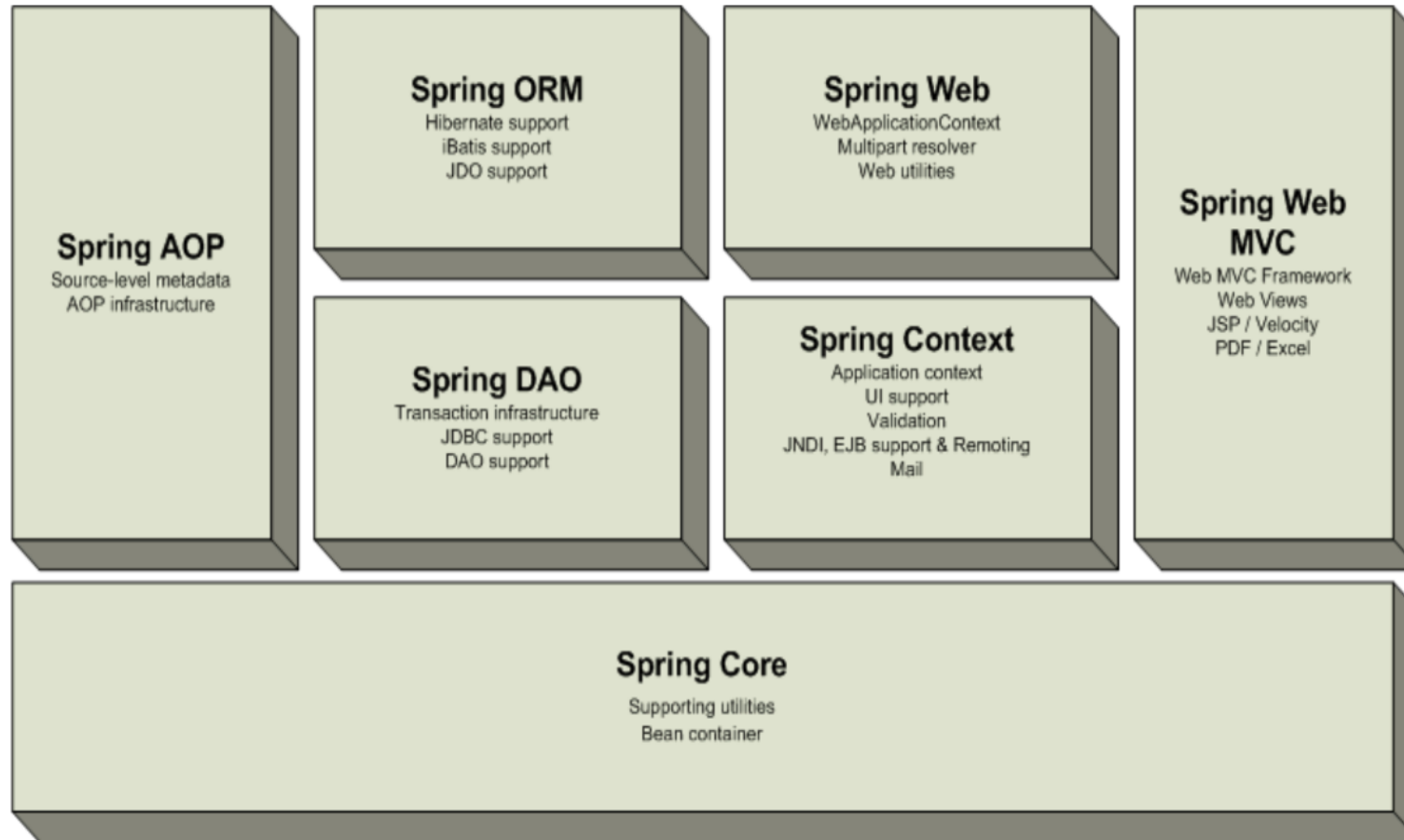


SPRING 기초

Spring Framework의 주요 기능 및 특징

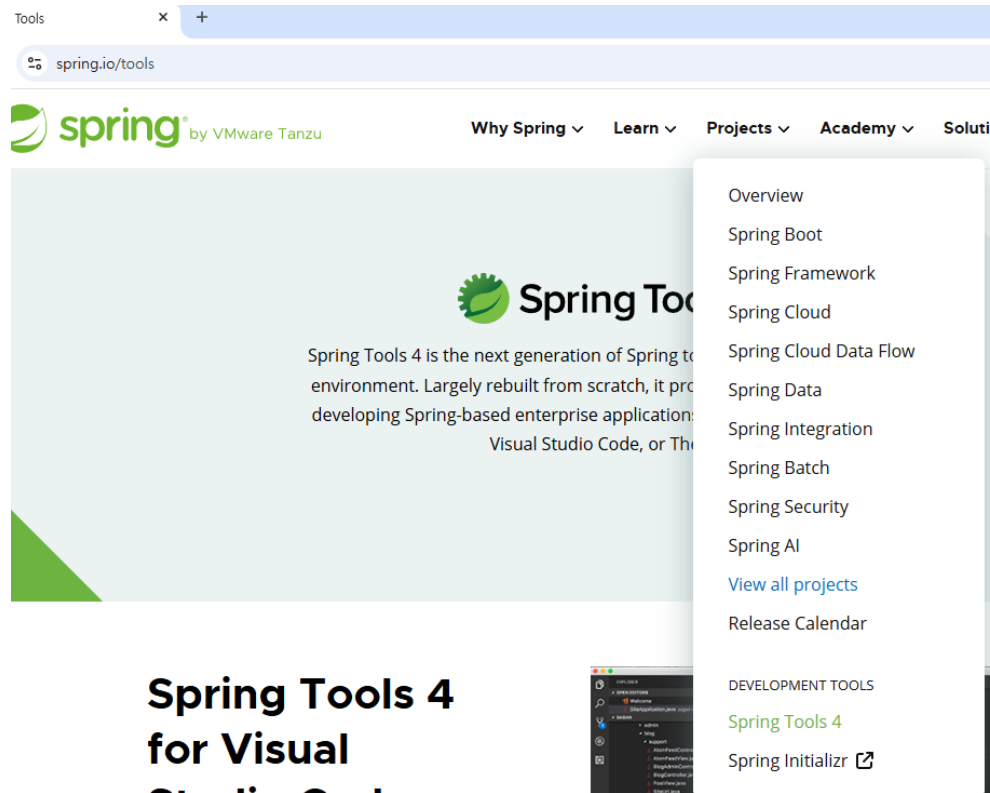
1. 스프링은 경량의 빈 **컨테이너(Factory)**다.
 - 스프링 컨테이너는 자바 객체의 생성, 소멸과 같은 라이프 사이클을 관리하며, 스프링 컨테이너로부터 필요한 객체를 검색하여 사용할 수 있다.
2. 스프링은 **DI(Dependency Injection)** 기능을 지원한다.
 - XML 설정 파일을 통해 객체간의 의존관계를 설정할 수 있다.
 - 객체는 의존하고 있는 객체를 직접 생성하거나 검색할 필요가 없다.
3. 스프링은 **AOP(Aspect Oriented Programming)**을 지원한다.
 - 로깅, 보안, 트랜잭션과 같은 공통 기능을 핵심 비즈니스 모듈로부터 분리해서 각 핵심 비즈니스 모듈에 적용할 수 있다.
4. 스프링은 **POJO(Plain Old Java Object)**를 지원한다.
 - 스프링 컨테이너에 저장되는 자바 객체는 특정한 인터페이스를 구현하거나 클래스를 상속받지 않아도 된다.
 - 따라서 기존에 작성한 클래스를 수정할 필요 없이 스프링에서 사용할 수 있다.
5. 스프링은 **트랜잭션 처리를 위한 일관된 방법을 제공한다.**
 - JDBC를 사용하든, 컨테이너가 제공하는 트랜잭션을 사용하든 XML 설정 파일을 통해 트랜잭션 코드에 상관없이 일관되게 트랜잭션을 제어할 수 있다.
6. 스프링은 **영속성과 관련된 다양한 API**를 지원한다.
 - JDBC를 비롯하여 iBatis, Hibernate, JPA 등 데이터베이스 처리를 위해 사용되는 라이브러리와의 연동을 지원한다.

Spring Framework 주요 구성 모듈



프로그램 다운로드

- 메뉴의 **projects-spring tools4**
- **spring tool suite3 wiki**를 클릭하여 legacy버전을 선택
- 링크:<https://github.com/spring-attic/toolsuite-distribution/wiki/Spring-Tool-Suite-3>



