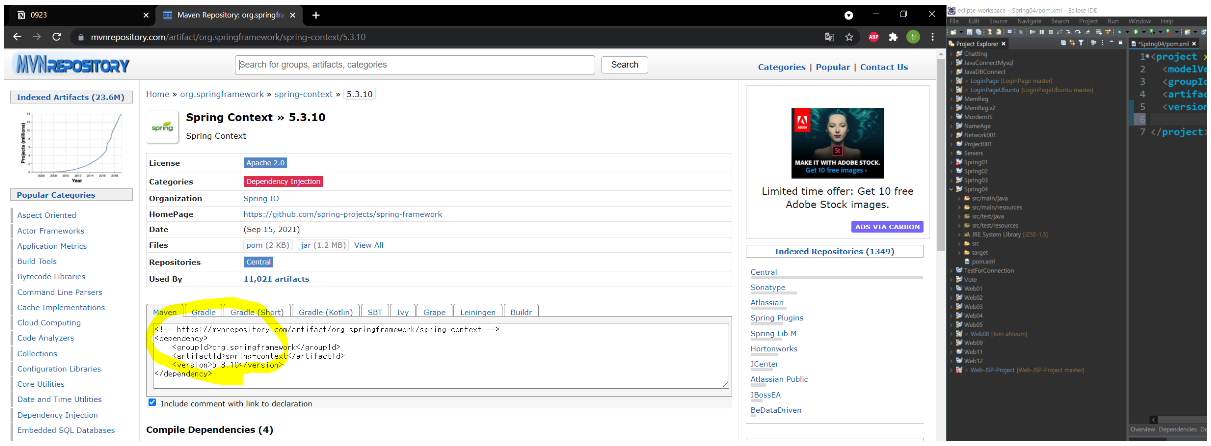
0923

Maven 프로젝트 심플로 하나 생성



Home » org.springframework » spring-context » 5.3.10

Spring Context » 5.3.10

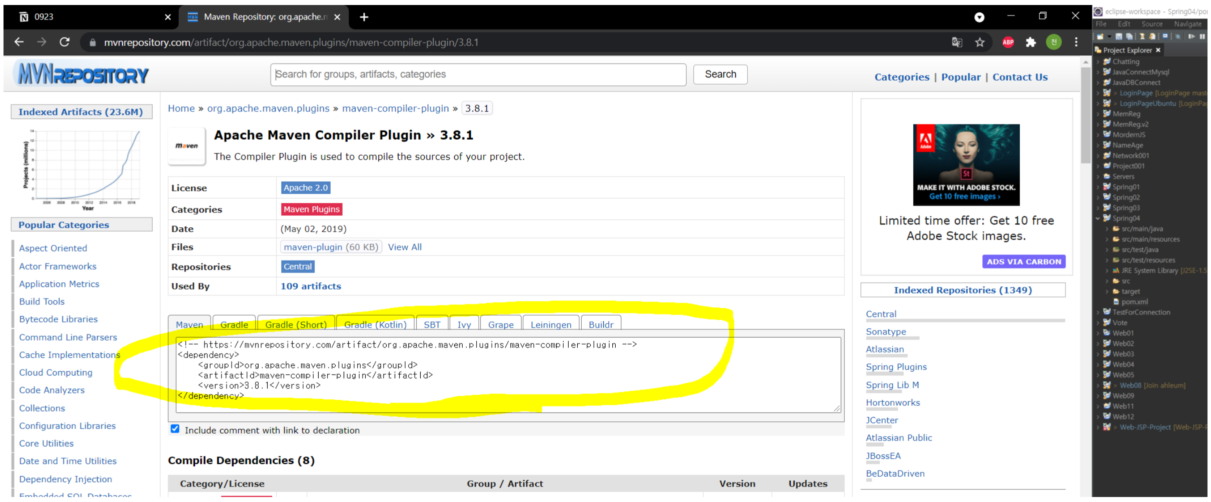
Spring Context

License	Apache 2.0
Categories	Dependency Injection
Organization	Spring IO
HomePage	https://github.com/spring-projects/spring-framework
Date	(Sep 15, 2021)
Files	pom (2 KB) jar (1.2 MB) View All
Repositories	Central
Used By	11,021 artifacts

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.10</version>
</dependency>
```

Include comment with link to declaration

Compile Dependencies (4)



Home » org.apache.maven.plugins » maven-compiler-plugin » 3.8.1

Apache Maven Compiler Plugin » 3.8.1

The Compiler Plugin is used to compile the sources of your project.

License	Apache 2.0
Categories	Maven Plugins
Date	(May 02, 2019)
Files	maven-plugin (60 KB) View All
Repositories	Central
Used By	109 artifacts

