Dependency injection - Java

Contents

[Dependency Injection Demo 1](#_Toc40356076)

[Description: 1](#_Toc40356077)

[Bean Factory 1](#_Toc40356078)

[Bean Injection 2](#_Toc40356079)

[Bean scopes 2](#_Toc40356080)

[Limitations 2](#_Toc40356081)

# Dependency Injection Demo

## Description:

This project shows the demonstration of beans creation and injection dynamically. This project focus on the creation and the design principle inversion of control. This is written in Java without using any frameworks other than Junit.

## Bean Factory

The bean factory introduced in this project prepares the beans required for IOC(Inversion of control) . The expectation of this factory is to load the beans initially when the application starts. After loading the beans on a specific scope it manages the creation of beans. The JVM takes care of garbage collection, I recommend to use the option “-XX:+UseParallelGC” for garbage collection time to time.

The bean factory instance can be obtained by implementing the interface “BeanFactoryAware”, The bean factory is injected at runtime for every instance of the classes implementing this interface.

Beans can be obtained by the method getBeanByClassName by passing the class name.

## Bean Injection

The bean injection is done at the runtime, this is done with the help of bean factory. The bean factory prepares itself when the application starts and then loads all the beans as per the definition in the xml file (BeanDefinitions.xml) found at the class path.

## Bean scopes

The bean scope defined in this application are of two types “singleton” and “prototype”. The singleton scoped beans are the beans where only one bean is created for a particular class through out the life cycle of the application.

Prototype scoped beans are the beans where each instance of a class is created when ever the bean factory is requested for a new bean.

## Limitations

* The BeanFactory will have maintain one instance of object for a specified class name.
* Naming the bean with a “alias” name is not done.
* Bean instance can be obtained by passing the class name, so one unique class name is expected to get the bean.
* Declaration of beans and classes to be defined in the BeanDefinitions.xml file.
* XML schema for the bean definitions is not done.
* BeanDefinitions.xml file is to be kept in the class path.
* Bean scopes are “singleton” and “prototype”
* Autowiring is not supported.

Tests

The Junit test written in this application demonstrates the dependency injection technique followed in this application. Refer the ApplicationTest class for more details on using this application.