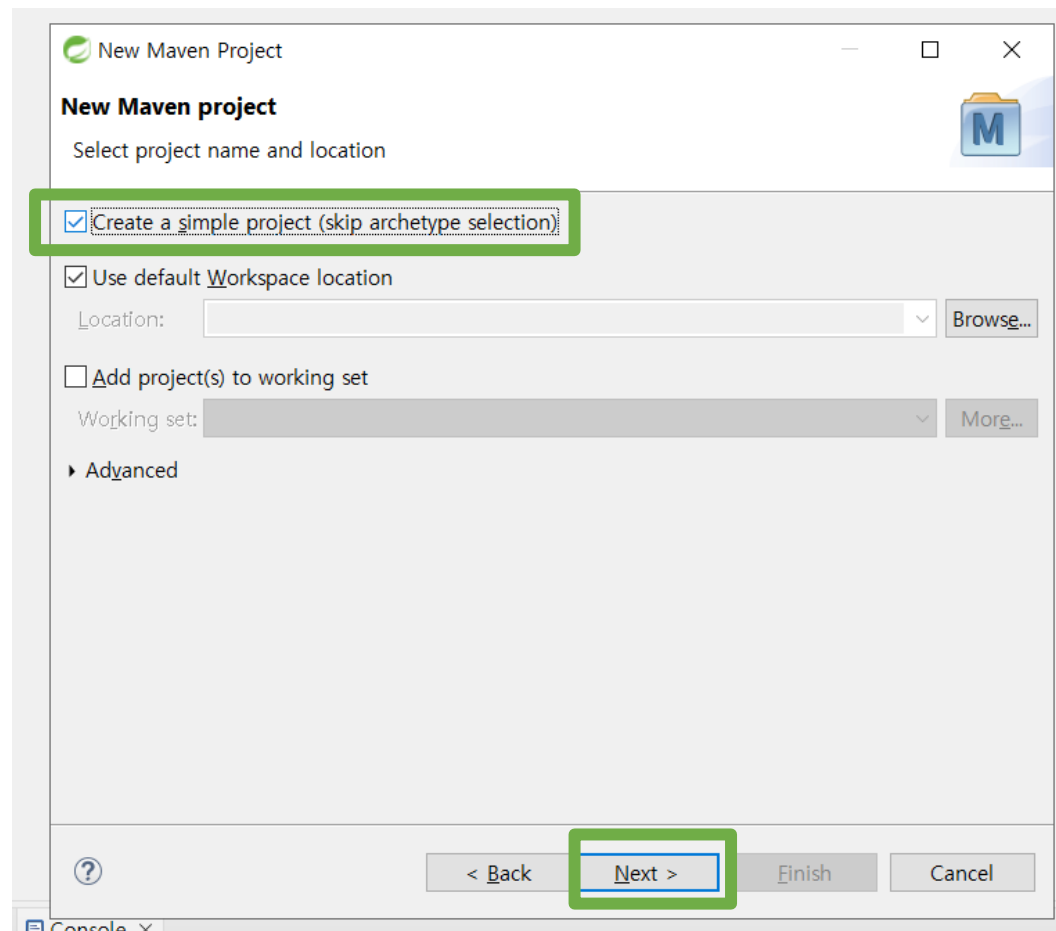
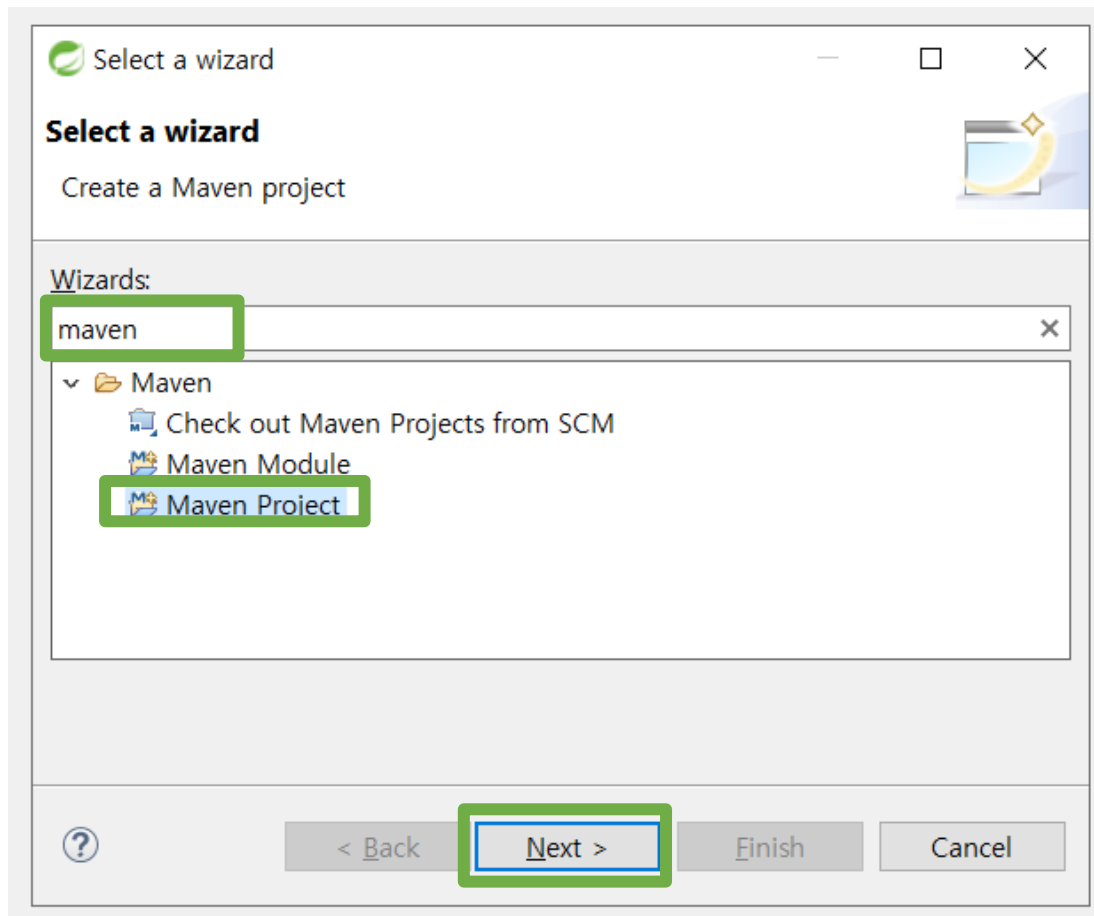
Looking for Spring Tool Suite 3?

Version 3 of the Spring Tool Suite is no longer under active development and does not receive any maintenance updates anymore. The last and final release can be found on the [Spring Tool Suite 3 wiki](#), alongside details of how to upgrade to **Spring Tools 4**.

MAVEN의 이해

- 컴파일 도구, 라이브러리 관리
- 자세한 사항은 별도 파일 제공


메이븐 프로젝트 생성



New Maven Project

New Maven project

Configure project



Artifact

Group Id:com.my

Artifact Id:springbasic

Version:0.0.1-SNAPSHOT

Packaging:jar

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

Browse...

Clear

Advanced

?

< Back

Next >

Finish

Cancel

Build path 설정을 통해 자바 선택

Package Explorer ×

basic

src/main/java

Console Progress Problems ×

2 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
The project cannot be built until build path errors are resolved	basic		Unknown	Java Problem
Unbound classpath container: 'JRE System Library [CDC-1.0/Foundation-1.0]' in project 'basic'	basic		Build path	Build Path Pr...

Build path entry is missing: org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardV

Source Projects Libraries Order and Export Module Dependencies

JARs and class folders on the build path:

- JRE System Library [CDC-1.0/Foundation-1.0] (unbound)
- Maven Dependencies

Edit Library

JRE System Library

Select JRE for the project build path.

System library

Execution environment: CDC-1.0/Foundation-1.0 (unbound)

Environments...

Alternate JRE:

Installed JREs...

