1. Environment 객체
   1)Environment 객체를 이용해서 스프링 빈 설정을 할 수 있다.
      Context -------------------------> Environment ------------------------------> PropertySources
                  ctx.getEnvironment()   env.getPropertySource()   property 추가 및 추출
                                                                                    추가 : propertySources.addLast()
                                                                                    추출 : env.getProperty()

2. Lab
   1)New > Spring Legacy Project > Simple Projects > Simple Spring Maven
      Project Name : EnvironmentDemo

   2)/src/main/resources/admin.properties 파일 생성

```
admin.id=javaexpert
admin.pwd=12345678
```

   3)/src/main/java/info.javaexpert package 생성

   4)/src/main/java/info.javaexpert.AdminConnection.java 생성

```
package info.javaexpert;

import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.context.EnvironmentAware;
import org.springframework.core.env.Environment;

public class AdminConnection implements EnvironmentAware, InitializingBean, DisposableBean{
   private Environment env;
   private String adminId;
   private String adminPwd;

   public void setEnv(Environment env) {
      this.env = env;
   }

   public void setAdminId(String adminId) {
      this.adminId = adminId;
   }

   public void setAdminPwd(String adminPwd) {
      this.adminPwd = adminPwd;
   }

   public String getAdminId() {
      return adminId;
   }

   public String getAdminPwd() {
      return adminPwd;
   }

   @Override
   public void destroy() throws Exception {
      System.out.println("destroy()");
   }

   @Override
   public void afterPropertiesSet() throws Exception {
      System.out.println("afterPropertiesSet()");
      setAdminId(env.getProperty("admin.id"));
      setAdminPwd(env.getProperty("admin.pwd"));
   }

```

```java
        //bean이 생성되기 전에 callback 으로 호출됨. 가장 먼저 호출됨.
        //MainClass에서 사용하는 env 정보가 넘어옴.
        @Override
        public void setEnvironment(Environment env) {
            System.out.println("setEnvironment()");
            setEnv(env);
        }
    }
```

5)/src/main/resources/beans.xml 생성

```xml
<?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 id="adminConnection" class="info.javaexpert.AdminConnection" />

</beans>
```

6)/src/main/java/MainClass.java 생성

```java
package info.javaexpert;

import java.io.IOException;

import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;
import org.springframework.core.env.ConfigurableEnvironment;
import org.springframework.core.env.MutablePropertySources;
import org.springframework.core.io.support.ResourcePropertySource;

public class MainClass {
    public static void main(String [] args){
        ConfigurableApplicationContext ctx = new GenericXmlApplicationContext();
        ConfigurableEnvironment env = ctx.getEnvironment();

        MutablePropertySources propertySouces = env.getPropertySources();
        //내가 원하는 정보를 얻을 때까지 모든 propertySources를 앞에서 부터 차례로 모두 검색함.
        try{
            propertySouces.addLast(new ResourcePropertySource("classpath:admin.properties"));   //property 추가

            System.out.println(env.getProperty("admin.id"));    //property 추출
            System.out.println(env.getProperty("admin.pwd"));
        }catch(IOException ex){}

        GenericXmlApplicationContext gCtx = (GenericXmlApplicationContext)ctx;
        gCtx.load("beans.xml");
        gCtx.refresh();

        AdminConnection adminConnection = gCtx.getBean("adminConnection", AdminConnection.class);
        System.out.println("admin ID : " + adminConnection.getAdminId());
        System.out.println("admin PWD : " + adminConnection.getAdminPwd());

        gCtx.close();
        ctx.close();
    }
}
```

3. Property 파일을 이용한 설정
   1)환경에 따라 자주 변경되는 내용의 분리
   2)XML의 Bean 설정 메타정보는 어플리케이션 구조가 바뀌지 않으면 자주 변경되지 않는다.

3)반면에 프로퍼티 값으로 제공되는 일부 설정정보(예-DataSource Bean이 사용하는 DB 연결정보)는 어플리케이션이 동작하는 환경(개발, 테스트, 스테이징, 운영)에 따라서 자주 바뀔 수 있다.
4)변경되는 이유와 시점이 다르다면 분리하는 것이 객체지향 설계의 기본 원칙이기에 설정에도 동일한 원칙을 적용할 수 있다.
5)환경에 따라 자주 변경될 수 있는 내용은 properties 파일로 분리하는 것이 가장 깔끔하다.
6)XML 처럼 복잡한 구성이 필요없고 키와 값의 쌍(key=value)으로 구성하면 된다.
7)환경에 따라 자주 변경되는 내용의 분리의 예시
  -value속성에 설정된 값들은 환경에 따라 변경될 수 있는 내용이다.
  -자주 변경되는 값들은 properties 파일에 넣어 분리하는 것이 좋다.

```
<beans.xml>
<bean id="dataSource"
          class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
    <property name="driverClass" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost/testdb" />
    <property name="username" value="spring" />
    <property name="password" value="book" />
</bean>
```