```
<!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-compiler-plugin -->
<dependency>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
</dependency>
```

Include comment with link to declaration

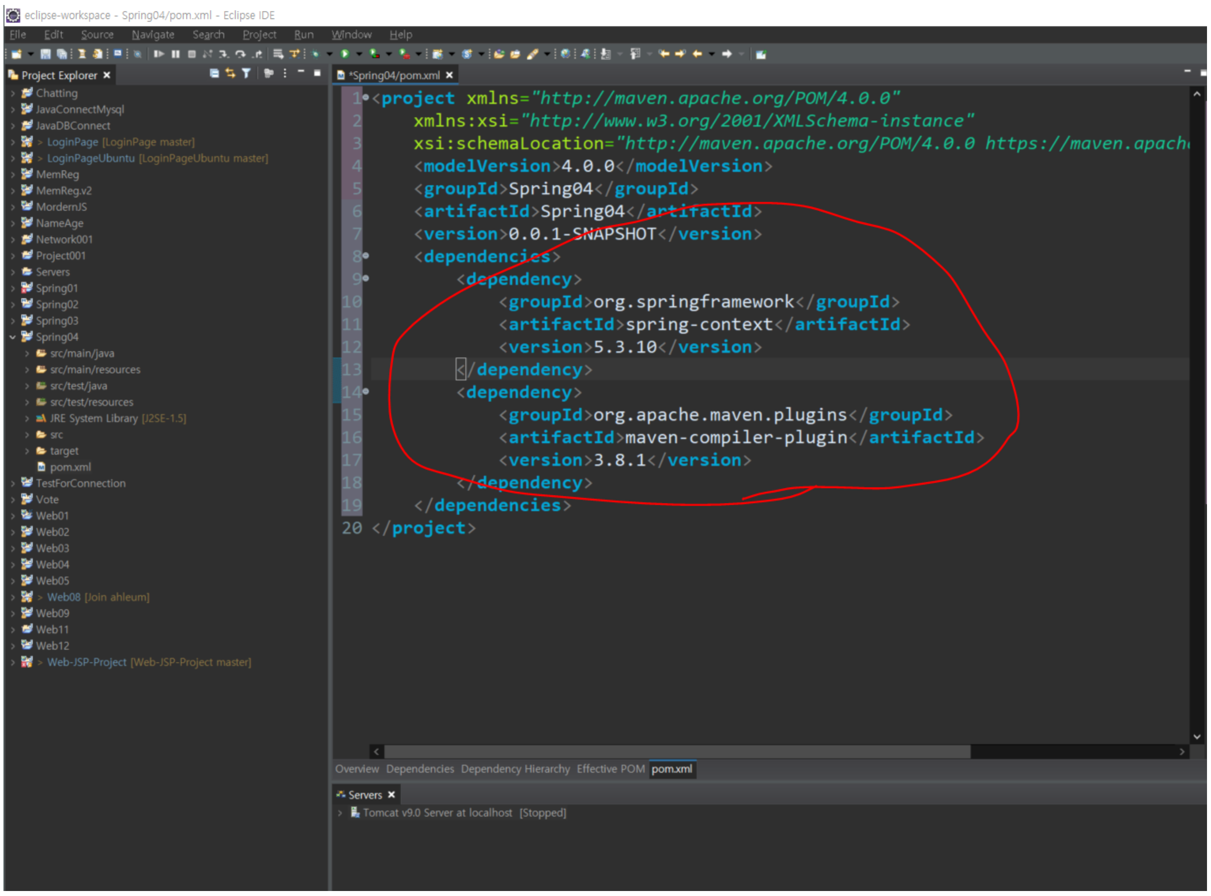
Compile Dependencies (8)

Category/License	Group / Artifact	Version	Updates

https://mvnrepository.com/

Spring context 가장 최신 버전 추가

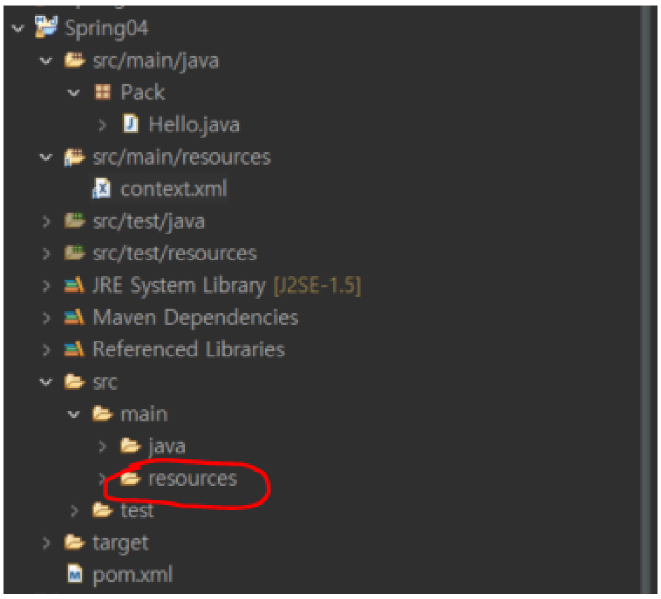
Apache Maven Compiler Plugin 최신 버전 추가


eclipse-workspace - Spring04/pom.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
Project Explorer
Chatting
JavaConnectMysql
JavaDBConnect
LoginPage [LoginPage master]
LoginPageUbuntu [LoginPageUbuntu master]
MemReg
MemReg.v2
MordernJS
NameAge
Network001
Project001
Servers
Spring01
Spring02
Spring03
Spring04
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
src
target
pom.xml
TestForConnection
Vote
Web01
Web02
Web03
Web04
Web05
Web08 [Join ahleum]
Web09
Web11
Web12
Web-JSP-Project [Web-JSP-Project master]
*Spring04/pom.xml
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Servers
Tomcat v9.0 Server at localhost [Stopped]

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apach
    <modelVersion>4.0.0</modelVersion>
    <groupId>Spring04</groupId>
    <artifactId>Spring04</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.10</version>
        </dependency>
        <dependency>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
        </dependency>
    </dependencies>
</project>
```

