Framewrok

* 정해진 구조에 맞춰서 개발할 수 있는 틀
  1. 보편적인 틀이기 때문에 꼭 모든 것을 사용할 필요는 없다.
* 반제품을 입맛에 맞게 만드는 것
* Ex) Servlet
  1. HttpServlet -> Framework
  2. 내가 확장해서 만든 게 어떤 것이다 라는 정보를 web.xml에 기재해서 알려주어야 한다.

EJB

* 만들어진 거대한 배
* 분산처리를 위해서 사용

Spring

* 레고조각으로 만든 거대한 배
* core부분 정독
* IoC, AoP는 내부에서 사용되기 때문에 내용을 이해할 필요가 있다.

Maven

* Maven Project를 사용하면 설정에 따른 라이브러리를 가져오기 때문에 편리하다.
* Maven 중앙 Repository나 회사 내부의 Repository에 접근해서 가져온다.
  1. repo.maven.apache.org 에서 library들을 다운 받는다.
* 접근 Repository 설정이 가능하다.
* Group Id
  1. Pakage로 생각하면 된다.
* Artifact Id
  1. Project이름
* Web 프로젝트이기 때문에 war 형태로 Packaging
* Dynamic Web Project를 Maven으로 변경가능
* Maven Project Catalogs에서 원하는 타입으로 생성 할 수 있다.
  1. Create a simple project 를 선택하면 Architecture 설정 없이 원하는 대로 만들어서 사용 할 수 있다.
  2. pom.xml을 수정한 다음 프로젝트 > Maven > Update Project를 한다.
* pom.xml
  1. properties는 상수를 의미
  2. 상수를 사용할 때, ${spring-framework.version} 와 같이 사용한다.
  3. dependencies
     1. library를 추가하는 부분
  4. build
     1. build를 자동으로 실행해준다.
     2. build시 필요한 plugin들을 명시한다.
  5. repositories 태그를 넣어서 다양한 repository를 생성해서 사용 할 수 있다
  6. .m2 Directory에 library들을 다운받아서 관리
     1. C:\Users\PARK Wanbae\.m2

Ant

* 자동으로 build해주는 tool
  1. makefile을 자동으로 실행해주는 도구

Ant와 Maven의 차이점

* Ant는 자동으로 Project를 build해주는 도구이다.
* Maven은 Ant를 확장해서 Project 전체를 관리 해주면서 library관리 등 다양한 기능들이 추가 되었다.

Container

* 생성부터 실행 소멸까지 관리

IoC

* 제어의 역전
* 제어를 프로그래머가 하는 것이 아닌 프로그램에게 맡긴다.?.
* TV factory를 만드는 작업을 대신 해주는 것이다.
  1. User, TV (interface), LTV, STV, **TV factory**
  2. 객체를 만드는 작업을 대신 하는 것
  3. 내가 지어야 하는 공장을 대신 Spring이 대신 지어준다.
     1. Bean Factory
        + Bean 올려서 실행하는 건 Container
        + 단순한 객체만 생성
        + 거의 사용 안함
        + 사용되는 시점에 객체를 생성해서 전달해준다.
     2. ApplicationContext
        + 기능이 Bean Factory에 비해서 많기 때문에 주로 이용된다.
        + AOP동작 가능
        + 등록된 객체들을 미리 생성하고 사용할 때 전달 해준다.
        + 하지만 Bean에 설정을 통해서 사용되는 시점에 객체를 생성해달라고 설정 가능하다.
  4. jsp/servlet container는 굉장히 많은 파일로 되어 있는 것에 반해서 Spring에서는 Bean이나 ApplicationContext하나의 클래스(Container)에서 실행해주기 때문에 경량 Container라고 언급한다.
* 가져다 쓰는 방식 2가지
* EJB가 dependency lookup
  1. getBean, 내가 코드에 작성해서 받아오는 방식
* Spring이 dependency injection
  1. 자동으로 Spring Container가 주입시켜주는 방식
  2. Constructor
  3. setter (IoC가 사용하기 때문에 꼭 만들어 주어야 한다. 내가 사용하는 것이 아니기 때문에 약속을 지켜야 한다.(함수명))
  4. annotation (@Autowired)
     1. 모든 객체를 내가 작성하는 것이 아닌 라이브러리 즉, 미리 만들어진 객체를 사용하는 경우에는 따로 Bean에 등록하는 과정을 거쳐야 한다.
     2. ex) JDBC 객체 Datasource 는 이미 구현된 객체를 가져다 써야 하기 때문에 Bean등록을 따로 xml에서 실행해주어야 한다.
* Injection
  1. Bean 등록 -> Injection할 객체를 명시한다.
  2. Java 코드로 등록