  -properties 파일로 분리한 정보는 ${}(property 치환자)을 이용하여 설정한다.
  -${} 값을 치환해주는 기능은 <context:property-placeholder> 태그에 의해 자동으로 등록되는 PropertyPlaceHolderConfigurer Bean이 담당한다.

```
<database.properties>
  db.driverClass=com.mysql.jdbc.Driver
  db.url=jdbc:mysql://localhost/testdb
  db.username=spring
  db.password=book

<beans.xml>
  <context:property-placeholder
        location="classpath:config/database.properties" />
  <bean id="dataSource"
        class="org.springfremework.jdbc.datasource.SimpleDriverDataSource">
      <property name="driverClass" value="${db.driverClass}" />
      <property name="url" value="${db.url}" />
      <property name="username" value="${db.username}" />
      <property name="password" value="${db.password}" />
  </bean>
```

4. Lab
1)New > Spring Legacy Project > Simple Projects > Simple Spring Maven
    Project Name : PropertyDemo

2)src/main/java > Create Package
  -info.javaexpert

3)POJO class 작성
  -info.javaexpert > right-click > New > Class
  -Class name : Hello

```
<Hello.java>
  package info.javaexpert;

  public class Hello{
    private String name;
    private Printer printer;
    private List<String> names;

    public Hello(){}

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

    public void setPrinter(Printer printer){
```

```
                this.printer = printer;
            }

            public void setNames(List<String> list){
                this.names = list;
            }

            public List<String> getNames(){
                return this.names;
            }

            public String sayHello(){
                return "Hello " + name;
            }

            public void print(){
                this.printer.print(sayHello());
            }
        }
```

-info.javaexpert > right-click > New > Interface
   Interface name : Printer

   <Printer.java>

```
        package info.javaexpert;

        public interface Printer{
            void print(String message);
        }
```

-info.javaexpert > right-click > New > Class
   Class Name : StringPrinter

   <StringPrinter.java>

```
        package info.javaexpert;

        public class StringPrinter implements Printer{
            private StringBuffer buffer = new StringBuffer();

            public void print(String message){
                this.buffer.append(message);
            }

            public String toString(){
                return this.buffer.toString();
            }
        }
```

-info.javaexpert > right-click > New > Class
   Class Name : ConsolePrinter

   <ConsolePrinter.java>

```
        package info.javaexpert;

        public class ConsolePrinter implements Printer{
            public void print(String message){
                System.out.println(message);
            }
        }
```

4)Bean Configuration XML 작성
   -/src/main/resources > right-click > New > Spring Bean Configuration File
      File name : beans.xml > Finish

```
        <?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 id="hello" class="info.javaexpert.Hello">
                <property name="name" value="Spring" />
                <property name="printer" ref="printer" />
                <property name="names">
                    <list>
                        <value>AOP</value>
                        <value>Spring</value>
                        <value>DI</value>
                    </list>
                </property>
            </bean>

            <bean id="printer" class="info.javaexpert.StringPrinter" />
            <bean id="consolePrinter" class="info.javaexpert.ConsolePrinter" />

        </beans>
```

5)DI Test 클래스 작성
    -/src/info.javaexpert > right-click > New > Class
        Class Name : MainClass

```
package info.javaexpert;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
    public static void main(String [] args){
        ApplicationContext ctx = new GenericXmlApplicationContext("classpath:beans.xml");

        Hello hello = (Hello)ctx.getBean("hello");
        System.out.println(hello.sayHello());
        hello.print();

        Printer printer = ctx.getBean("printer", StringPrinter.class);
        System.out.println(printer.toString());

        List<String> list = hello.getNames();
        for(String value : list){
            System.out.println(value);
        }
    }
}
```

6)Test
    -/src/info.javaexpert.test/HelloBeanTest.java > right-click > Run As > Java Application

```
Hello Spring
Hello Spring
AOP
Spring
DI
```

7)jUnit으로 테스트
    -/src/test/java > right-click > New > JUnit Test Case > HelloTest > Finish

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;

import java.util.List;

import org.junit.Test;
```

```java
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import info.javaexpert.Hello;
import info.javaexpert.Printer;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations="classpath:beans.xml")
public class HelloTest {
    @Autowired
    ApplicationContext ctx;