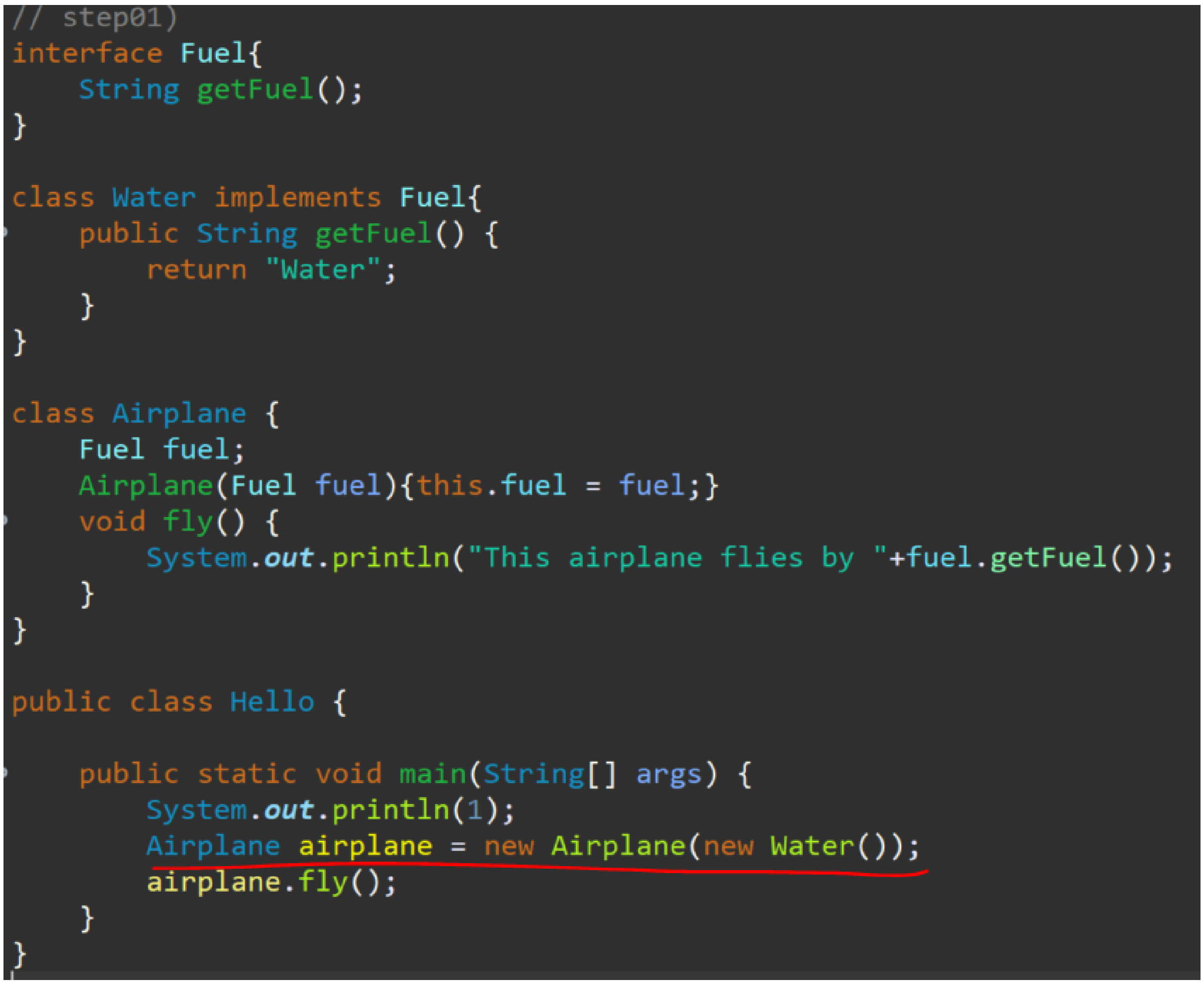
추가하고 alt + f5 force 옵션 체크

a를 사용하기 위해서 필요한 b 같은 경우는 alt + f5 force 옵션 체크 과정에서 자동으로 다운로드 된다.



