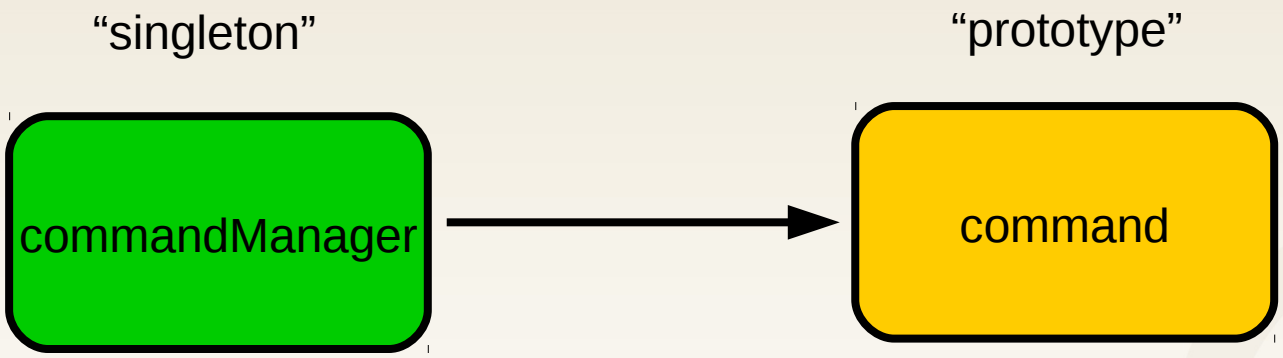
Singleton ve Prototype Arasındaki Bağımlılıklar

Singleton ve Prototype Nesneler Arasındaki Bağımlılıklar

- Singleton bean prototype bean'a bağımlı olabilir
- Bağımlılık **instantiation zamanı**nda çözülür
- Dolayısı ile singleton bean'ın ömrü boyunca **aynı prototype bean** kullanılır
- Başka bir ifade ile prototype bean'de singleton gibi davranmış olur

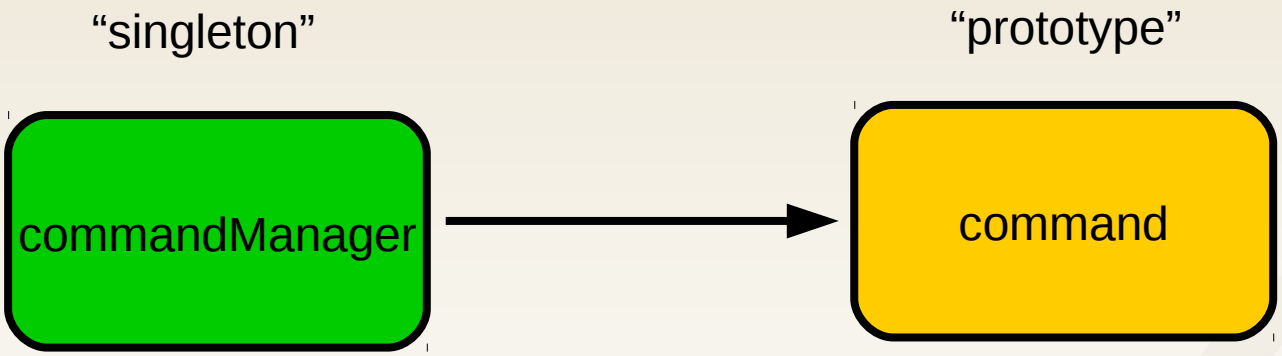
Singleton ve Prototype Nesneler Arasındaki Bağımlılıklar




```
executeCommand(Object state) {
        command.setState(state);
        command.execute();
}
```

Bazı durumlarda singleton bean içerisindeki her metot invokasyonunda prototype scope bean'den yeni bir instance'ın kullanılması gerekebilir

Singleton ve Prototype Nesneler Arasındaki Bağımlılıklar




```
executeCommand(Object state) {
        Command command = applicationContext
                                    .getBean(Command.class);
        command.setState(state);
        command.execute();
}
```

Böyle bir durumda prototype scope bean'i singleton scope bean'e doğrudan enjekte etmek yerine, çalışma zamanında her metot invokasyonunda prototype bean'e ApplicationContext ile lookup yaparak erişmek gerekir

Singleton ve Prototype Nesneler Arasındaki Bağımlılıklar

```
public class CommandManager implements ApplicationContextAware {

    private ApplicationContext applicationContext;

    @Override
    public void setApplicationContext(ApplicationContext appContext) {
        this.applicationContext = appContext;
     }

    public void executeCommand(Object state) {
        Command command = applicationContext.getBean(Command.class);
        command.setState(state);
        command.execute();
    }
}
```

CommandManager sınıfı **ApplicationContextAware** arayüzünü implement ettiği için bean yaratılırken **ApplicationContext** kendisini bean'a enjekte eder. Çalışma anında da **getBean()** metodu aracılığı ile farklı command instance'ları elde edebilir

ApplicationContextAware Arayüzü

- ApplicationContext'e bean içerisinden erişebilmek için **ApplicationContextAware** arayüzünü implement etmek gerekir
- Spring Container, ApplicationContextAware arayüzünü implement eden bean'lere **kendisini enjekte** eder
- Bu sayede bean içerisinden ApplicationContext içerisindeki bean'lara **programatik erişim** mümkün hale gelir

Singleton ve Prototype Nesneler Arasındaki Bağımlılıklar

```
<beans...>
   <bean id="commandManager"
         class="x.y.CommandManager" scope="singleton"/>


   <bean id="command"
         class="x.y.Command" scope="prototype"/>
</beans>
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com