1. 시작하기 전에..

- 무엇을 스터디하나요?

- 무엇을 얻어갈까요?

2. 개별 수준 점검

- 자기 자신에게 어떤 점검을 위해서라도 종종 질문을 던져보는 건 자신을 일꺠우는 좋은 방법인 것 같아요.

\* 스스로 하기 힘드니까 제가 도와 드릴꼐요.

- 왜 하나요?

하나. 스터디에서 어느 정도의 내용을 다루어야할지 파악해보기 위해서

둘. 알맞은 조를 짜기 위해

2.1 설문

질문 하나. Spring을 이용한 실무 경험이 있나요?

질문 둘. 스스로 Spring 개발 환경을 구축해본 적이 있나요?

질문 셋. Feature List 도출/산정 작업을 주도적으로 해본 경험이 있나요?

실무에 대한 경험이 있다면 Spring MVC의 간단한 사용방법은 모두 알고 있을 것입니다.

스스로 Spring 개발 환경을 구축해보았다면 상황에 맞추어 필요로 하는 라이브러리나 Spring 모듈을 추가할 수 있을 것입니다.

Feature List 도출 / 산정 작업은 사실 모두가 주도적이어야 한다고 생각해요. 자기 의견을 당당히 말할 수 있어야 하고요.

외국 사례들을 살펴보면 실제로 의견을 맞추기 위해서 Planning Poker를 하기도 하자나요?

본인의 생각없이 남의 의견을 수렴만 해보았다면 이번 기회에 본인의 의견을 내는 연습을 해봐요.

3. 개발 환경 구성

개발 환경을 모두 동일하게 셋팅하는게 좋아요.

서로 다른 환경에서 개발을 한다면 아무리 Platform 독립적인 Java여도 힘들어요.

뿐만 아니라, Spring 라이브러리를 하나씩 가져와서 셋팅해보는 작업도 무시할 수 없는 작업이므로 미리 경험을 해본다면 나쁘지 않죠.

먼저, 개발을 하는데 가장 필요한 JDK와 IDE를 설치해볼 것입니다.

Spring Tool Suite ( 이하 STS )를 설치하기 위해서는 JDK를 필요로 해요. 그래서 JDK를 먼저 설치해주어야 합니다.

JDK : JAVA 1.6.37 <http://www.oracle.com/technetwork/java/javase/downloads/jdk6u37-downloads-1859587.html>

다음은 IDE의 설치를 진행하도록 할게요.

IDE : STS 2.9.2 <http://www.springsource.org/downloads/sts-ggts>

STS의 버전을 고를 떄는 고려할 점이 여러가지 있는데요. 2.9.2 이상의 버전을 사용하게 되면 이클립스 JUNO (4.2.1)을 기반으로 하기 때문에

많은 메모리를 차지하게 됩니다. 안그래도 느린 개발환경을 더욱 느리게 하게 됨으로 사양이 높지 않다면 2.9.2 버전을 추천드립니다.

(그 이상을 버전을 깔더라도 특별히 다른 부분은 없습니다.)

STS에는 Maven이 포함되어 있어서 별도로 Maven을 설치하는 작업이 필요하지 않아요.

또한 Spring을 이용한 개발을 할 때 유용하게 쓸 수 있는 여러 플러그인들을 기본 장착하고 나와서 개발에도 용이하죠.

(기본 Eclipse에 Plugin을 따로 설치하는 방법도 있어요.)

소스에 대한 형상 관리를 위해서 git을 사용해볼께요.

<http://rogerdudler.github.com/git-guide/index.ko.html>

GIT을 사용하기 위한 기본적인 환경은 STS를 설치하면서 함께 설치가 되요. 그렇기 때문에 따로 GIT을 사용하기 위한 설치는 필요치 않습니다.

함께 사용할 git repository url은

[https://github.com/spring-board-study](https://github.com/JeongInnLee/spring-board-study) 입니다.

여기서 나누어진 조에 따라서 group1, 2, 3, 4로 나누어서 사용하시면 됩니다.