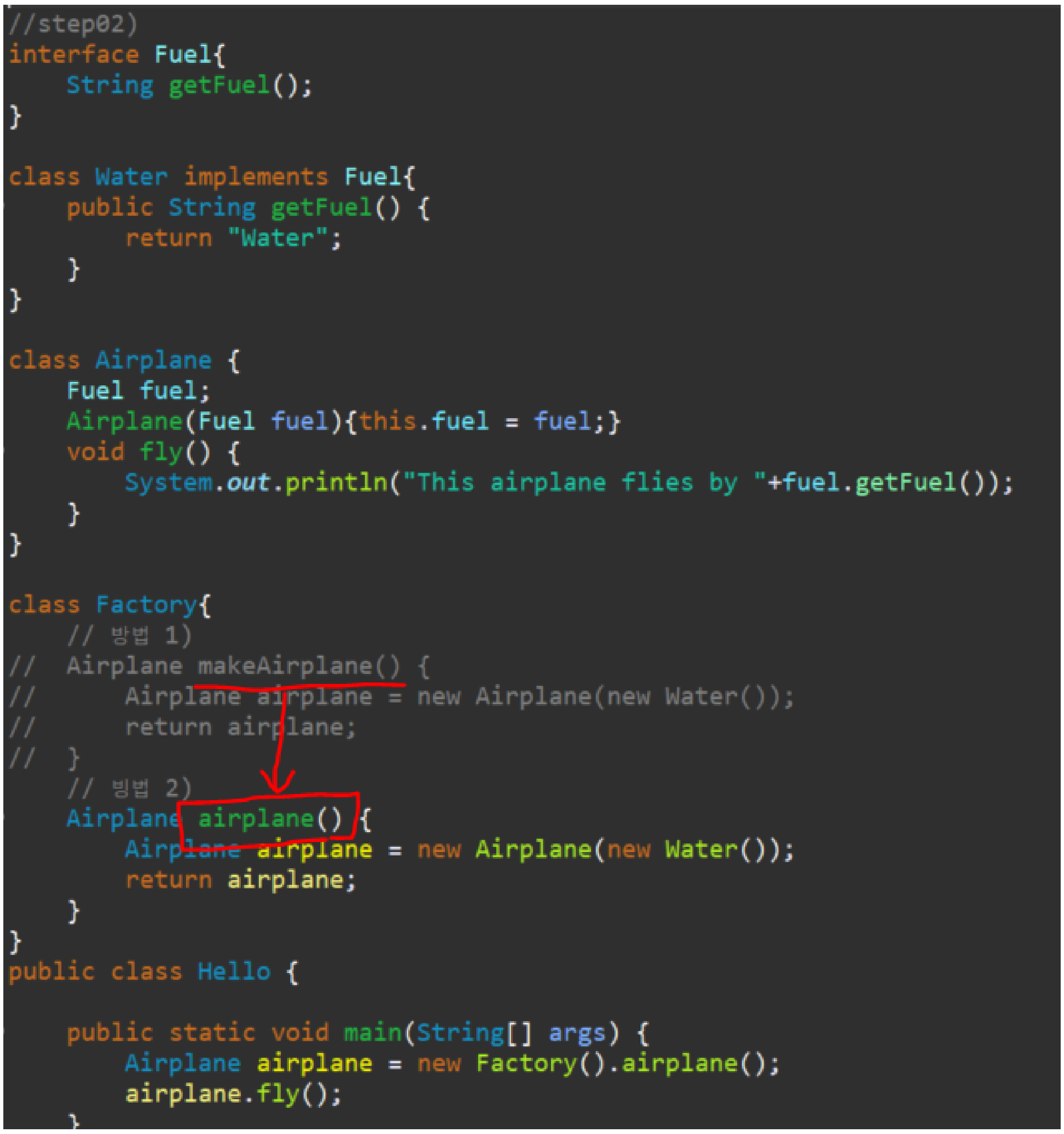
src-main-resources 디렉토리 안에 context.xml 파일 생성

1. 평소에 쓰던 방식 객체를 직접 생성하고 함수 사용



```java
// step01)
interface Fuel{
    String getFuel();
}

class Water implements Fuel{
    public String getFuel() {
        return "Water";
    }
}

class Airplane {
    Fuel fuel;
    Airplane(Fuel fuel){this.fuel = fuel;}
    void fly() {
        System.out.println("This airplane flies by "+fuel.getFuel());
    }
}

public class Hello {

    public static void main(String[] args) {
        System.out.println(1);
        Airplane airplane = new Airplane(new Water());
        airplane.fly();
    }
}
```

