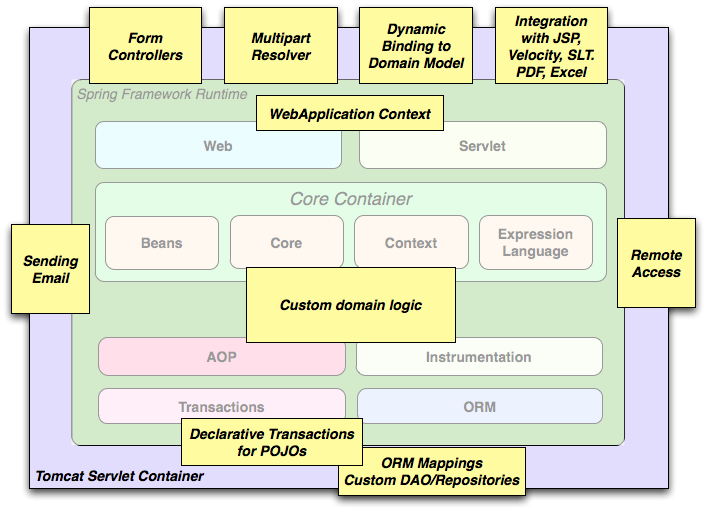
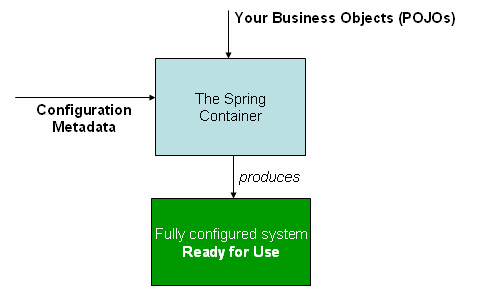
1. SPING
   1. 아키텍처 그림



* 1. 실습

1. IOC
   1. 일반적으로 자바 어플리케이션의 모든 객체의 생성과 의존관계는 프로그래머의 의해서 지정되는 과정이었으나
   2. EJB 컨테이너, 서블릿 컨테이너 같은 것은 이러한 객체의 생성과 소멸 같은 생명 주기를 관리하여 기존의 제어권를 모두 컨테이너들에게 넘긴다 .
   3. 여기서 SPRING IOC 컨테이너는 톰캣 컨테이너 안에서 SPPRING의 IOC CONTAINER 에서 모든 객체를 관리하고 생성하고 소멸하고 심지어는 객체 들간의 의존 관계까지 결정할 수 있는 권한을 위임 받아서 유저의 요청에 따라서 IOC CONTAINER 에 있는 Bean 객체들을 활성화시켜 유저들에게 서비스를 제공한다.
2. DI
   1. 의존성 주입 - 객체들 간의 의존 관계를 IOC 컨테이어에 의해서 의존관계를 주입할 Bean 객체와 주입되는 Bean을 가지고 의존관계를 설정한다.
3. IOC 컨테이너 개념



* 1. Spring Framework에서 Bean은 어플리케이션을 구성하고, IoC container에 의해 관리되어 지는 객체를 의미한다. Bean은 간단히 말해 IoC container에 의해 객체화되고, 조립되고, 또는 관리되는 객체이다. Bean들과 Bean들간의 종속성은 container가 사용하는 설정 메타데이터에 의해 결정된다.
  2. org.springframework.beans.factory.BeanFactory 인터페이스는 Spring IoC Container의 핵심 인터페이스이다. Spring IoC Container는 객체를 생성하고, 객체 간의 종속성을 이어주는 역할을 한다.