    @Test
    public void test() {
        Hello hello = (Hello)ctx.getBean("hello");
        assertEquals("Hello Spring", hello.sayHello());
        hello.print();

        Printer printer = (Printer)ctx.getBean("printer");
        assertEquals("Hello Spring", printer.toString());
    }

    @Test
    public void test2(){
        Hello hello = (Hello)ctx.getBean("hello");

        Hello hello2 = ctx.getBean("hello", Hello.class);
        assertSame(hello, hello2);

        assertEquals(3, hello2.getNames().size());
    }
}
```

-right-click > Run As > Junit Test
-결과 -> Junit View에 초록색 bar

8)/src/main/resources/value.properties 생성
<value.properties>

```
myname=Spring
myprinter=printer
value1=HTML5
value2=CSS3
value3=JavaScript
```

9)/src/config/beans.xml 에서 [Namespaces] tab
-목록에서 'context-http://www.springframework.org/schema/context' check
-<context:property-placeholder />를 사용하기 위해서

<beans.xml>

```xml
<context:property-placeholder
        location="classpath:value.properties" />

<bean id="hello" class="info.javaexpert.Hello">
    <property name="name" value="${myname}" />
    <property name="printer" ref="${myprinter}" />
    <property name="names">
        <list>
            <value>${value1}</value>
            <value>${value2}</value>
            <value>${value3}</value>
        </list>
    </property>
</bean>
```

10)Test

-/src/main/java/info.javaexpert.MainClass.java

--right-click > Run As > Java Application

```
Hello Spring
Hello Spring
HTML5
CSS3
JavaScript
```

-/src/test/java/HelloTest.java

--right-click > Run As > JUnit Test

--Green Bar

11)/src/main/resources/value.properties 수정

<value.properties>

```
myname=Spring
myprinter=printer
value1=JUnit
value2=AOP
value3=DI
printer1=stringPrinter
printer2=consolePrinter
```

12)Hello.java 코드 수정

-/src/info.javaexpert/Hello.java

```
package info.javaexpert;

import java.util.List;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import javax.annotation.Resource;

@Component("hello")
public class Hello {
    @Value("${myname}")
    private String name;

    @Resource(name="${printer1}")
    private Printer printer;

    @Value("${value1}, ${value2}, ${value3}")
    private List<String> names;

    public List<String> getNames(){
        return names;
    }
    public String sayHello(){
        return "Hello " + name;
    }

    public void print(){
        this.printer.print(sayHello());
    }
}
```

13)StringPrinter.java 수정

```
package info.javaexpert;

import org.springframework.stereotype.Component;

@Component("stringPrinter")
public class StringPrinter implements Printer {
    private StringBuffer buffer =  new StringBuffer();

```

```
            @Override
            public void print(String message) {
                this.buffer.append(message);
            }

            @Override
            public String toString(){
                return this.buffer.toString();
            }
        }
```

14)beans.xml 수정하기

```
        <?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:context="http://www.springframework.org/schema/context"
            xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-3.2.xsd">

            <context:property-placeholder location="classpath:value.properties "/>
            <context:component-scan base-package="info.javaexpert" />
        </beans>
```

15)MainClass.java 수정하기

```
        package info.javaexpert;

        import java.util.List;

        import org.springframework.context.ApplicationContext;
        import org.springframework.context.support.GenericXmlApplicationContext;

        public class MainClass {
            public static void main(String [] args){
                ApplicationContext ctx = new GenericXmlApplicationContext("classpath:beans.xml");

                Hello hello = (Hello)ctx.getBean("hello");
                System.out.println(hello.sayHello());
                hello.print();

                Printer printer = ctx.getBean("stringPrinter", StringPrinter.class);
                System.out.println(printer.toString());

                List<String> list = hello.getNames();
                for(String value : list){
                    System.out.println(value);
                }
            }
        }
```

5. Lab

ApplicationContext.xml --> XML 파일을 이용하는 방법
-Spring 설정 XML 파일에 property 파일에 대해 명시한다.
-admin.properties
-sub_admin.properties

ApplicationConfig --> Java 파일을 이용하는 방법
-Spring 설정 Java 파일에 property 파일을 명시한다.
-admin.properties
-sub_admin.properties

1)/src/main/java/info.javaexper package 생성
-/src/main/java > right-click > New > Package

-Package name : info.javaexpert

2)/src/main/java/info.javaexpert.AdminConnection.java 생성