Workspace default JRE (jdk-23.0.1)


```
package d250107;
```

```
public class Main1 {
```

```
    public static void main(String[] args) {
```

```
        //학생은 반드시 아이디, 이름, 성적을 가진다 라고 정의할 경우
```

```
        //학생과 성적 중 최종 사용하고자 하는 객체는 무엇인지 파악
```

```
        //학생과 성적중 누구를 먼저 생성할 것인가?성적
```

```
        Sungjuk sungjuk=new Sungjuk(100, 90, 80);
```

```
        Student student=new Student(1, "hongkildong", sungjuk);
```

```
        System.out.println(student);
```

```
        Sungjuk sungjuk2=new Sungjuk(80, 90, 100);
```

```
        Student student2=new Student(1, "kimminsu");
```

```
        student2.setSungjuk(sungjuk2);
```

```
        System.out.println(student2);
```

```
    }
```

```
}
```

```
/*
 * 학생은 학번, 이름, 성적을 가진다.
 * 성적은 국어, 영어, 수학 점수를 갖는다.
 */
```

```
class Student{
    private int id;
    private String name;
    private Sungjuk sungjuk;
```

```
    public Student() {
        //기본생성자에서 선언해서 값을 주입하는 방법
    }
```

```
    public Student(int id, String name, Sungjuk sungjuk) {
        this.id = id;
        this.name = name;
        this.sungjuk = sungjuk;
    }
```

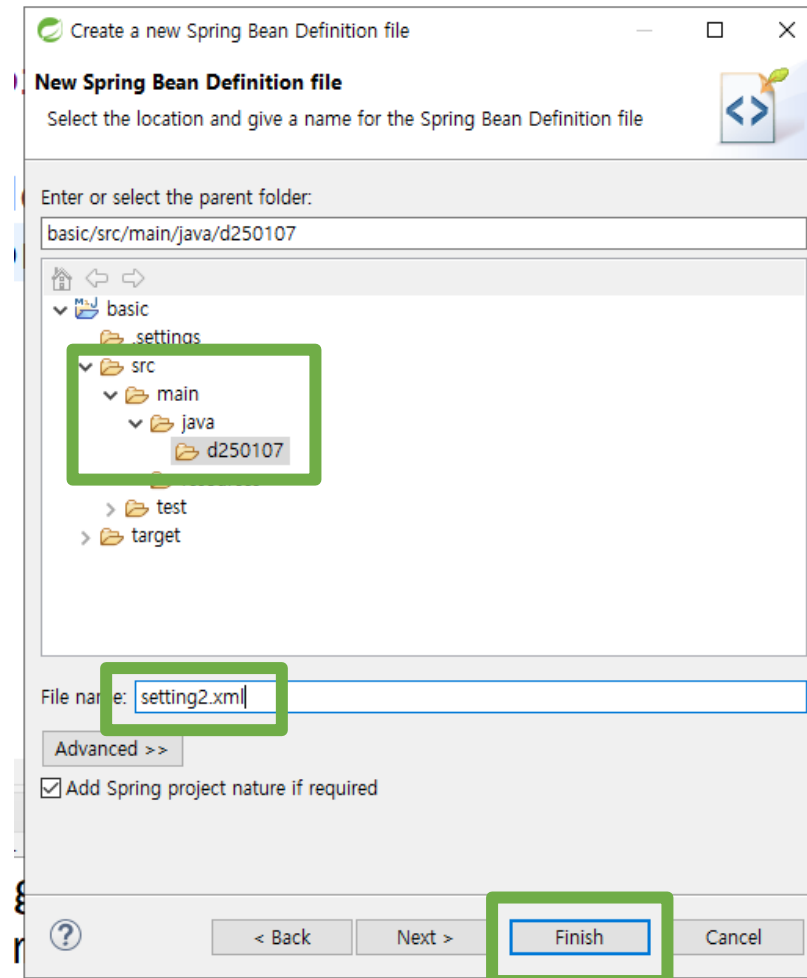
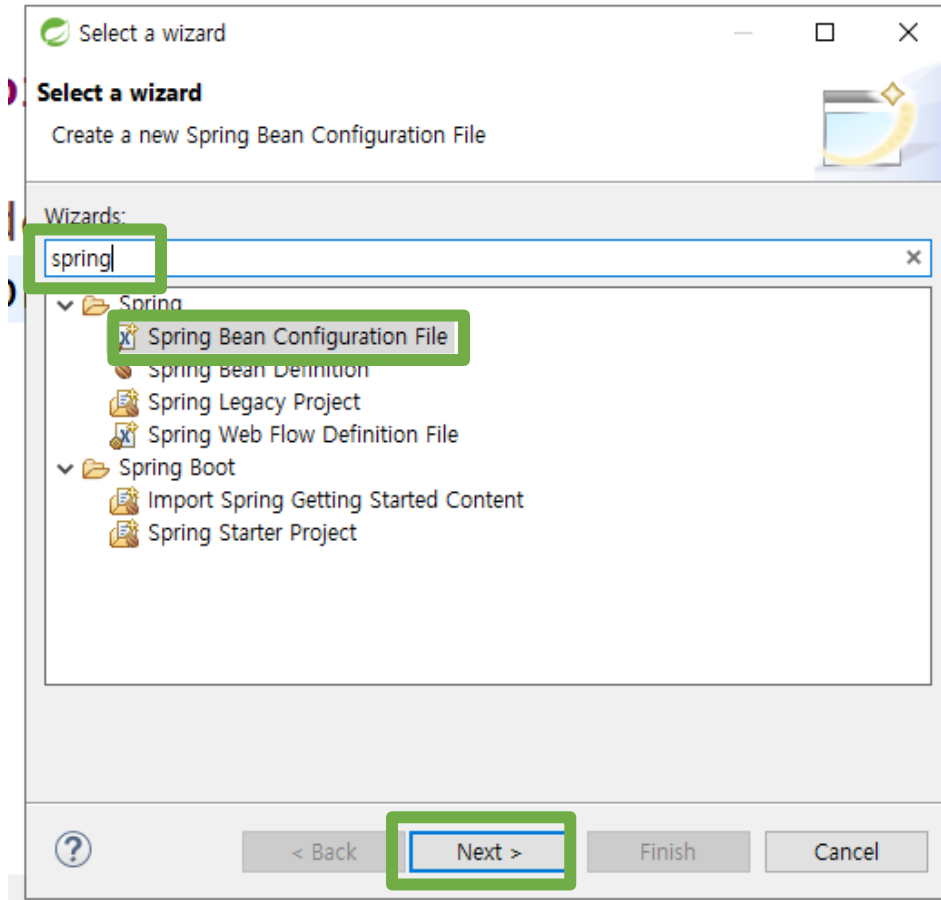
```
    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }
    public int getId() {
        return id;
    }
```

```
    public void setId(int id) {
        this.id = id;
    }
```

```
    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }
}
```