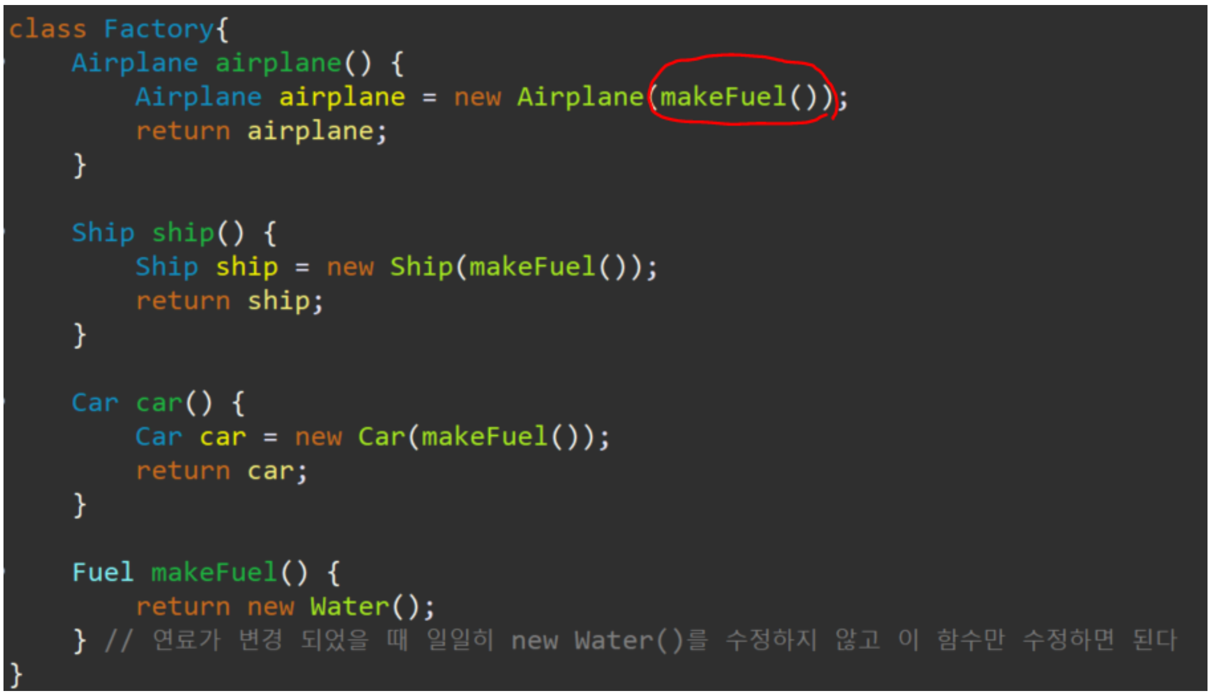
2. makeAirplane 에서 airplane 으로 메소드명 변경



```
//step02)
interface Fuel{
    String getFuel();
}

class Water implements Fuel{
    public String getFuel() {
        return "Water";
    }
}

class Airplane {
    Fuel fuel;
    Airplane(Fuel fuel){this.fuel = fuel;}
    void fly() {
        System.out.println("This airplane flies by "+fuel.getFuel());
    }
}

class Factory{
    // 방법 1)
//  Airplane makeAirplane() {
//      Airplane airplane = new Airplane(new Water());
//      return airplane;
//  }
    // 방법 2)
    Airplane airplane() {
        Airplane airplane = new Airplane(new Water());
        return airplane;
    }
}
public class Hello {

    public static void main(String[] args) {
        Airplane airplane = new Factory().airplane();
        airplane.fly();
    }
```

3. 일일이 연료 객체 생성 ⇒ 메소드로 연료 객체 생성

```
interface Fuel{
  String getFuel();
}

class Water implements Fuel{
  public String getFuel() {
    return "Water";
  }
}

class Airplane {
  Fuel fuel;
  Airplane(Fuel fuel){this.fuel = fuel;}
  void fly() {
    System.out.println("This airplane flies by "+fuel.getFuel());
  }
}

class Ship {
  Fuel fuel;
  Ship(Fuel fuel){this.fuel=fuel;}
}

class Car {
  Fuel fuel;
  Car(Fuel fuel){this.fuel=fuel;}
}

class Factory{
  Airplane airplane() {
    Airplane airplane = new Airplane(makeFuel());
    return airplane;
  }

  Ship ship() {
    Ship ship = new Ship(makeFuel());
    return ship;
  }

  Car car() {
    Car car = new Car(makeFuel());
    return car;
  }

  Fuel makeFuel() {
    return new Water();
  } // 연료가 변경 되었을 때 일일히 new Water()를 수정하지 않고 이 함수만 수정하면 된다
}
public class Hello {

  public static void main(String[] args) {
    Airplane airplane = new Factory().airplane();
    airplane.fly();

    Ship ship = new Factory().ship();
    Car car = new Factory().car();
  }
}
```

