

0907 - 자바 코드로 직접 스프링 빈 등록

springConfig

```
package hello.hello_spring;

import hello.hello_spring.repository.MemberRepository;
import hello.hello_spring.repository.MemoryMemberRepository;
import hello.hello_spring.service.MemberService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class springConfig {

    @Bean
    public MemberService memberService() {
        return new MemberService(memberRepository());
    }

    @Bean
    public MemberRepository memberRepository() {
        return new MemoryMemberRepository();
    }
}
```

MemberService

```
package hello.hello_spring.service;

import hello.hello_spring.domain.Member;
```

```

import hello.hello_spring.repository.MemberRepository;
import hello.hello_spring.repository.MemoryMemberRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

public class MemberService {

    private final MemberRepository memberRepository;

    public MemberService(MemberRepository memberRepository) {
        this.memberRepository = memberRepository;
    }

    /**
     * 회원가입
     */
    public Long join(Member member) {
        //중복 회원 검증
        validateDuplicateMember(member);
        memberRepository.save(member);
        return member.getId();
    }

    private void validateDuplicateMember(Member member) {
        memberRepository.findByName(member.getName())
            .ifPresent(m -> {
                throw new IllegalStateException("이미 존재하는 회원입니다.");
            });
    }

    /**
     * 전체 회원 조회
     */
    public List<Member> findMembers() {
        return memberRepository.findAll();
    }
}

```

```

    }

    /**
     * 회원 조회
     */
    public Optional<Member> findOne(Long memberId) {
        return memberRepository.findById(memberId);
    }
}

```

MemoryMemberRepository

```

package hello.hello_spring.repository;

import hello.hello_spring.domain.Member;
import org.springframework.stereotype.Repository;

import java.util.*;

public class MemoryMemberRepository implements MemberRepository {

    private static Map<Long, Member> store = new HashMap<>();
    //시퀀스
    private static long sequence = 0L;

    @Override
    public Member save(Member member) {
        member.setId(++sequence);
        store.put(member.getId(), member);
        return member;
    }

    @Override
    public Optional<Member> findById(Long id) {
        return Optional.ofNullable(store.get(id));
    }
}

```

```

    }

    @Override
    public Optional<Member> findByName(String name) {
        return store.values().stream()
            .filter(member -> member.getName().equals(name))
            .findAny();
    }

    @Override
    public List<Member> findAll() {
        return new ArrayList<>(store.values());
    }

    public void clearStore(){
        store.clear();
    }
}

```

```

pring] [ restartedMain] h.hello_spring.HelloSpringApplication : Starting HelloSpringApplication using Java 17.0.8 with PID 14428 (C:\develop\springBoot\he
pring] [ restartedMain] h.hello_spring.HelloSpringApplication : No active profile set, falling back to 1 default profile: "default"
pring] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 9904 (http)
pring] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
pring] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.28]
pring] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
pring] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 464 ms
pring] [ restartedMain] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page: class path resource [static/index.html]
pring] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
pring] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9904 (http) with context path '/'
pring] [ restartedMain] h.hello_spring.HelloSpringApplication : Started HelloSpringApplication in 0.686 seconds (process running for 98.782)
pring] [ restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged

```

• 필드 주입

```

@Autowired
private MemberService memberService;

```

• setter 주입

```
@Autowired
public void setMemberService(MemberService memberService) {
    this.memberService = memberService;
}
```

- **생성자 주입**

```
private final MemberService memberService;

@Autowired
public MemberController(MemberService memberService) {
    this.memberService = memberService;
}
```

- 의존관계가 실행 중에 동적으로 변하는 경우가 거의 없기 때문에 생성자 주입을 권장
- 실무에서는 주로 정형화된 컨트롤러, 서비스, 리포지토리 같은 코드는 컴포넌트 스캔 사용
- 정형화 되지 않거나 상황에 따라 구현 클래스를 변경해야 하면 설정을 통해 스프링 빈으로 등록