2020년도 2학기

스프링프레임워크

[교재 Class03 스프링 컨테이너 및 설정파일] p49~

스프링 실습 프로젝트 정리 실습

- SpringProject1 : //개발자에 의한 객체 결합
  - com.myspring.step1 : 객체간의 결합도가 높은 예제
  - com.myspring.step2 : 다형성을 이용해 결합도를 낮춤
  - com.myspring.step3 : Factory디자인패턴을 이용해 결합도를 낮춤
  - com.myspring.step4 : 별도의 텍스트파일을 이용해 소스를 다시 컴파일하지않도록 수정

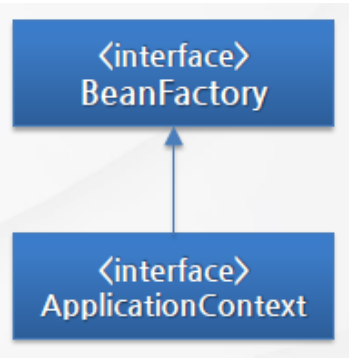
- SpringProject2 : //스프링컨테이너에 의한 객체 결합
  - com.myspring.step1

스프링 컨테이너 p54

: 스프링 프레임워크를 초기화 하는 역할

- 처리 순서
  - 스프링 컨테이너 생성
  - Bean들이 들어있는 XML 파일 읽음.
  - XML 파일에 등록된 Bean들의 Life Cycle과 Dependency가 관리되기 시작
- 스프링 DI컨테이너의 종류

BeanFactory	- 스프링 설정 파일에 등록된 <bean>객체를 생성하고 관리하는 가장 기본적인 컨테이너 - 컨테이너가 구동될때가 아닌 클라이언트의 요청(Lookup)에 의해서 생성되는 지연로딩(Lazy Loading) 방식 (일반적으로 사용 x)
ApplicationContext	- BeanFactory를 상속한 컨테이너로, BeanFactory가 제공하는 <bean>객체관리 이외 트랜잭션 관리, 메시지 기반 다국어처리 등 다양한 기능 제공 - 컨테이너가 구동되는 시점에 객체생성하는 즉시로딩(pre-loading) 방식 - 웹 애플리케이션 개발 가능 - GenericXmlApplicationContext, XmlWebApplicationContext 들의 구현 클래스들이 있다.