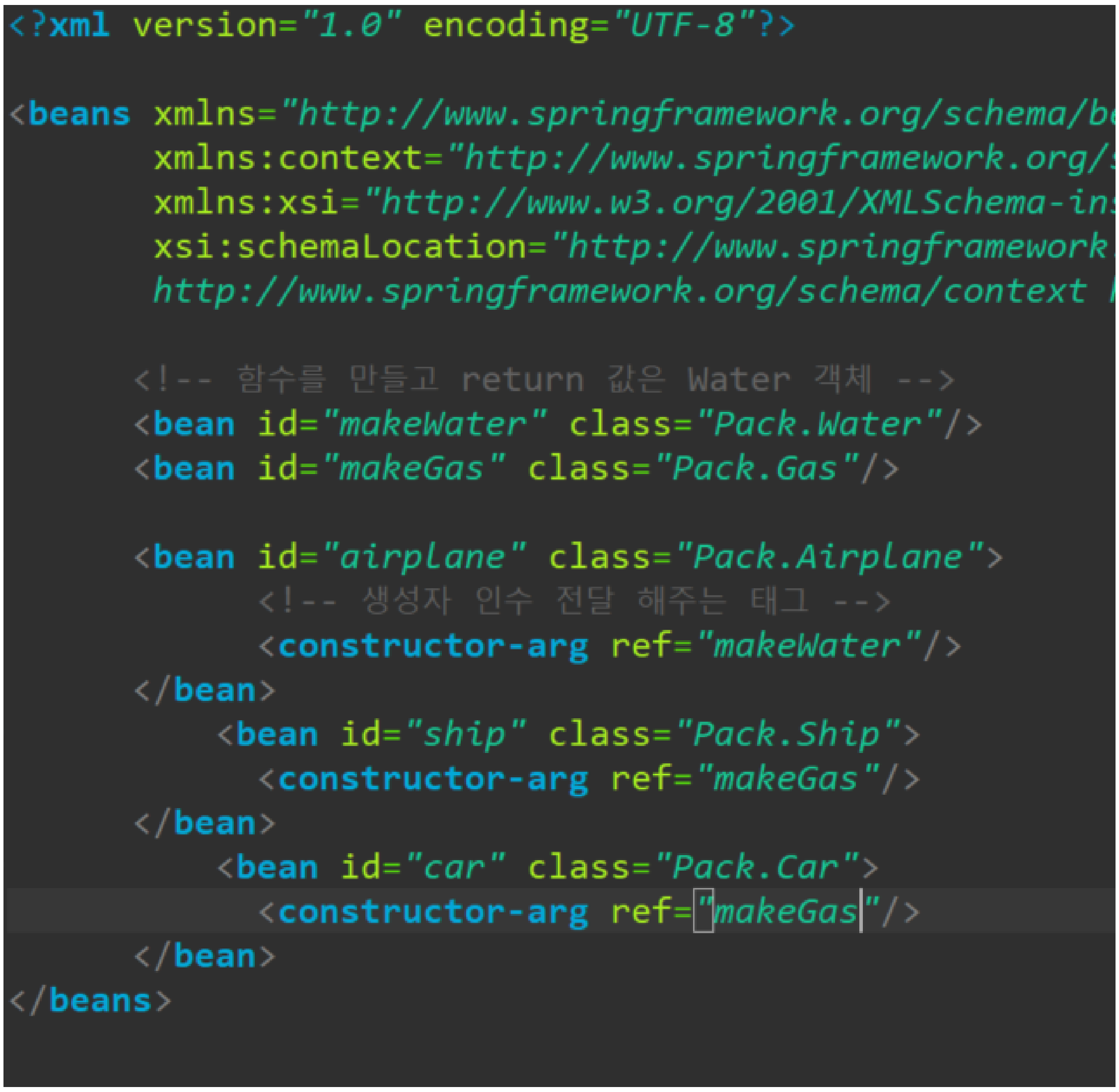


```
class Factory{
    Airplane airplane() {
        Airplane airplane = new Airplane(makeFuel());
        return airplane;
    }

    Ship ship() {
        Ship ship = new Ship(makeFuel());
        return ship;
    }

    Car car() {
        Car car = new Car(makeFuel());
        return car;
    }

    Fuel makeFuel() {
        return new Water();
    } // 연료가 변경 되었을 때 일일히 new Water()를 수정하지 않고 이 함수만 수정하면 된다
}
```

4. 어노테이션 활용하여 객체 생성

```
//step05)
interface Fuel{
  String getFuel();
}

class Water implements Fuel{
  public String getFuel() {
    return "Water";
  }
}

class Airplane {
  Fuel fuel;
  Airplane(Fuel fuel){this.fuel = fuel;}
  void fly() {
    System.out.println("This airplane flies by "+fuel.getFuel());
  }
}

@Configuration
class Factory{
  @Bean
  Fuel makeFuel() {
    return new Water();
  }

  @Bean
  Airplane airplane() {
    return new Airplane(makeFuel());
  }
}
public class Hello {
  public static void main(String[] args) {
    AnnotationConfigApplicationContext ctx =
        new AnnotationConfigApplicationContext(Factory.class);

    //Airplane airplane = new Factory().airplane();
    //내가 직접하지 않고 스프링이 대신해서 객체 생성
    Airplane airplane = ctx.getBean("airplane", Airplane.class);
    airplane.fly();

    ctx.close();
  }
}
```

5. xml 활용



```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/be
       xmlns:context="http://www.springframework.org/s
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
       xsi:schemaLocation="http://www.springframework
       http://www.springframework.org/schema/context

      <!-- 함수를 만들고 return 값은 Water 객체 -->
      <bean id="makeWater" class="Pack.Water"/>
      <bean id="makeGas" class="Pack.Gas"/>

      <bean id="airplane" class="Pack.Airplane">
            <!-- 생성자 인수 전달 해주는 태그 -->
            <constructor-arg ref="makeWater"/>
      </bean>
          <bean id="ship" class="Pack.Ship">
            <constructor-arg ref="makeGas"/>
      </bean>
          <bean id="car" class="Pack.Car">
            <constructor-arg ref="makeGas"/>
      </bean>
</beans>
```

```
interface Fuel{
  String getFuel();
}
```

```
class Water implements Fuel{
  public String getFuel() {
    return "Water";
  }
}

class Gas implements Fuel{
  public String getFuel() {
    return "Gas";
  }
}

class Airplane {
  Fuel fuel;
  Airplane(Fuel fuel){this.fuel = fuel;}
  void fly() {
    System.out.println("This airplane flies by "+fuel.getFuel());
  }
}

class Ship {
  Fuel fuel;
  Ship(Fuel fuel){this.fuel=fuel;}
  void sail() {
    System.out.println("This ship sails by "+fuel.getFuel());
  }
}

class Car {
  Fuel fuel;
  Car(Fuel fuel){this.fuel=fuel;}
  void run() {
    System.out.println("This car runs by "+fuel.getFuel());
  }
}

public class Hello {

  public static void main(String[] args) {
    GenericXmlApplicationContext ctx =
        new GenericXmlApplicationContext("Context.xml");

    Airplane airplane = ctx.getBean("airplane", Airplane.class);
    airplane.fly();

    Ship ship = ctx.getBean("ship", Ship.class);
    ship.sail();

    Car car = ctx.getBean("car", Car.class);
    car.run();
  }
}
```