그럼, 간단한 사용방법을 알아보겠습니다.

Windows -> show view를 선택해서 Git Repositories를 눌러줍니다.

Git Repositories가 추가된 것을 볼 수 있습니다.

여기서 우리는 이미 만들어진 remote git repository 를 사용할 것이기 때문에 "Clone a Git repository"를 선택해줍니다.

그럼 리파지토리를 추가하기 위해 다음과 같은 입력창이 생기게 됩니다.

만약에 그룹1이라고 한다면 다음과 같은 주소를 넣으면 됩니다.

https://github.com/spring-board-study/group1.git

(이 url 부분에는 git의 주소를 넣어주어야 하는데요. 타이핑을 하게 되면 힘들 수 있으므로 사이트에 접근을 해서 주소를 따오도록 합니다.)

주소만 넣어주면 아래의 내용은 자동으로 채워지게 되어 있으며 Authentication의 경우 공개 repository로 설정이 되어 있기 때문에

별도의 설정이 필요 없습니다.

설정을 끝맞추고 Finish를 누르게 되면 Repositories 에 추가 된것을 볼 수 있습니다.

이제 형상관리를 위한 준비는 끝이 났습니다.

다음으로 template 프로젝트를 생성해서 Hello Spring을 찍고 이를 git에 올려보도록 하겠습니다.

4. Getting Started -  Hello Spring

- 프로젝트 생성

Spring 프로젝트를  생성하는 방법은 여러 방법이 있습니다.

대표적으로는 STS에서 지원하는 Spring Template Project를 사용해도 되고, Maven Project로 생성하여 한땀 한땀 정성들여서 만들 수도 있습니다.

그리고, maven을 사용하지 않는 환경이라고 한다면 일반적인 dynamic 프로젝트로 생성하여 사용할 수도 있습니다.

본 스터디에서는 경험하고 공부하는 것에 주 목적을 두기 위해서 maven project로 생성하여 하나씩 설정해보도록 하겠습니다.

마지막으로 finish를 선택하게 되면 굉장히 심플한 프로젝트가 생성된 것을 볼 수 있습니다.

pom.xml도 굉장히 단순한 설정만을 가지고 있구요.

여기서 그럼 우리가 필요로 하는 설정들을 하나하나 추가해볼까요?

- pom.xml 설정

가장 먼저 properties를 설정하도록 하겠습니다.

우리가 사용하려고 하는 Spring의 버전, 그리고 Java의 버전과 같은 정보들을 Properties로 관리하게 되면

서로 간에 실수를 줄일 수 있게 되고 버전 정보에 대한 관리가 쉬워집니다.

<properties>

<java-version>1.6</java-version>

<org.springframework-version>3.1.0.RELEASE</org.springframework-version>

</properties>

그리고, 우리는 Junit 3가 아닌 jUnit4에 익숙함으로 junit 버전 또한 4.7로 변경해주도록 합니다.

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.7</version>

<scope>test</scope>

</dependency>

다음은 Spring을 사용하기 위한 모듈들을 등록해보도록 하겠습니다.

근데, 가장 먼저 드는 의문은 web 개발을 하기 위해서 Spring의 어떤 모듈들이 필요로 할까요?

그리고, 그 모듈들이 저장되어 있는 리파지토리는 어디일까요?

해답은 역시 <http://www.springsource.org/spring-framework> 에 있습니다.

**RESOLVING SPRING FRAMEWORK ARTIFACTS VIA MAVEN**

<repository>

<id>springsource-repo</id>

<name>SpringSource Repository</name>

<url>http://repo.springsource.org/release</url>

</repository>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.1.3.RELEASE</version>

</dependency>

메이븐을 통해서 Spring을 사용하고자 한다면 다음과 같이 설정하라는 것이겠죠?

일단 저것만 추가해볼까요?

우리는 Spring-context라는 모듈만을 추가했지만 실제로는 Spring-context와 연관성을 가지고 있는 클래스들이 또 있기 떄문에 다음과 같이 다양한 module들이 dependency가 걸리게 됩니다.