IoC (Inversion of Control – 제어의 역전)

* 프로그램의 제어 흐름 구조가 바뀌는 것이다.
* 기존의 프로그램의 제어 흐름 구조
  1. 일반적으로 main()같은 프로그램이 시작되는 지점에서 사용할 오브젝트를 결정, 생성하고 오브젝트 내의 method를 호출하는 작업을 반복한다.
  2. 위와 같은 구조에서 각 오브젝트는 프로그램 흐름을 결정하거나 사용할 오브젝트를 구성하는 작업에 능동적으로 참여한다.
  3. 즉, 모든 종류의 작업을 사용하는 쪽에서 제어하는 구조이다.
* IoC에서 오브젝트는 자신이 사용할 오브젝트를 스스로 생성하거나 선택하지 않으며, 자신이 어떻게 만들어지고 어디서 사용되는지 알 수 없다. 즉, 모든 제어 권한을 자신이 아닌 다른 대상에게 위임하는 것이다.
* 작업을 수행하는 쪽에서 Object를 생성하는 제어 흐름의 개념을 거꾸로 뒤집는다.
* IoC에서는 Object가 자신이 사용할 Object를 생성하거나 선택하지 않는다.
* 또한 Object는 자신이 어떻게 생성되고 어떻게 사용되는지 알 수 없다.
* 모든 Object는 제어 권한을 위임 받는 특별한 Object에 의해서 만들어 지고 사용된다.

참고 URL: <http://isstory83.tistory.com/91>

IoC 구현 방법

* DL (Dependency Lookup) 의존성 검색
* DI (Dependency Injection) 의존성 주입
  1. Setter Injection
  2. Constructor Injection
  3. Annotation Injection

1.xml 설정파일에 직접 등록

* applicationContext.xml에 생성할 값들을 알려주어야 한다.
  1. 공장 생성하는 단계로 이해, 어떤 상품들이 있는지 등록

2. 자바코드로 빈 등록

* @Configuration으로 applicationContext.xml과 같은 설정 파일이라는 것을 알려준다.

Setter Injection

1. xml 설정파일에 직접 등록
   1. bean은 약속을 가지고 있는 자바 객체
   2. ejb bean -> ejb 규칙을 가지고 있는 자바 객체
   3. property -> setter 방식으로 입력하겠다는 의미
      1. value = “홍길동” 혹은 ref=”name” 과 같이 값을 넣어주는데, ref의 경우 ‘name’에 해당하는 id의 값을 넣어준다는 의미이다.
   4. constructor-arg -> Constructor 방식으로 입력하겠다는 의미
   5. 규칙
      1. 변수 -> private
      2. getter, setter -> public
      3. Default Constructor를 가지고 있어야 한다.
      4. Spring Container가 property name=”name” 에서 name에 해당하는 setName() 함수를 찾아서 값을 입력해준다.
   6. 왜 주입을 해서 사용하는지 생각해 보는 것이 중요하다.
   7. 주입하는 여러 가지 방법들
      1. setter
      2. Constructor
      3. Annotation (@Autowired)
   8. Maven Repository에 Library를 잘 받아오지 않는 경우
      1. C:\Users\PARK Wanbae\.m2\repository 에 있는 library들을 지우고 다시 Maven update를 통해서 다시 다운로드를 한다.

Spring MVC

* MVC model-1
  1. 각 각의 jsp로 접근이 가능하다.
* MVC model-2
  1. FrontController -> web.xml에 반드시 등록해야 한다.
     1. Dispatcher Servlet
     2. 보안상의 이유로 접근 통로를 1개로 제한하는 방법이다.
     3. 요청은 한 곳에서 받고 요청에 대한 정보를 확인 한 후에 각 각 맡은 처리 구간으로 요청을 보낸다.
     4. Spring이 제공하는 프레임
  2. Handler Mapping -> web.xml에 반드시 등록해야 한다.
     1. 요청된 URL이 실행될 곳을 알고 있어야 한다.
     2. Spring이 제공하는 프레임
  3. 항상 만든 것을 Spring에 설명해 주어야 한다.
     1. 과거에는 xml을 사용해서 설정 점 점 Class를 이용해서 설정한다.
  4. view resolver
     1. URL의 이름을 확인
     2. ex) list.jsp 인 경우 list의 값을 가져와서 정보들을 붙어셔 반환 해준다.
     3. 언제 view 엔진이 변경될지 모르기 때문이다.
  5. ModelAndView
     1. 가져온 정보들을 Model에 넣어주면 Spring이 정보를 Request로 넣어준다.