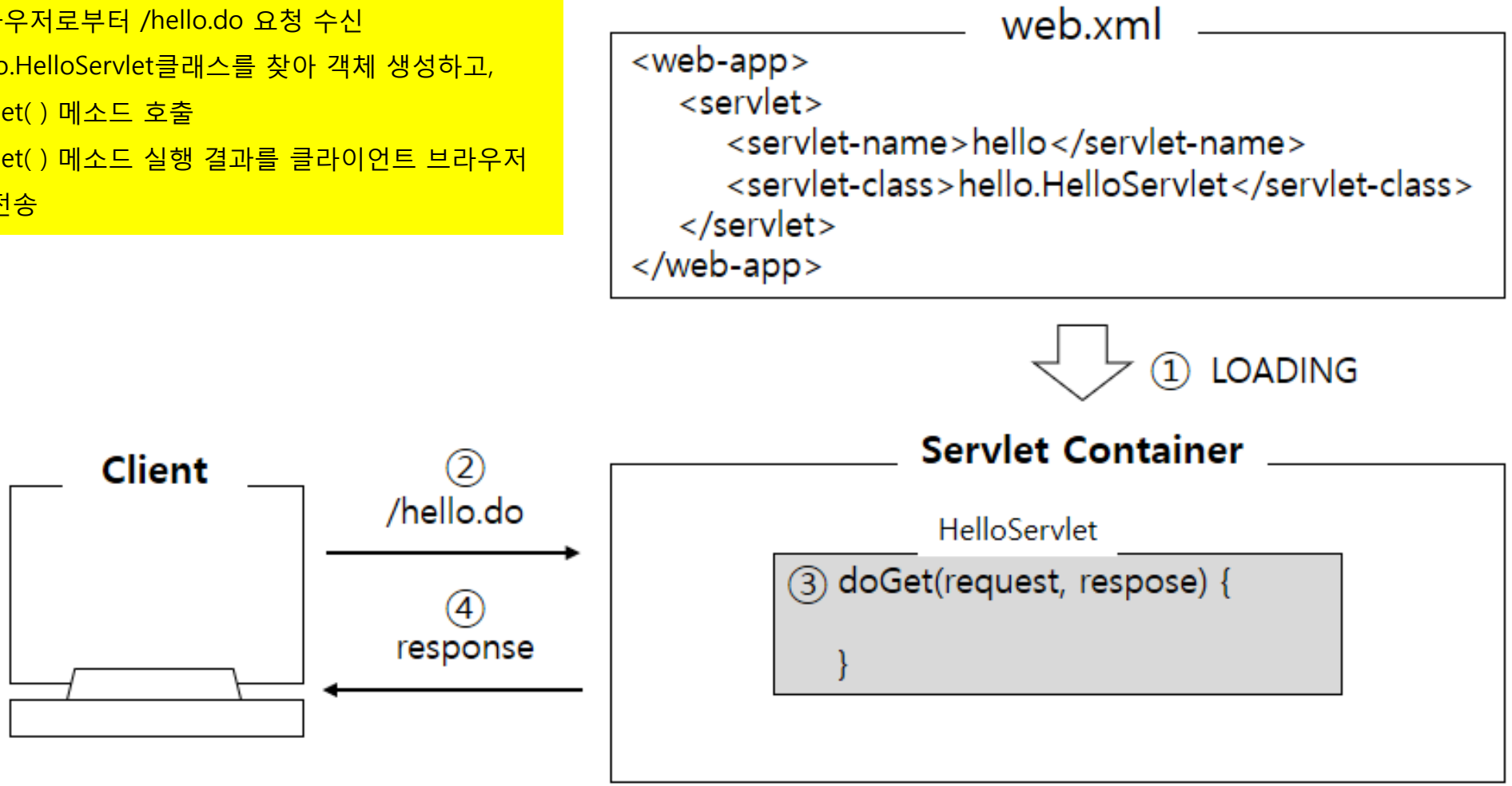


구현 클래스	기능
GenericXmlApplicationContext	파일 시스템이나 클래스 경로에 있는 XML 설정 파일을 로딩하여 구동하는 컨테이너. 직접 생성
XmlWebApplicationContext	웹 기반의 스프링 애플리케이션을 개발할 때 사용하는 컨테이너. 직접 생성하지않음.

스프링 컨테이너 vs Servlet 컨테이너

- Servlet 컨테이너의 동작

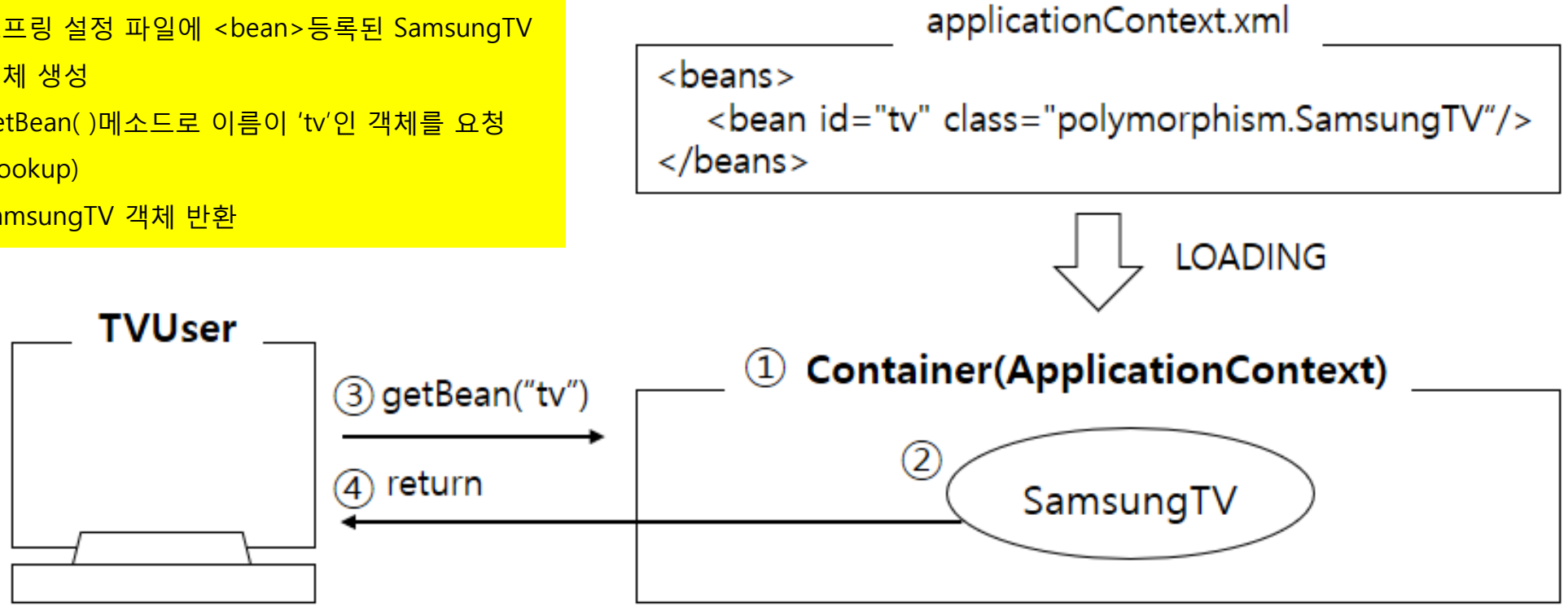
① WEB-INF/web.xml 파일을 로딩하여 구동
② 브라우저로부터 /hello.do 요청 수신
③ hello.HelloServlet클래스를 찾아 객체 생성하고, doGet( ) 메소드 호출
④ doGet( ) 메소드 실행 결과를 클라이언트 브라우저로 전송



