에서 ${user.name}이라고 사용하면 액션의 getUser().getName()을 호출

> params라는 기본적인 Intercepter 필요

- Thread Local의 사용

> Struts와의 비교 : 둘 다 Thread로 액션이 실행되는데 Struts의 경우는 같은 액션에 대해 멤버 필드가 공유되어 값의 안정성을 확신할 수 없는 반면 WebWork는 Thread Local을 사용하여 Thread 마다 별도의 메모리가 할당되어 같은 액션이라도 리퀘스트마다 별도로 실행됨.



**3. Struts2**

(1) 도입 배경 : Java 기반의 Web Application을 개발하는데 사용되는 프레임워크로 Struts1에 WebWork가 결합됨

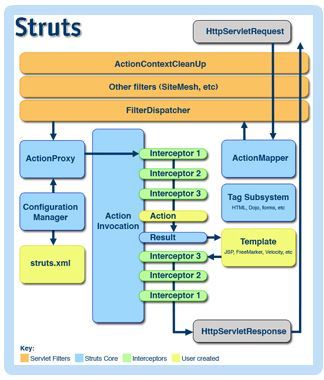
Struts1 + WebWork => Struts2

(2) 특징

- Action 클래스는 Struts2 프레임워크에 종속되지 않도록 디자인할 수 있고 쉽게 테스트가 가능

- 다양한 오픈 소스 프레임워크를 쉽게 사용할 수 있도록 Plugin을 제공하여 Spring과 같은 뛰어난 프레임워크를 간단하게 함께 사용 가능

- Filter와 Interceptor 기반의 MVC 아키텍처를 도입하여 매우 유연하고 설정이 간단함



(3) Struts2 의 달라진 점

- WebWork2 MVC 아키텍처 도입 : Struts 와는 전혀 다른 WebWork2의 아미텍처를 적용 함

- 인터셉터(Interceptor)

> 인터셉터는 HTTP Request를 중간에 가로채서 처리된다는 것은 Filter와 비슷한 개념이나 액션의 실행 전과 후에 처리 할 코드를 다르게 구성할 수 있음

> 인터셉터는 AOP 개념으로 Struts2 자체 내에 많은 인터셉터가 구현이 되어 있으며, 개발자가 필요에 따라 새로운 인터셉터를 등록하여 사용할 수도 있음.

> 복수의 인터셉트를 순서에 따라 실행도 가능

- 간단한 설정과 빠른 리로딩

> Struts2에서는 환경 설정 또한 WebWork2를 거의 그대로 적용하여 기본값으로 사용하여도 많은 설정을 생략할 수 있게 되어 있다.

>그리고, 설정도 extends와 include가 가능해 더욱 편리해졌으며, 변경 된 설정은 웹 컨테이너를 재시작하지 않고 빠르게 리로딩됨

- 프레임워크에 종속적이지 않은 액션 : Struts1과는 다르게 Struts2의 액션은 Struts2 프레임워크에 종속되지 않게 디자인하여 프레임워크를 사용하면서도 프레임워크에 종속적이지 않으며, 쉽게 테스트가 가능

 - 의존성 주입(Depencency Injection)의 지원 : 객체 간의 의존성 결합도를 낮추기 위해 구글 Guice 프레임워크를 내부적으로 사용하고 있으며, Spring과 같은 뛰어난 프레임워크를 함께 사용할 수 있음

 - Plugin 기능 : 유용한 많은 오픈 소스 프레임워크를 쉽게 사용할 수 있도록 Plugin을 제공함(JFreeChart, JaspaerReport, Sitemesh.. 등)

- 다양한 Result 타입 : Sturts2는 가장 많이 사용하는 JSP외에도 많은 뷰 기술(Freemark, Velocity 등)을 지원하는 Result 타입을 사용할 수 있으며, 개발자 스스로 만들어 사용도 가능

- 강력하고 많은 커스텀태그 지원 : 재사용성이 높은 많은 사용자 인터페이스 태그(커스텀태그)를 제공하며, 테마와 템플릿도 사용이 가능

**4. Spring MVC**

**"Spring MVC를 알기 위해선 Spring을 알아야 한다 ! "**

(1) 정의

- Spring은 AOP 개념과 IoC 개념을 바탕으로 사용자가 만든 "Business Object(POJO) 들을 엮어서 하나의 어플리케이션을 만들어 주는 컨테이너 프레임워크

- 사실상의 자바 엔터프라이즈 표준 프레임워크

(2) 등장 배경

- EJB를 대체할 만한 대안기술이 필요했다. 그것이 POJO 기반의 Spring 프레임워크이다.

(3) EJB란?

- 자바 엔터프라이즈 기술의 꽃

- EJB 스펙이 약속한 것 : 어플리케이션 개발을 쉽게 해줄 것이다.

어플리케이션 개발자는 로우레벨의 트랜잭션, 상태관리, 멀티쓰레딩, 리소스풀링 등의 복잡한 로우레벨 API등을 알 필요가 없다.