<https://javatpoint.com/spring-3-mvc-tutorial>

Handler Mapping을 따로 만들지 않고 Controller위에 @(annotation)을 이용해서 사용하기도 한다.

web.xml에서 filter

* 요청을 받을 때 filter를 두고 걸러 낼 수 있다.
  1. 위쪽: 요청이 들어올 때
  2. 아래쪽: 응답이 나갈 때
  3. /\*는 모든 요청에 대해 필터링
  4. ex) Character set을 Encoding으로 넣을 때 Filter를 이용해서 들어오는 요청을 UTF-8로 필터링해서 받아 올 수 있다.

web.xml에서 listener

* 특정한 event가 일어날 때 동작하도록 할 수 있다.
  1. Event자체만 등록시키고 URL이 필요가 없다.
  2. Context, request, session에 변경이 생길 때를 이야기 할 수 있다.
  3. ex) org.springframework.web.context.ContextLoaderListener는 WAS가 load되는 경우 실행되는 listener.
  4. listener가 실행되는 경우

웹 어플리케이션에서 사용되는 객체들: filter, listener

* WAS가 load되는 경우
* session
* request -> filter

WebInitializer

* web.xml에 등록할 내용을 java파일 Class에 등록한 예제

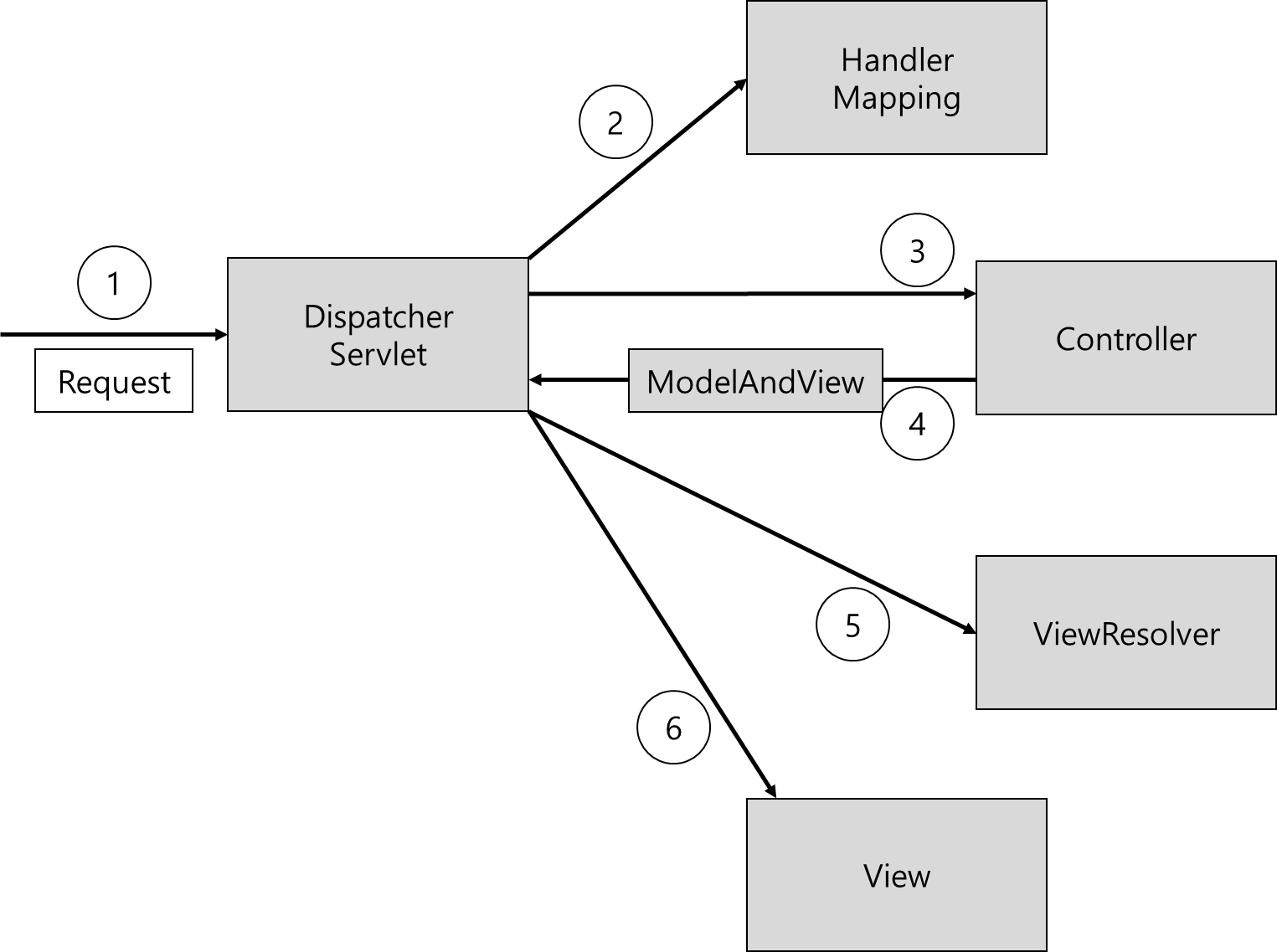
web.xml을 사용하지 않는다는 것을 의미하는 코드

<failOnMissingWebXml>false</failOnMissingWebXml>

<http://140616ing.tistory.com/46>

Spring Web MVC

* Spring Web MVC 패턴은…



전체 흐름은 View의 역할을 하는 객체에게 forwording을 하는 것이다.

1. 사용자에게 URL값으로 전달된 Request를 Dispatcher Servlet(Front Controller)로 모두 받아온다.
2. Handler Mapping에는 각 URL값마다 mapping된 Controller정보를 저장해 두는데 해당 정보를 통해서 Request는 Controller를 찾아가서 실행한다. 이때, Spring에서 사용자는 @(Annotation)을 통해서 Mapping정보를 간편하게 설정할 수 있다.