스프링 설정파일 만들기



라이브러리 Pom.xml 추가

```
<!--  
https://mvnrepository.com/artifact/org.springframework/spring-  
context -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-context</artifactId>  
    <version>5.3.39</version>  
</dependency>
```

빈 객체 생성

```
<bean id="sj" class="d250107.Sungjuk">  
<constructor-arg value="50"/>  
<constructor-arg value="70"/>  
<constructor-arg value="80"/>  
</bean>
```

Id는 변수명, class는 패키지포함 경로
Constructor-arg는 생성자 값전달
Property의 name은 객체 변수명동일(실제는 set함수)

```
<bean id="student" class="d250107.Student">  
<property name="id" value="1001"/>  
<property name="name" value="홍길동"/>  
<property name="sungjuk" ref="sj"/>  
</bean>
```

Spring 설정파일로 부터 객체 전달받기

```
public class Main2 {  
  
    public static void main(String[] args) {  
        //경로는 패키지부터시작하여 /로 구분후 설정  
        ApplicationContext app=  
            new ClassPathXmlApplicationContext  
                ("d250107/setting2.xml");  
  
        Student student=(Student)app.getBean("student");  
        System.out.println(student);  
  
    }  
  
}
```

인터페이스를 활용한 빈설정

- Print인터페이스 생성
- ConsolePrint생성(PersonInfo 멤버변수)
- GridPrint생성(PersonInfo 멤버변수)
- PersonInfo생성(id, name, address)
- 빈설정파일 생성(setting.xml)

```
interface Print {  
    public void print();  
}
```

```
public class GridPrint implements Print{
    PersonInfo info;

    public GridPrint() {}
    public GridPrint(PersonInfo info) {this.info = info;}
    public PersonInfo getInfo() {return info;}
    public void setInfo(PersonInfo info) {this.info = info;}

    public void print() {
        System.out.println("+-----+");
        System.out.println("| 주민번호 | 이름 | 주소 |");
        System.out.println("+-----+");
        System.out.println ("| "+info.getId()+" | "+info.getName()+" | "+info.getAddress()+" |");
        System.out.println("+-----+");
    }

}
```



```
public class ConsolePrint implements Print{

    PersonInfo info;

    public ConsolePrint() {}
    public ConsolePrint(PersonInfo info) {this.info = info;}
        public PersonInfo getInfo() {return info;}
    public void setInfo(PersonInfo info) {this.info = info;}

    public void print() {
        System.out.println("주민번호:"+info.getId());
        System.out.println("이름:"+info.getName());
        System.out.println("주소:"+info.getAddress());

    }

}
```

```
public class PersonInfo {
```

```
    String id;
```

```
    String name;
```

```
    String address;
```

```
    public PersonInfo() {}
```

```
    public PersonInfo(String id, String name,  
        String address) {
```

```
        this.id = id;
```

```
        this.name = name;
```

```
        this.address = address;
```

```
    }
```

```
}
```

```
public static void main(String[] args) throws InterruptedException {  
    ApplicationContext app=  
    new ClassPathXmlApplicationContext  
    ("d250107_1/setting.xml");  
  
    ConsolePrint p=(ConsolePrint) app.getBean("consolePrint");  
    GridPrint p=(GridPrint) app.getBean("gridPrint");  
    Print p=(Print) app.getBean("consolePrint");  
}
```

```
public static void main(String[] args) {  
    ApplicationContext app  
    =new ClassPathXmlApplicationContext("d0616_02/setting.xml");  
    Print print=(Print)app.getBean("print");  
    print.print();  
}
```

예제

- 학생객체는 아이디, 이름, 성적객체, 프린터객체가 존재하도록 설정
- 학생 정보를 보기 확인하기 위한 gridPrint와 consolePrint를 작성하시오.
- 학생객체, 성적객체, GridPrint, ConsolePrint 객체 생성하여 xml 파일로 작성



setting.xml



Sungjuk.java



Student.java



Print.java



PrintInfo.java



Main.java



GridPrint.java



ConsolePrint.java

```
#config.properties  
objectname1=gridPrint  
objectname2=consolePrint
```

```
public class Main {  
  
    public static void main(String[] args) {  
        Properties pro=new Properties();  
  
        try(FileInputStream in=new FileInputStream  
            ("D:/app  
dev/stsworkspcace/basic/src/main/java/d250107_2/config.properties")){  
            pro.load(in);  
            System.out.println(pro.getProperty("objectname1"));  
            System.out.println(pro.getProperty("objectname2"));  
            /*  
            이 코드 완성후 d250107_1.Main클래스의 main함수로 이동하여 반복문처리  
            config.properties파일의 값을 변경하여 코드는 변경하지 않고  
            설정값 변경만으로 동작이 변경되는지 확인  
            */  
        }catch(Exception e) {e.printStackTrace();}  
    }  
}
```

