6-5 使用Inject注入

@Inject 標註注入相依物件架構

@Inject為JSR 330標準，使用javax.inject.Inject

預設為採用類型配置(by Type)，如需要使用 B Name配置直接透過@Named

適用在Attribute 或setter 或建構子上注入

使用@Inject注入的Bean定義，不能使用兩個Method進行Bean定義

設定每次請求產生一個個體物件的Employees Bean

使用@Bean進行配置

使用@Scope配置每次請求產生一個個體物件

```
@Scope(ConfigurableBeanFactory.SCOPE_SINGLETON)
public Company company() {
    Company com=new Company();
    com.setCompanyName("Tibame");
    com.setAddress("台北市");
    com.setPhone("02-12345678");
    return com;
}
//產生一個個體員工物件
@Bean
@Scope(ConfigurableBeanFactory.SCOPE_PROTOTYPE)
public Employees employees(Company com) {
    Employees emp=new Employees();
    emp.setCompany(com);
    return emp;
}
```

Employees JavaBean個體隨機編號

每一個個體產生的員工編號產生編號

驗證注入控制項的每一個個體物件

```
package com.tibame.domain;
//Employees JavaBean
public class Employees implements java.io.Serializable{
    public Employees() {
        this.id=(int)(Math.random()*10000)+1+"";
    }
    private String id;
    private String name;
    private Company company; //員工歸屬一個公司(DI)
    public String getId() {
        return id;
    }
```

@Inject 注入元件於Controller

因為使用javax.inject

- 需要再pom.xml配置依賴相關的物件模組

控制項中使用@Inject進行注入

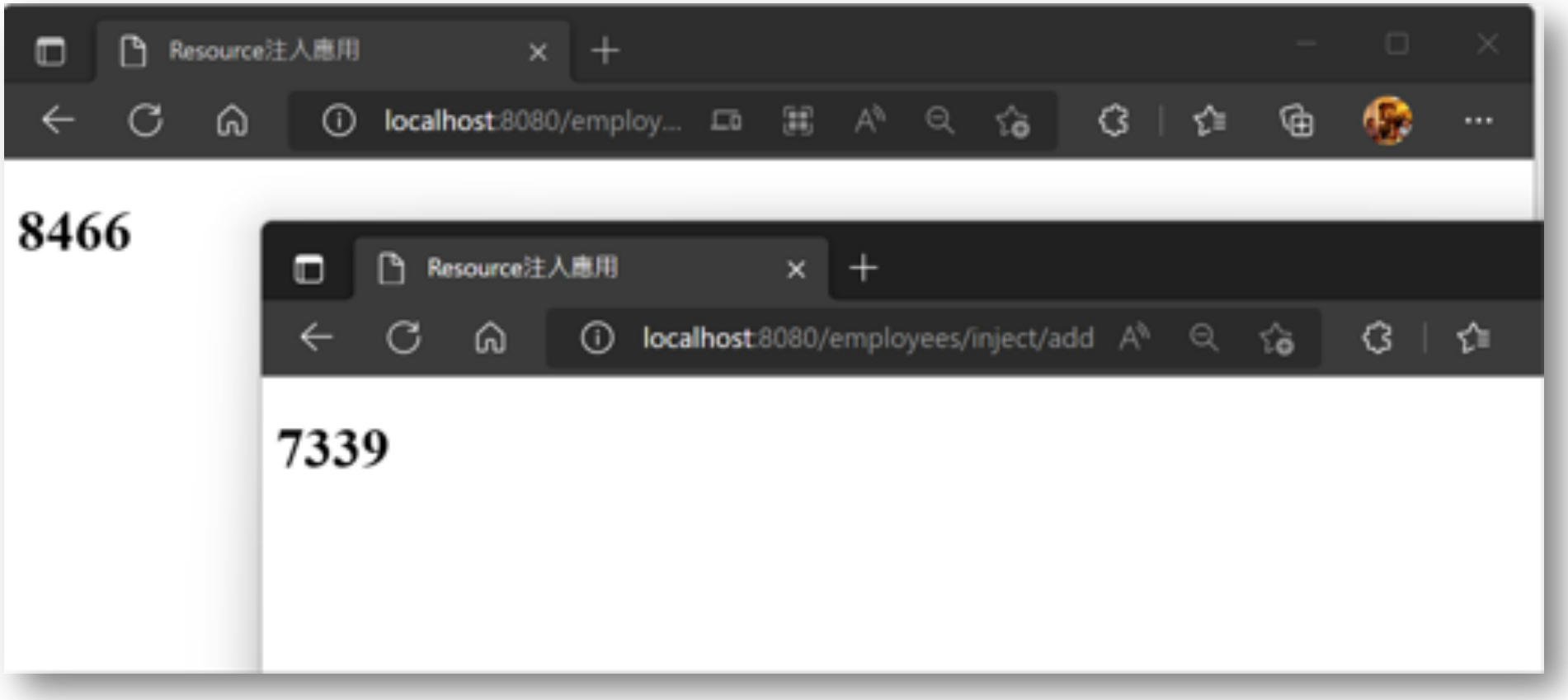
參數參數注入，表現每次請求為一個相對的個體因應

```
<!-- https://mvnrepository.com/artifact/ja
<dependency>
    <groupId>javax.inject</groupId>
    <artifactId>javax.inject</artifactId>
    <version>1</version>
</dependency>
```

```
//@Inject注入應用
@Controller
public class InjectController {
    //Attribute
    //使用參數注入
    @GetMapping(path="/employees/inject/add")
    @Inject
    public String customersAdd(Employees employees) {
        //隱含參考注入的Customers物件到View Page
        return "employeesadd";
    }

}
```

@Inject 注入元件於Controller執行結果



總結：6-5 使用Inject注入

了解@Inject注入依賴物件架構之後，我們來了解spring Bean產生的順序架構