```
<AdminConnection.java>
   package info.javaexpert;

   public class AdminConnection {
      private String adminId;
      private String adminPwd;
      private String subAdminId;
      private String subAdminPwd;

      public String getAdminId() {
         return adminId;
      }

      public void setAdminId(String adminId) {
         this.adminId = adminId;
      }

      public String getAdminPwd() {
         return adminPwd;
      }

      public void setAdminPwd(String adminPwd) {
         this.adminPwd = adminPwd;
      }

      public String getSubAdminId() {
         return subAdminId;
      }

      public void setSubAdminId(String subAdminId) {
         this.subAdminId = subAdminId;
      }

      public String getSubAdminPwd() {
         return subAdminPwd;
      }

      public void setSubAdminPwd(String subAdminPwd) {
         this.subAdminPwd = subAdminPwd;
      }
   }
```

3)/src/main/resources 두 개의 properties 파일 생성

```
<admin.properties>
   admin.id=javaexpert
   admin.pwd=12345678

<sub.admin.properties>
   sub.admin.id=javasoft
   sub.admin.pwd=987654321
```

4)/src/main/resources/beans.xml 생성

```
<beans.xml>
   <?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:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context-3.2.xsd">
```

```
        <context:property-placeholder location="classpath:admin.properties,
        classpath:sub.admin.properties" />

        <bean id="adminConnection" class="info.javaexpert.AdminConnection">
            <property name="adminId">
                <value>${admin.id}</value>
            </property>
            <property name="adminPwd">
                <value>${admin.pwd}</value>
            </property>
            <property name="subAdminId">
                <value>${sub.admin.id}</value>
            </property>
            <property name="subAdminPwd">
                <value>${sub.admin.pwd}</value>
            </property>
        </bean>
    </beans>
```

5)/src/main/java/MainClass.java 생성

<MainClass.java>

```
package info.javaexpert;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
    public static void main(String [] args){
        AbstractApplicationContext ctx =
                new GenericXmlApplicationContext("classpath:beans.xml");
        AdminConnection connection = ctx.getBean("adminConnection", AdminConnection.class);
        System.out.println("admin ID : " + connection.getAdminId());
        System.out.println("admin PWD : " + connection.getAdminPwd());
        System.out.println("sub admin ID : " + connection.getSubAdminId());
        System.out.println("sub admin PWD : " + connection.getSubAdminPwd());

        ctx.close();
    }
}
```

6. Lab

1)/src/main/resources 두 개의 properties 파일 생성

<admin.properties>

```
admin.id=javaexpert
admin.pwd=12345678
```

<sub.admin.properties>

```
sub.admin.id=javasoft
sub.admin.pwd=987654321
```

2)/src/main/java/info.javaexpert.ApplicationConfig.java>

<ApplicationConfig.java>

```
package info.javaexpert;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

@Configuration
public class ApplicationConfig {
        @Value("${admin.id}")
        private String adminId;
        @Value("${admin.pwd}")
        private String adminPwd;
        @Value("${sub.admin.id}")
        private String subAdminId;
        @Value("${sub.admin.pwd}")
        private String subAdminPwd;

        @Bean
        public static PropertySourcesPlaceholderConfigurer Properties(){
            PropertySourcesPlaceholderConfigurer configurer =
                    new PropertySourcesPlaceholderConfigurer();

            Resource [] locations = new Resource[2];
            locations[0] = new ClassPathResource("admin.properties");
            locations[1] = new ClassPathResource("sub.admin.properties");
            configurer.setLocations(locations);

            return configurer;
        }

    @Bean
    public AdminConnection adminConfig(){
        AdminConnection adminConnection = new AdminConnection();
        adminConnection.setAdminId(adminId);
        adminConnection.setAdminPwd(adminPwd);
        adminConnection.setSubAdminId(subAdminId);
        adminConnection.setSubAdminPwd(subAdminPwd);
        return adminConnection;
    }
}
```

3)/src/main/java/info.javaexpert.MainClass1.java

```
<MainClass1.java>
    package info.javaexpert;

    import org.springframework.context.annotation.AnnotationConfigApplicationContext;

    public class MainClass1 {
        public static void main(String[] args) {
            AnnotationConfigApplicationContext ctx =
                    new AnnotationConfigApplicationContext(ApplicationConfig.class);
            AdminConnection conn = ctx.getBean("adminConfig", AdminConnection.class);