자바의 WORA(Write-Once, Run Anywhere) 철학을 따라 한번 개발되면 다양한 플랫폼에서 컴파일과 코드의 수정 없이 사용될 것이다.

- 장점

> 선언적 트랜잭션 관리 : CMT(Container-managed Transaction), EJB의 가장 성공적인 기술

> Remoting(분산 컴포넌트 가능)

> Thread Management/Instance Pooling

> Resource Pooling

> Security : 선언적인 방식에 의한 Role 기반의 보안설정

- 단점

> 복잡한 설계구조/아키텍처

> 과도한 기술(Over engineering)의 적용 : 성능저하, 학습비용증가

> 객체지향적인 설계와 구현의 장애

> 환경, 툴, 스펙에 종속적인 개발

> 복잡한 설정파일(Deployment Descriptor)

> 테스트의 어려움

(4) POJO란?

- POJO

> EJB의 불필요한 복잡도는 제거하고 장점은 유지

> 객체지향기술을 살린 설계와 구현이 가능

> 특정 환경, 서버에 종속되지 않는 기술

> 빠르고 손쉬운 테스트가 가능

- 특징

> 특정 규약(Contract)에 의존적이지 않음

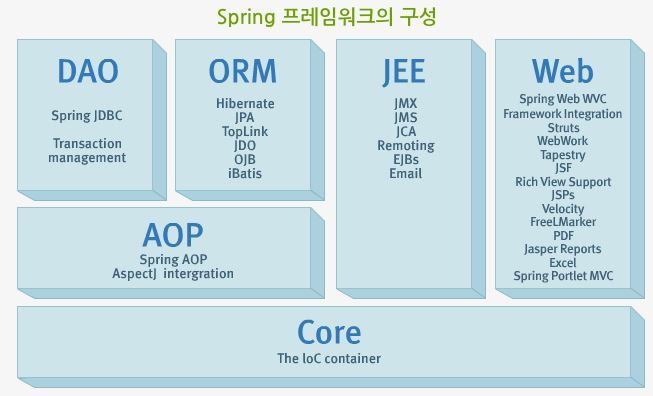
> 환경(Environment)에 의존적이지 않음

> 코드가 간결해 지고, 특정 환경에 의존적이지 않음

> 객체지향(OO) 디자인원칙에 집중할 수 있도록 해줌

> 생산성과 품질을 향상 시킬 수 있음

(5) Spring 프레임워크의 구성



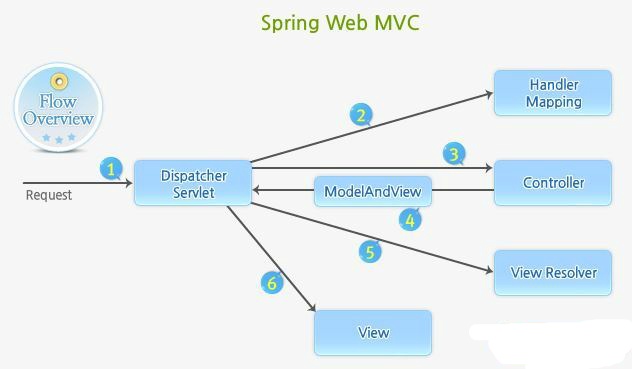
(6) Spring Web MVC 특징

- 모델2 아키텍처

- 역할의 분명한 분리

- 기존의 웹 어플리케이션 프레임워크와(Struts, Webwork, JSF 등) 통합도 지원

(7) Flow Overview



① : 클라이언트의 요청에 대한 최초 진입 지점은 DispatcherServlet이 담당한다. 대부분의 MVC를 지원하는 프레임워크가 메인 Servlet을 가지는 것처럼 Spring MVC 또한 메인 Servlet이 최초 진입지점으로 다음의 작업을 처리하게 된다.

② : DispatcherServlet은 Spring Bean Definition에 설정되어 있는 Handler Mapping 정보를 참조하여 해당 요청을 처리하기 위한 Controller를 찾는다.

③ : DispatcherServlet은 선택된 Controller를 호출하여 클라이언트가 요청한 작업을 처리한다.

④ : Controller는 Business Layer와의 통신을 통하여 원하는 작업을 처리한 다음 요청에 대한 성공유무에 따라 ModelAndView 인스턴스를 반환한다. ModelAndView 클래스에는 UI Layer에서 사용할 Model 데이터와 UI Layer로 사용할 View에 대한 정보가 포함된다.

⑤ : ④번에서 전달받은 ModelAndView 에 저장되어 있는 View 정보가 논리적인 View 이름일 경우에는 빈 설정 파일에 정의되어 있는 ViewResolver 클래스를 이용하여 클라이언트에게 출력할 View 객체를 얻게 된다.

⑥ : DispatcherServlet은 ViewResolver를 통하여 전달된 VIew에게 ModelAndView를 전달하여 마지막으로 클라이언트에게 원하는 UI를 제공할 수 있도록 한다. 마지막으로 클라이언트에게 UI를 제공할 책임은 View 클래스가 담당한다.