MyBatis - DB 연결

1. xml dependency 추가

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>MyBatis</groupId>
  <artifactId>MyBatis</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
      <version>3.5.7</version>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.26</version>
    </dependency>
    <dependency>
      <groupId>commons-dbcp</groupId>
      <artifactId>commons-dbcp</artifactId>
      <version>1.4</version>
    </dependency>
    <dependency>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
    </dependency>
  </dependencies>
</project>
```

https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp

- [ ] my batis
- [ ] my sql
- [ ] commons-dbcp

2. src-main-resources 디렉토리에 mybatis-config.xml, Mapper.xml 파일 생성

mybatis-config.xml

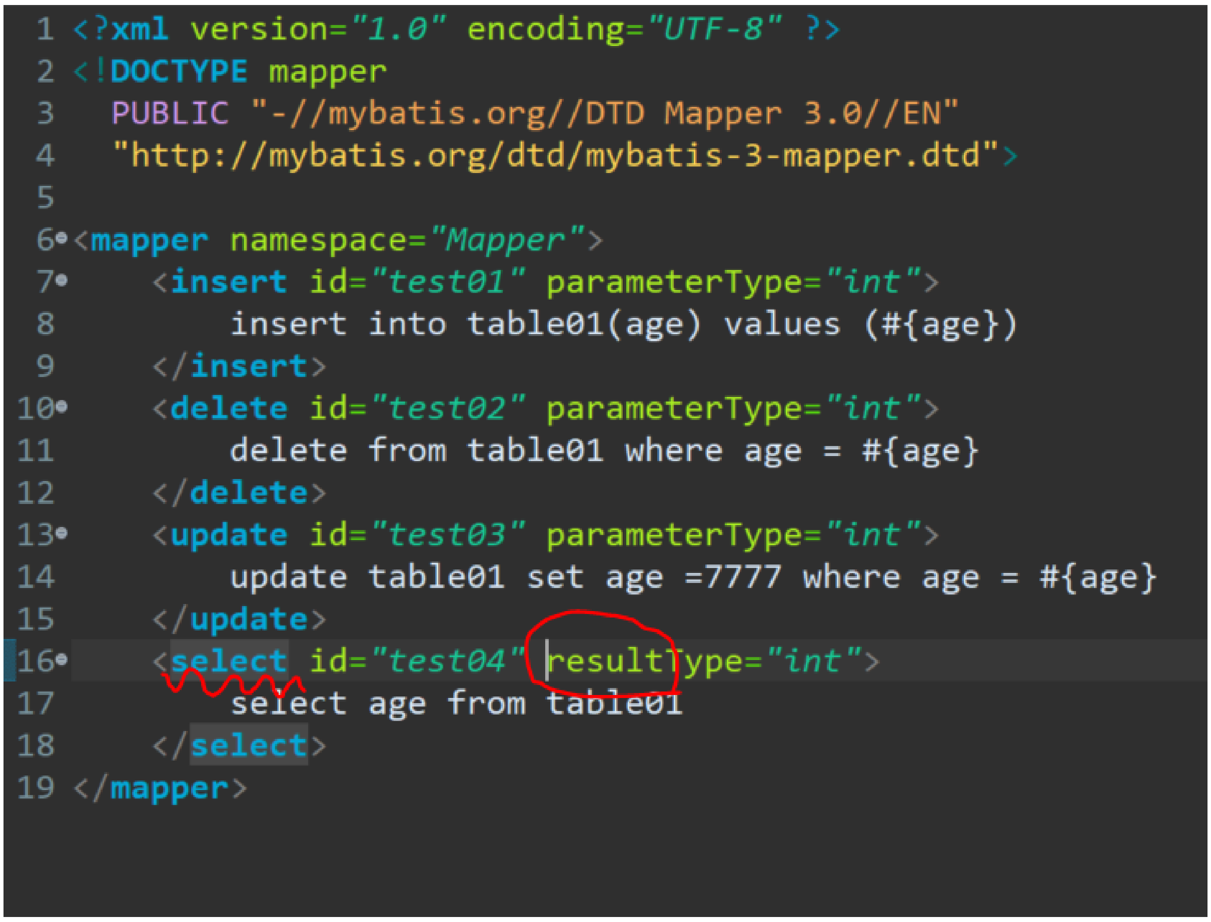
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC" />
      <dataSource type="POOLED">
        <property name="driver" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost/db01" />
        <property name="username" value="root" />
        <property name="password" value="1234" />
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="Mapper.xml" />
  </mappers>
</configuration>
```