            System.out.println("admin ID : " + conn.getAdminId());
            System.out.println("admin PWD : " + conn.getAdminPwd());
            System.out.println("sub admin ID : " + conn.getSubAdminId());
            System.out.println("sub admin PWD : " + conn.getSubAdminPwd());
        }
    }
```

7. Profile 속성을 이용한 설정
    -동일한 Spring Bean을 여러개 만들어 놓고 상황(환경)에 따라서 적절한 스프링 빈을 사용할 수 있다.
    -profile 속성을 사용한다.
    -역시 Java 파일을 이용하는 방법과 XML 설정 파일을 이용하는 방법이 있다.

8. Lab
    1)Package 생성
        -/src/main/java > right-click > New > Package
        -Package name : info.javaexpert

    2)XML 설정 파일 2개 생성
        -/src/main/resource > right-click > New > Spring Bean Configuration File

-File name : run.xml

```
<?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"
   profile="run">    <---이것이 핵심

   <bean id="serverInfo" class="info.javaexpert.ServerInfo">
      <property name="ipNum" value="192.168.56.5" />
      <property name="portNum" value="80" />
   </bean>
</beans>
```

-/src/main/resource > right-click > New > Spring Bean Configuration File
-File name : dev.xml

```
<?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"
   profile="dev">    <---이것이 핵심

   <bean id="serverInfo" class="info.javaexpert.ServerInfo">
      <property name="ipNum" value="192.168.56.5" />
      <property name="portNum" value="80" />
   </bean>
</beans>
```

3)ServerInfo.java 생성
-/src/main/java/info.javaexpert.ServerInfo.java

```
package info.javaexpert;

public class ServerInfo {
   private String ipNum;
   private String portNum;
   public String getIpNum() {
      return ipNum;
   }
   public void setIpNum(String ipNum) {
      this.ipNum = ipNum;
   }
   public String getPortNum() {
      return portNum;
   }
   public void setPortNum(String portNum) {
      this.portNum = portNum;
   }
}
```

4)MainClass 생성
-/src/main/java/info.javaexpert.MainClass.java

```
package info.javaexpert;
import java.util.Scanner;

import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      System.out.print("Select dev or run : ");
      String config = scan.next();  //"dev" or "run"

```

```
                GenericXmlApplicationContext ctx = new GenericXmlApplicationContext();
                ctx.getEnvironment().setActiveProfiles(config);
                ctx.load("dev.xml", "run.xml");

                ServerInfo info = ctx.getBean("serverInfo", ServerInfo.class);
                System.out.println("IP : " + info.getIpNum());
                System.out.println("Port : " + info.getPortNum());
                ctx.close();
            }
        }
```

5)결과

-입력시 dev를 넣으면 dev환경인 localhost/8080이 나오고, 만일 run이라고 넣으면 192.168.56.5/80이 나온다.

9. Lab

1)Package 생성

-/src/main/java > right-click > New > Package

-Package name : info.javaexpert

2)Java 설정 파일 2개 생성

-/src/main/java > right-click > New > Class

-Class name : ApplicationConfigDev

```
package info.javaexpert;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;

@Configuration
@Profile("dev")
public class ApplicationConfigDev {

    @Bean
    public ServerInfo serverInfo(){
        ServerInfo info = new ServerInfo();
        info.setIpNum("localhost");
        info.setPortNum("8080");
        return info;
    }
}
```

-/src/main/java > right-click > New > Class

-Class name : ApplicationConfigRun

```
package info.javaexpert;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;

@Configuration
@Profile("run")
public class ApplicationConfigRun {

    @Bean
    public ServerInfo serverInfo(){
        ServerInfo info = new ServerInfo();
        info.setIpNum("192.168.56.5");
        info.setPortNum("80");
        return info;
    }
}
```

3)ServerInfo.java 생성

-/src/main/java/info.javaexpert.ServerInfo.java

```java
package info.javaexpert;

public class ServerInfo {
    private String ipNum;
    private String portNum;
    public String getIpNum() {
        return ipNum;
    }
    public void setIpNum(String ipNum) {
        this.ipNum = ipNum;
    }
    public String getPortNum() {
        return portNum;
    }
    public void setPortNum(String portNum) {
        this.portNum = portNum;
    }
}
```

4)MainClass 생성

-/src/main/java/info.javaexpert.MainClass.java

```java
package info.javaexpert;
import java.util.Scanner;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Select dev or run : ");
        String config = scan.next();  //"dev" or "run"

        AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext();
        ctx.getEnvironment().setActiveProfiles(config);
        ctx.register(ApplicationConfigDev.class, ApplicationConfigRun.class);
        ctx.refresh();

        ServerInfo info = ctx.getBean("serverInfo", ServerInfo.class);
        System.out.println("IP : " + info.getIpNum());
        System.out.println("Port : " + info.getPortNum());
        ctx.close();
    }
}
```

5)결과

-입력시 dev를 넣으면 dev환경인 localhost/8080이 나오고, 만일 run이라고 넣으면 192.168.56.5/80이 나온다.