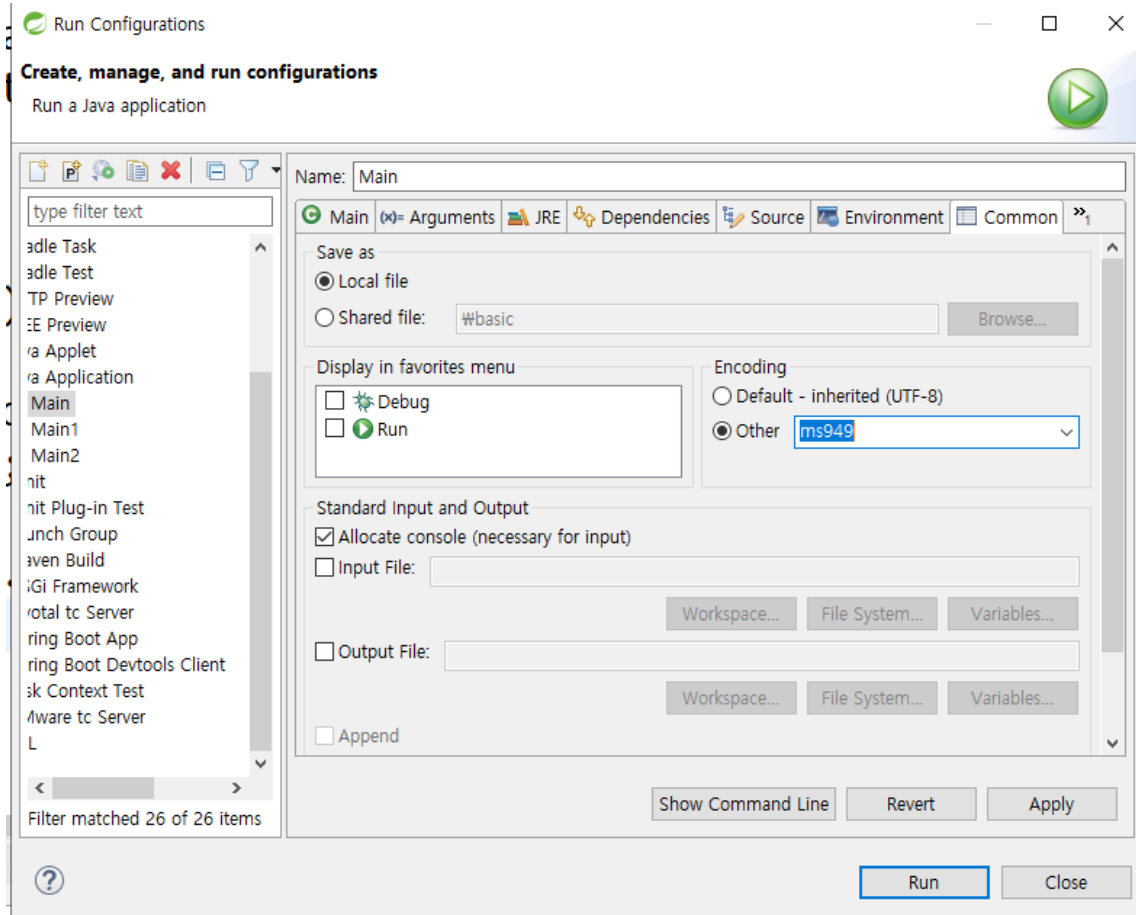
```
public class Main{

    public static void main(String[] args) throws InterruptedException {
        ApplicationContext app=new ClassPathXmlApplicationContext("d250107_1/setting.xml");

        while(true) {
            Properties pro=new Properties();
            try(FileInputStream in=new FileInputStream("D:/app
dev/stsworkspcace/basic/src/main/java/d250107_1/config.properties")){
                pro.load(in);
                Print p=(Print) app.getBean(pro.getProperty("finalObject"));
                p.print();
            }catch(Exception e) {e.printStackTrace();}
            Thread.sleep(1000);
        }
    }
}
```

#config.properties
finalObject=gridPrint

콘솔환경깨짐 설정



Common탭에서 other를 ms949로 변경

Controller-Service-DAO처리해보기

```
public class Main {  
  
    public static void main(String[] args) {  
        //Main(main)->Controller.exec()->Service.exec()->DAO.exec()  
        //역순으로 데이터가 전달이 되어 최종 main함수에서 db데이터를 받아봄.  
  
        //최종동작하기를 원하는 결과는 Controller에서 exec를 실행하여 결과를 받아보는 것임.  
        ApplicationContext app=  
            new ClassPathXmlApplicationContext  
                ("d250107_3/setting.xml");  
        Controller controller=(Controller) app.getBean("controller");  
        System.out.println(controller.exec());  
    }  
  
}
```

Controller

```
public class Controller {  
    Service service;  
  
    public Controller() {  
        System.out.println("----Controller 객체 생성----");  
    }  
    public Controller(Service service) {this.service=service;}  
    public Service getService() {return service;}  
    public void setService(Service service) {this.service = service;}  
  
    public String exec() {  
        System.out.println("컨트롤러에서 서비스 호출");  
        return service.exec();  
    }  
}
```

Service

```
public class Service {
    DAO dao;
    public Service() {System.out.println("----Service 객체 생성----");}
    public Service(DAO dao) {
        System.out.println("----Service 객체 생성(필드 생성자)----"); //bean 생성시 호출되는 함수확인
        this.dao=dao;
    }
    public DAO getDao() {return dao;}
    public void setDao(DAO dao) {this.dao = dao;}
    public String exec() {
        System.out.println("서비스에서 DAO 객체를 활용하여 exec를 호출함");
        return dao.exec();
    }
}
```