Mapper.xml

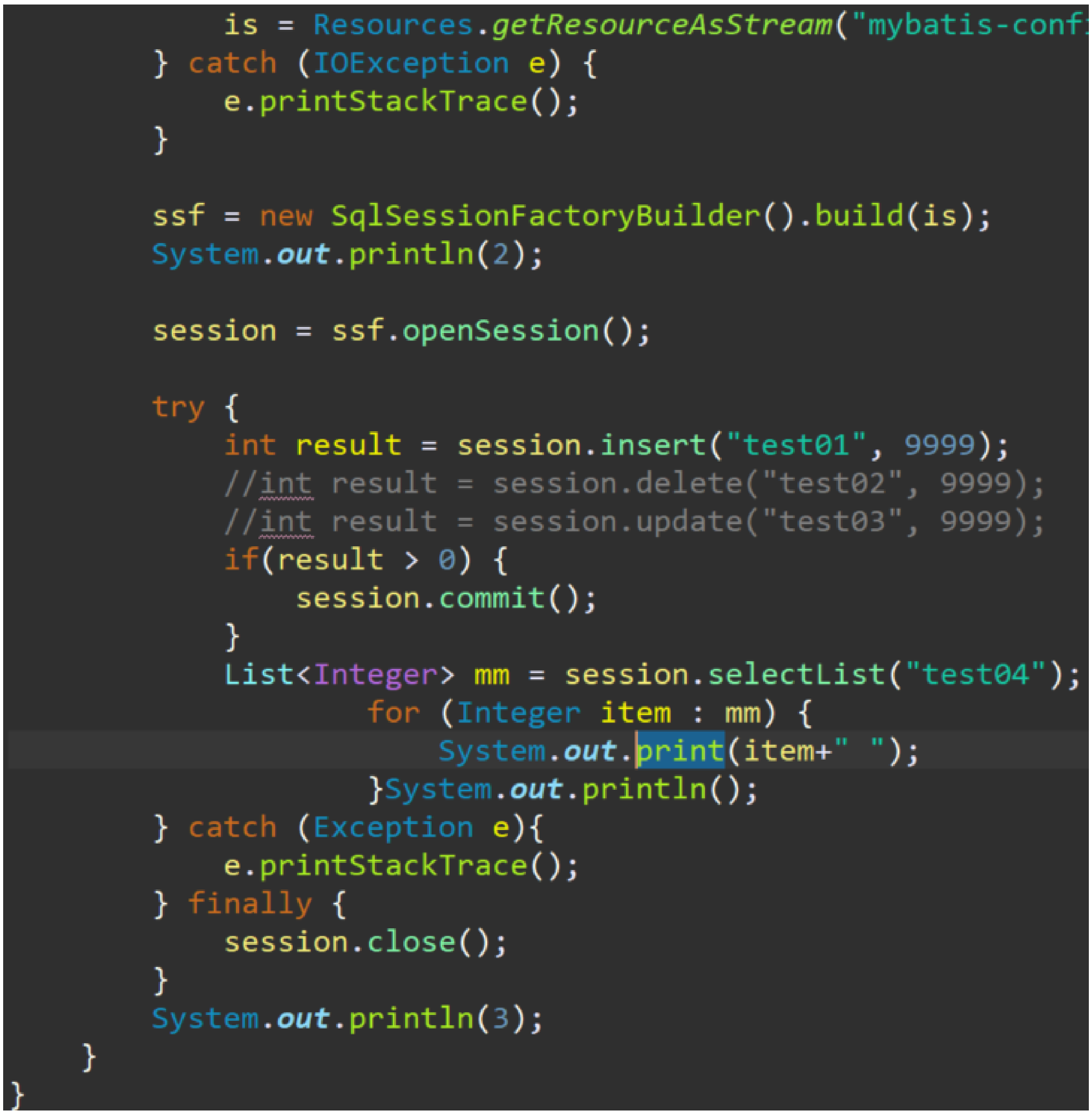
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="Mapper">
  <insert id="test01" parameterType="int">
    insert into table01(age) values (#{age})
  </insert>
</mapper>
```

Hello.java

```
package Pack;

import java.io.IOException;
import java.io.InputStream;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class Hello {

  public static void main(String[] args) {
    System.out.println(1);
    SqlSessionFactory ssf = null;
    SqlSession session = null;

    InputStream is = null;

    try {
      is = Resources.getResourceAsStream("mybatis-config.xml");
    } catch (IOException e) {
      e.printStackTrace();
    }

    ssf = new SqlSessionFactoryBuilder().build(is);
    System.out.println(2);

    session = ssf.openSession();

    try {
      int result = session.insert("test01", 9999);
      System.out.println(result);
      if(result>0) {
        session.commit(); // 커밋 해줘야지 워크벤치에서 확인 가능
      }
    } catch (Exception e){
      e.printStackTrace();
    } finally {
      session.close();
    }

    System.out.println(3);
  }
}
```



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="Mapper">
    <insert id="test01" parameterType="int">
        insert into table01(age) values (#{age})
    </insert>
    <delete id="test02" parameterType="int">
        delete from table01 where age = #{age}
    </delete>
    <update id="test03" parameterType="int">
        update table01 set age =7777 where age = #{age}
    </update>
    <select id="test04" resultType="int">
        select age from table01
    </select>
</mapper>
```



```
            is = Resources.getResourceAsStream("mybatis-conf
        } catch (IOException e) {
            e.printStackTrace();
        }

        ssf = new SqlSessionFactoryBuilder().build(is);
        System.out.println(2);

        session = ssf.openSession();

        try {
            int result = session.insert("test01", 9999);
            //int result = session.delete("test02", 9999);
            //int result = session.update("test03", 9999);
            if(result > 0) {
                session.commit();
            }
            List<Integer> mm = session.selectList("test04");
                    for (Integer item : mm) {
                        System.out.print(item+" ");
                    }System.out.println();
        } catch (Exception e){
            e.printStackTrace();
        } finally {
            session.close();
        }
        System.out.println(3);
    }
}
```