1. 실습

|  |
| --- |
|  |
| package ex1;  public class Ex1\_MessegeApp {  private String msg;  // setter는 Property이다.  // 스프링은 외부 조립기에 해당, 외부 조립기는 객체들의 관계를 맞추는 역할을 한다.  // 스프링 컨터이너에서는 싱글톤 레지스토리가 있어서 빈으로 동록한 모든 객체를  // 싱글톤 레지스터리에 등록해놓고 사용한다. 그러면 이 빈은 싱글토으로 관리한다.  public void setMsg(String msg) {  this.msg = msg;  }    // 스프링 컨테이너에 의해서 연결된 레퍼런스로 접근할 메서드  public String printMsg(){  StringBuffer sb = new StringBuffer();  sb.append("당신이 입력한 메세지 ").append("msg").append("입니다.");  return sb.toString();  }    } |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">    <!-- bean 객체를 등록 - Bean 생성과 관리를 모두 스프링 컨테이너에게 위임함 -->  <bean id="msg" class="ex1.Ex1\_MessegeApp">  <property name="msg" value="안녕하세요!" />  </bean>    <bean id="today" class="ex1.Exam1\_Today">  <property name="today" value="오늘의 날짜는?" />  </bean>    </beans> |
| <%@page import="ex1.Ex1\_MessegeApp"%>  <%@page import="org.springframework.context.support.GenericXmlApplicationContext"%>  <%@page import="org.springframework.context.support.GenericApplicationContext"%>  <%@page import="org.springframework.context.ApplicationContext"%>  <%@page import="com.sun.glass.ui.Application"%>  <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>ex1\_di.jsp</title>  </head>  <body>  <%  // Web 환경이 지원되는 컨테이너 - ApplicationContext  ApplicationContext ctx = new GenericXmlApplicationContext("ex1/ex\_1.xml"); // 컨테이너에 빈을 등록하고  // 컨테이너로부터 msg 라고 붙인 빈 객체를 가져온다.  Ex1\_MessegeApp em = ctx.getBean("msg",Ex1\_MessegeApp.class);  String msg = em.printMsg();  %><%=msg %>  </body>  </html> |
|  |

|  |
| --- |
|  |
| package ex2;  // Spring 에서 주입 하는 대상  public class Ex2\_Resource {  private String name;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }    } |
| package ex2;  public class Ex2\_Message {    private Ex2\_Resource resource;  public void setResource(Ex2\_Resource resource) {  this.resource = resource;  }    // Spring으로부터 di 를 받기 위한 메서드  public String msgPrint(){  String name = resource.getName();  return name + "님 안녕하세요";      }  } |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">    <!-- Spirng에 의해서 DI됨 -->  <!-- 주입할 Bean 객체 -->  <bean id="res" class="ex2.Ex2\_Resource">  <property name="name" value="김길동"></property>  </bean>    <!-- 주입받는 객체 res:래퍼런스 싱글톤 레지스토리에서 주소를 가져와서 DI 함 ! -->  <bean id="msg" class="ex2.Ex2\_Message">  <property name="resource" ref="res" />  </bean>    </beans> |
| <%@page import="org.springframework.context.support.GenericXmlApplicationContext"%>  <%@page import="org.springframework.context.support.GenericApplicationContext"%>  <%@page import="org.springframework.context.ApplicationContext"%>  <%@page import="ex2.Ex2\_Message"%>  <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>ex2\_diref.jsp</title>  </head>  <body>  <%  ApplicationContext ctx = new GenericXmlApplicationContext("ex2/ex2\_di.xml");  /\*  ctx.getBean(BeanName,requiredType)  requiredType 은 객체의 실제 클래스의 슈퍼클래스 인터페이스 타입을 찾도록 지정하는 것이다.  requiredType - type the bean must match.  Can be an interface or superclass of the actual class, or null for any match.  For example, if the value is Object.class, this method will succeed whatever the class of the returned instance.  \*/  Ex2\_Message msg = ctx.getBean("msg",Ex2\_Message.class);    %>  <%=msg.msgPrint()%>  </body>  </html> |
|  |

* 1. ㅇ

1. ㅇㅇ
2. ㅇ
3. ㅇ
4. ㅇ
5. ㅇ
6. ㅇ