3. Handler Mapping정보를 가지고 mapping되는 Controller가 작동한다.
4. Controller에서는 Model 혹은 View정보를 전달하는데, 인자로 Model을 받아서 Model에 정보를 추가하거나 View 이름 정보를 전달 할 수 있다.
5. (JstlView.Class)ViewResolver는 Controller로부터 전달된 string이나 model에 대해서 URL을 완성시켜주는 기능을 수행한다.
   1. prefix = “/WEB-INF/jsp”
   2. suffix = “.jsp”
   3. prefix + ModelAndView + suffix
   4. 위의 경로는 jsp파일이 위치하는 경로인데 WEB-INF 하위디렉토리에 jsp파일을 위치시키면 Client가 직접적으로 접근 할 수 없기 때문에 보안상 중요하다.



1. Dispatcher Servlet은 request에 의해서 완성된 경로 (prefix + ModelAndView + suffix)에 해당하는 파일로 Client에게 응답한다.

JSP & JSTL

* Spring은 JSP와 JSTL 뷰에 대한 다수의 해결책을 제공한다. WebApplicationContext에 정의한 일반적인 ViewResolver로 JSP나 JSTL을 사용한다.
* 하지만 실제로 뷰를 랜더링할 JSP를 작성해야 한다.

ViewResolver

* Spring에서 사용되는 다른 뷰와 마찬가지로 JSP에서도 뷰를 처리할 ViewResolver가 필요하다. JSP View를 처리할 때 가장 많이 사용되는 ViewResolver는 InternalResourceViewResolver와 ResourceBundleViewResolver이다. 2가지 ViewResolver는 모두 WebApplicationContext에 선언되어 있다.

WAS는 web.xml을 읽어와서 동작하기 때문에 web.xml에 오류가 생기면 서버 실행이 되지 않을 수 있다.

VO객체 -> 도메인?

1. IoC

- Bean Factory

- ApplicationContext

ⅰDependency Lookup

ⅱDependency Injection

- Setter Injection

- Constructor Injection

- Annotation Injection

2. Spring Web MVC