여기까지 하고 프로젝트에 구현을 추가해가며 그 때 그 떄 필요한 모듈들을 추가해보도록 하겠습니다.

- Hello Spring 구현

스프링을 통한 프로젝트를 이미 경험해보신 분들은 알고 계시겠지만

스프링 웹 어플리케이션을 만들기 위한 시작은

의존성 주입이 이루어질 수 있도록 톰캣이 시작하면서 spring container를 올리게 하는 것입니다.

그 설정은 다음과 같습니다.

<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/root-context.xml</param-value>

</context-param>

<!-- Creates the Spring Container shared by all Servlets and Filters -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

contenxtConfigLocation은 Spring의 설정 파일이 어디에 위치하고 있는지를 나타내고 그 값은

Spring의 ContextLoaderListener에 의해서 사용되어 spring container를 로딩하게 됩니다.

이렇게 되면 사실상 Spring Container에 대한 등록은 끝난 것입니다.

빌드를 해볼까요?

board.war파일이 생성된 것을 보아 빌드가 성공적으로 이루어졌다는 것을 확인할 수 있습니다.

하지만, 무엇인가 모자르죠??

사내에서 사용하는 Spring template Project의 pom.xml을 살펴보게 되면 다음과 내용을 볼 수 있습니다.

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<configuration>

<webappDirectory>${deploy-path}</webappDirectory>

</configuration>

</plugin>

이것 플러그인은 war 빌드를 할 때 그에 대한 war deploy 경로를 지정해줄 수 있게 해줍니다.

이렇게 빌드를 하고 난 classes 폴더와 lib 폴더가 생성된 것을 볼 수 있지요.

이제 톰캣을 설정하여 올려보도록 하겠습니다.

(톰캣 설정은 단순하므로 설명없음)

서버를 띄우니 이제

java.lang.ClassNotFoundException: org.springframework.web.context.ContextLoaderListener

이런 오류가 발생합니다.

spring-context의 경우 우리가 spring container를 사용하기 위한 기본적인 것들은 포함이 되어 있지만 Spring-web을 사용하기 위한

모듈들이 포함되어 있지 않기 때문입니다.

그럼, spring-web을 구현하기 위한 모듈을 dependencies에 추가해보도록 하겠습니다.

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework-version}</version>

</dependency>

다시 서버를 띄워보면 ! 성공을 하는군요.

이제 Hello Spring을 출력하기 위해서 root-context.xml 파일을 설정해볼까?

root-context.xml에 다음과 같이 추가합니다.

<context:component-scan base-package="com.nhncorp.study" />

com.nhncorp.study 패키지 아래에 있는 모든 클래스에 대해서 @Component 어노테이션을 포함한 모든 하위 어노테이션을 스캔하여 빈으로 등록하도록 합니다.

그리고, com.nhncorp.study.board.service에 다음과 같이 클래스를 정의합니다.

@Service

public class HelloSpringService implements InitializingBean {

public void afterPropertiesSet() throws Exception {

System.out.println("Hello Spring");

}

}

여기까지하면 우리는 Hello Spring을 확인할 수 있습니다.

- 프로젝트 git에 commit(push)

git에서의 commit은 remote repository 에서의 merge가 아닙니다.

git에서의 commit은 local repository에서의 소스 코드 merge를 나타내며 그 commit들은 모아져서 push 명령어를 통해서 remote repository에 반영되게 됩니다.

Team -> Share Project를 선택하게 되면 다음과 같은 창이 나타납니다.

여기서 우리는 git을 이용하기 때문에 Git을 선택하구요.

다음과 같이 repository를 설정해주고 board 프로젝트를 선택하면 project가 share 됩니다.

그 결과는 다음과 같이 Git Repositories를 조회하면 확인할 수 있습니다.

이를 remote repository에 적용하기 위해서는 앞서 말했듯이 commit - push를 수행하면 됩니다.

Q/A 및 실습