스프링 컨테이너 vs Servlet 컨테이너

- Spring 컨테이너의 동작

① TVUser 클라이언트가 스프링 설정 파일을 로딩하여 컨테이너 구동
② 스프링 설정 파일에 <bean>등록된 SamsungTV 객체 생성
③ getBean( )메소드로 이름이 'tv'인 객체를 요청 (Lookup)
④ SamsungTV 객체 반환



실습 [SpringProject2] 의 [com.myspring.step1]

스프링 컨테이너 설정파일

- <beans>루트 엘리먼트
  - 스프링 설정파일 이름을 상관없음. 단, <beans>루트 엘리먼트 사용
  - <bean>의 생명주기를 관리하고 여러 가지 서비스를 제공하는 가장 중요한 역할 담당
  - <beans>엘리먼트 시작태그에 네임스페이스를 비롯한 XML 스키마 관련 정보 설정
  - spring-beans.xsd스키마문서가 schemaLoacation 등록 : <import>,<bean>,<description>,<alias>의 자식 엘리먼트로 사용 가능
- <import> 엘리먼트
  - 개발자가 만든 모든 클래스를 <bean>으로 등록/관리 하지만,
    단순히 <bean>등록 외에도 트랜잭션 관리, 예외처리, 다국어 처리등의 복잡한 다양한 설정필요로 설정 파일이 길어짐
    ▶▷▶기능별 여러 XML파일로 분리 설정

```
예 ) applicationContext.xml    :
    <beans>
            <import resource="context-datasource.xml"/>
            <import resource="context-transaction.xml"/>
    </beans>
```

- <bean>엘리먼트
  - 클래스 등록시 사용하는 엘리먼트로서 id속성(생략가능), class속성(필수) 사용

```
예 ) applicationContext.xml    :
    <beans xmlns=" ..........>
        <bean
      class="com.myspring.step04.SamsungTV"></bean>
    </beans>
```

```
id속성명은 자바식별자작성규칙에 따름 (특수기호가 포함된 id는 name속성으로 대신)
-→ 속성명 예
   id="7userService" (x 숫자로 시작)
   id="user service" (x 공백 포함)
   id="user#Service:Impl" (x 특수기호 사용)
  name="http://www.daum.net"
```

<bean>엘리먼트 속성 (1)

(1)<bean>엘리먼트 속성 : init-method 속성

- Servlet의 init( )과 같은 역할 : 객체 생성시 초기화 작업

```
예 ) <bean id="tv" class="com.myspring.step04.SamsungTV" init-method="initMethod" />
```

```
SamsungTV.java
pakage com.myspring.step04;
pulic class SamsungTV implements TV{
        public void initMethod( ){
                System.out.println("객체 초기화 작업 처리...""); }
        .....
```

(2) <bean>엘리먼트 속성 : destory-method 속성

```
예 ) <bean id="tv" class="com.myspring.step04.SamsungTV" destory-method="destoryMethod" />
```

```
SamsungTV.java
pakage com.myspring.step04;
pulic class SamsungTV implements TV{
        public void destoryMethod( ){
                System.out.println("객체 삭제 전 처리할 작업...""); }
        .....
```

<bean>엘리먼트 속성 (2)

(3) <bean>엘리먼트 속성 : lazy-init 속성

- 즉시로딩(pre-loading)방식이지만, 요청할 때 생성하도록

```
예 ) <bean id="tv" class="com.myspring.step04.SamsungTV" lazy-init="true" />
```

(4)<bean>엘리먼트 속성 : scope 속성

- scope속성의 기본값은 싱글톤 : bean 객체가 단 하나만 생성

```
예 ) <bean id="tv" class="com.myspring.step04.SamsungTV" scope="singleton" />
```

```
public class TVUser {
   public static void main(String[] args) {
       AbstractApplicationContext factory= new GenericXmlApplicationContext("applicationContext.xml");
       TV tv1 = (TV) factory.getBean("tv");
       TV tv2 = (TV) factory.getBean("tv");
       TV tv3 = (TV) factory.getBean("tv");    .....
```

```
< 결과 >
.....
===> SamsungTV 객체 생성
.....
```

```
예 ) <bean id="tv" class="com.myspring.step04.SamsungTV" scope="prototype" />
```

```
public class TVUser {
   public static void main(String[] args) {
       AbstractApplicationContext factory= new GenericXmlApplicationContext("applicationContext.xml");
       TV tv1 = (TV) factory.getBean("tv");
       TV tv2 = (TV) factory.getBean("tv");
       TV tv3 = (TV) factory.getBean("tv");    .....
```

```
< 결과 >
.....
===> SamsungTV 객체 생성
===> SamsungTV 객체 생성
===> SamsungTV 객체 생성
.....
```