Wednesday, January 5, 2022 10:34 AM

Wiring --------- connecting

One bean must be connected to another bean
Why to connect ? One bean HAS-A dependency of another-bean

Bean object creation task is OFFLOADED to spring context ,
Spring context must find the bean and inject it } wiring !!!

AutoWiring ------
We ask the spring context
Find the bean for the class and inject it !!!!!

Autowiring by TYPE ----------

@Autowired
private ContactDetails contactDetails;

Spring context finds a bean having this TYPE , if yes then that bean reference is injected in the contactDetails property !!!!

Factory = to produce and give objects
Configurable
1. Singleton -- always return
2. Return a new object for each request
3. Create a pool of objects and return the object from pool whenever requested
4. ……

javax.sql.DataSource = DB Connection Factory ----- it will produce connections (usually in a pool)
We will use Spring Framework to Inject a DataSource in our code !!!

Interface javax.sql.DataSource (JDBC)
|
|
Class org.springframework.DriverManagerDataSource implements DataSource !!! Given by spring framework

SO the above Impl class is a READY MADE bean class
We will configure it and use it in our class

```
@Component
Class BookDAOBean          ( MessageBean)
{
        @Autowired
        DataSource  ds;          (ContactDetails )

        Public void addBook(int id, int cost, String name)
        {
                Connection  con = ds.getConnection();
                PreparedStatement …..
```

```
            …
            …
        }

}
```

```
    <bean id = "dataSource"
class = "org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name = "driverClassName" value = "com.mysql.cj.jdbc.Driver"/>
  <property name = "url" value = "jdbc:mysql://localhost:3306/bookshop"/>
  <property name = "username" value = "root"/>
  <property name = "password" value = "123456789"/>
</bean>
```

_____

FLOW1 ----container flow
Container reads beans.xml
|
|
Create object of DriverManagerDataSource Bean
|
|--inject the connectivity properties given in xml
|
Create object of BookDAO
|
|-----inject DS
|
WAIT ------------------------

FLOW2 ----------execute flow
|
|
Call getBean
|
|----return already created BookDAO bean
Call addBean method
|
|
Use ds , get con , get pstmt , fire query

_____

Scenario 1 = I added Datasource bean and the DAO bean in **beans.xml**
ClasspathxmlAppcontext

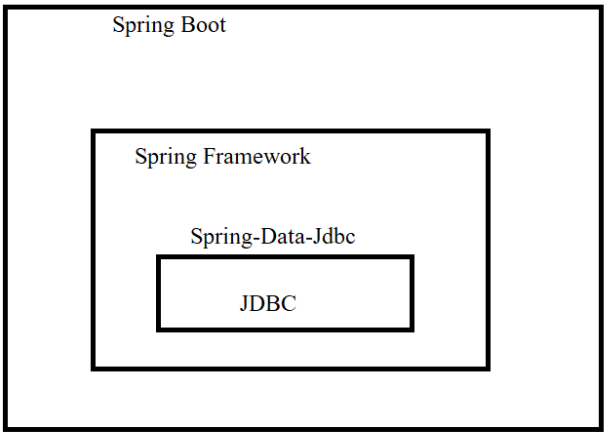
Scenario2 = I KEEP datasource bean in **beans.xml** and I shift DAO bean configuration to **Java Config**
AnnotationConfigAppContext ----------java config class
|
|
IMPORT RESOURCE --- we included the xml ---
xml beans are processed by container
|
|

We created the BOOKDAO
|
|injection of DS

Scenario 3 = I KEEP datasource bean in **beans.xml** and instead of using Java Config for BOOKDAO we use **annotation config .**

Scenario 4 = I use **Java config** for data source bean and **annotation config** for BookDAO

Spring Boot !!



Spring Boot --- **by default sets a lot of configurations(AutoConfigurations)** that may be needed by spring framework
Project development Speed increases !!!

Component Architecture

Container ----------component } Integration due to JEE standards , Loose coupling

Container wants any that implements Servlet interface or extends HttpServlet

Any servlet that wants to be integrated must implement the Servlet !!! Loose coupling

---

```
class Library
{
    Private ArrayList<Book>  books;  // tight coupled
}


class Library
{
    Private List<Book>  books;   //loose coupled  --- I may pass ArrayList or LinkedList
```

```
}


class Library
{
        Private Collection<Book>  books;   //MORE loose coupled   --I may pass ArrayList or LinkedList
        or TreeSet,HashSet
}
```

Web Services ---- REST API ----------

Web application	Web services
Uses http	Uses http
Web server(tomcat)	Web server (tomcat)
Web client(browser)	browser , postman , core java program, servlet
Http methods --- GET and POST	Http methods ----- GET ,POST, PUT ,DELETE
OUTPUT is VIEW(UI) (html,css)	OUTPUT is DATA (xml , json ,plain text ,....)

Service = some code that does a functionality and gives some output data !!!
Service is always on the server side , also runs on server side .
It can be requested using a URL !!!

REST ful WEB SERVICE = **RE**presentational **S**tate **T**ransfer

Client calls a REMOTE method (is on server)
Method runs REMOTELY (on server)
Method returns a value REMOTELY (on server)
Value is given to client   LOCALLY (on client)

Remote Procedure Call / RMI Remote Method Invocation

Remote means call and execution are not in same JVM /process

Client call ----------->Server
<<<----------method return value---

Client may be in CPP, Java , python
|
|
CALL must be translated to a LANGUAGE NUETRAL format
HTTP methods can be used to send the call
|
|
Server may be in Java, PHP , .net

Server ----------Return value must be in language neutral format
|
XML /JSON/plain txt/RSS feed, sent as HTTP response

Uniform Method Call = Http Method

REST service

Variety of Data Representations that are language/platform neutral = XML,JSON,Plain

---

HW for Afternoon ---- try all the scenarios discussed in class

Add getAllBooks method in the BookDAO bean , call the method

HW ---- run the same example using SpringBoot as discussed in class

---