DAO

```
public class DAO {  
  
    public DAO() {System.out.println("----DAO 객체 생성----");}  
  
    public String exec() {  
        System.out.println("dao에서 데이터베이스를 처리했습니다.");  
        return "db데이터";  
    }  
  
}
```

setting.xml

```
<!-- dao객체 -->
<bean id="dao" class="d250107_3.DAO" >
</bean>

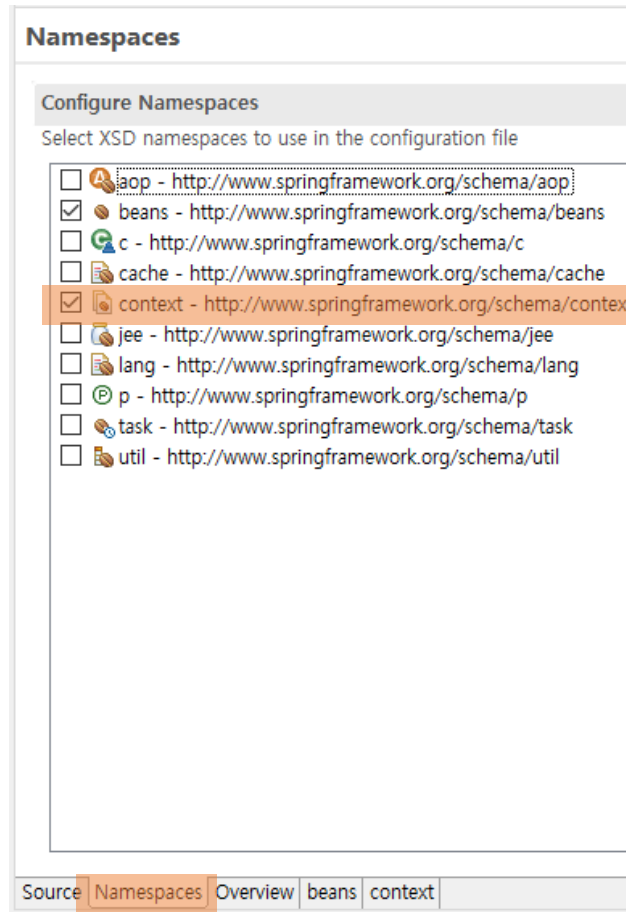
<!-- service객체 -->
<bean id="service" class="d250107_3.Service">
<constructor-arg ref="dao"></constructor-arg>
</bean>

<!-- controller객체 -->
<bean id="controller" class="d250107_3.Controller">
<property name="service" ref="service"></property>
</bean>

</beans>
```

위의 코드를 어노테이션(@)로 처리하기

- Setting.xml파일 설정



```
<context:component-scan base-package="d250107_4"/>
```

```
@Controller
public class Controller {
    @Autowired
    Service service;
```

```
@Service
public class Service {
    @Autowired
    DAO dao;
```

```
@Repository
public class DAO {
```

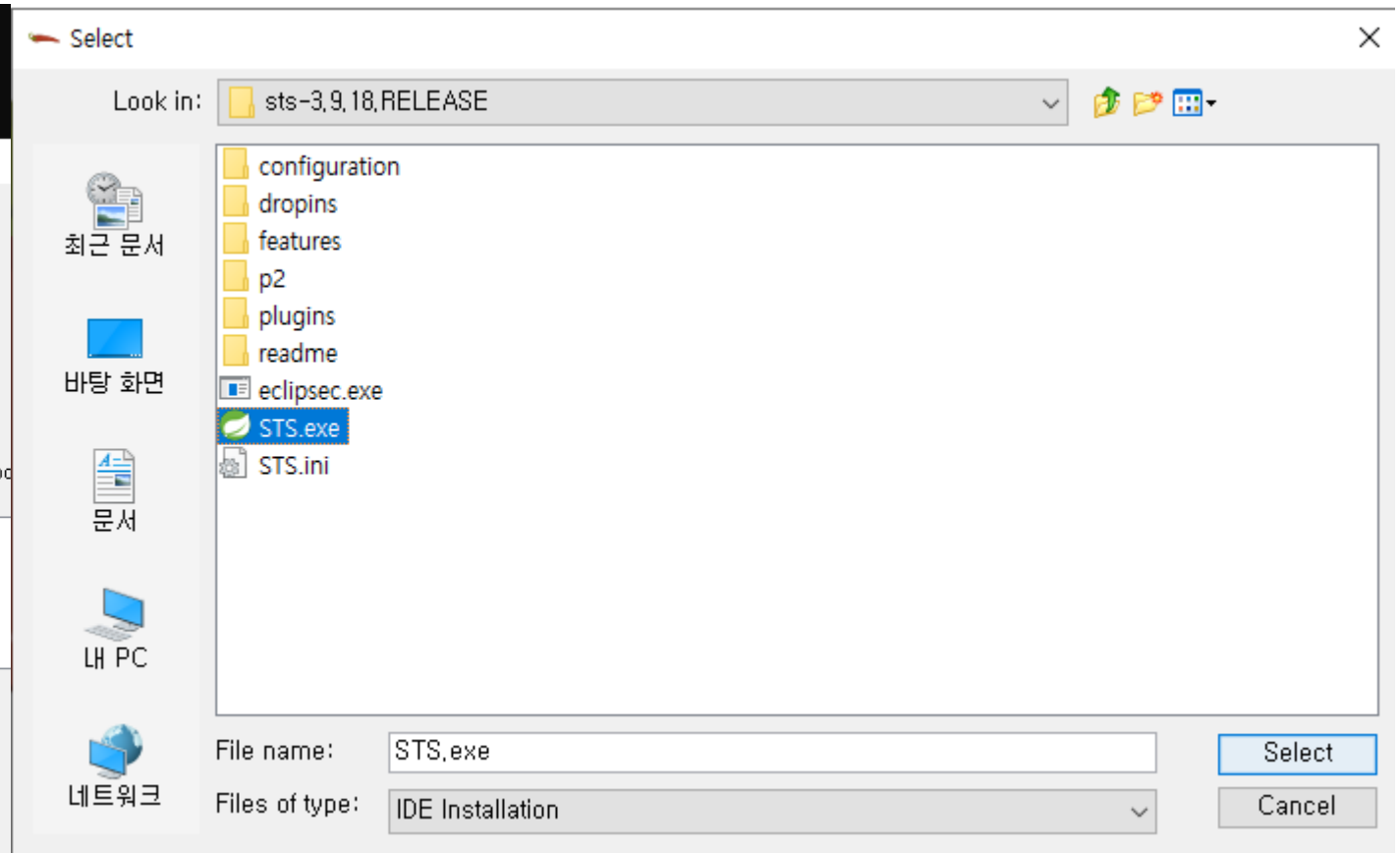
The screenshot shows the 'Project Lombok v1.18.36 - Installer' window. On the left is a large red chili pepper icon. The main text area contains:

Javac (and tools that invoke javac such as *ant* and *maven*)
Lombok works 'out of the box' with javac.
Just make sure the lombok.jar is in your classpath when you compile.
Example: `javac -cp lombok.jar MyCode.java`

IDEs
Lombok can update your Eclipse or eclipse-based IDE to fully support all Lombok annotations.
Select IDE installations below and hit 'Install/Update'.
☒ C:\Users\Wseoil\workspace\java-2023-09\workspace\workspace.exe

[Show me what this installer will do to my IDE installation.](#)

At the bottom, there are links: <https://projectlombok.org>, version `v1.18.36`, and [View full changelog](#).





(and tools that invoke javac such as *ant* and *maven*)

Example: `javac -cp lombok.jar MyCode.java`

Lombok can update your Eclipse or eclipse-based IDE to fully support all Lombok features. Select IDE installations below and hit 'Install/Update'.

- Specify location...

Show me what this installer will do to my IDE installation.

<https://projectlombok.org> v1.18.36 [View full changelog](#)

A stylized illustration of a single red chili pepper. The pepper is elongated and tapers towards the bottom, with a bright red color and a glossy sheen. It has a green stem at the top, which is slightly curved and has a small green leaf-like structure at its base. The background is plain white.

Lombok has been installed on the selected IDE installations.
Don't forget to:

- If you start STS with a custom `-vm` parameter, you'll need to add:
`-vmargs -javaagent:lombok.jar`
 as parameter as well.

- PLATFORM: JDK23 support added.
- BUGFIX: Eclipse projects using the `com.pro-crafting.tools:jasperreports-maven-plugin` will now compile.

<https://projectlombok.org> v1.18.36 [View full changelog](#)

Quit Installer

Quit Installer

*Phonebook.java basic/pom.xml x

```
8      <!-- https://mvnrepository.com/artifact/org
9<dependency>
10      <groupId>org.springframework</groupId>
11      <artifactId>spring-context</artifactId>
12      <version>5.3.39</version>
13 </dependency>
14 <!-- https://mvnrepository.com/artifact/org.p
15<dependency>
16      <groupId>org.projectlombok</groupId>
17      <artifactId>lombok</artifactId>
18      <version>1.18.36</version>
19      <scope>provided</scope>
20 </dependency>
21
```

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString

```
public class Phonebook {  
    int id;  
    String name;  
    String hp;  
    String memo;  
}
```

@Data

```
public class Phonebook {  
    int id;  
    String name;  
    String hp;  
    String memo;  
}
```

라이브러리 없이 로그사용하기

```
public class MainLog1 {  
  
    private static final Logger log  
    =Logger.getLogger(d250107_6.MainLog1.class.getName()); //패키지명 포함  
  
    public static void main(String[] args) {  
        log.warning("Log warnning");  
        log.info("Log info");  
    }  
  
}
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

public class MainLog1 {

    private static final Logger log
    =Logger.getLogger(d250107_6.MainLog1.class.getName());


    public static void main(String[] args) {
        log.info("1)log info_안내");
        log.warning("2)log warnning_경고");
        log.severe("3)log severe_심각");

        log.setLevel(Level.WARNING); //WARNING이상만 화면에 표시
        log.info("4)set level log info_안내");
        log.warning("5)set level log warnning_경고");
        log.severe("6)set level log severe_심각");

        log.setLevel(Level.ALL); //모든 로그정보 화면에 표시
        log.info("7)set level log info_안내");
        log.warning("8)set level log warnning_경고");
        log.severe("9)set level log severe_심각");
    }
}
```

파일에 기록하기

```
try {  
    FileHandler handler=new FileHandler("app.log");  
    handler.setFormatter(new SimpleFormatter());  
    Log.addHandler(handler);  
  
    Log.info("1)Log info_안내");  
    Log.warning("2)Log warnning_경고");  
    int result=1/0;  
}catch (Exception e) {  
    Log.severe("3)Log severe_심각");  
}
```

로그인으로 로그기록하기

```
try {
    FileHandler handler=new FileHandler("app.log",true);
    handler.setFormatter(new SimpleFormatter());
    Log.addHandler(handler);
    String id=null;
    String password="1111";
    //시나리오 : id, password일치할 때, 일치하지 않을 때, id or
    password값이 null일때
    if(id.equals("admin") && password.equals("1111")) {
        Log.info("Login sucess!!");
    }else {
        Log.warning("Login fail!! id:"+id);
    }

}

}catch (Exception e) {
    Log.severe(e.getMessage());
}

}
```