Trainer: Nilesh Ghule

<nilesh@sunbeaminfo.com>

Maven

pom - Project Object Model

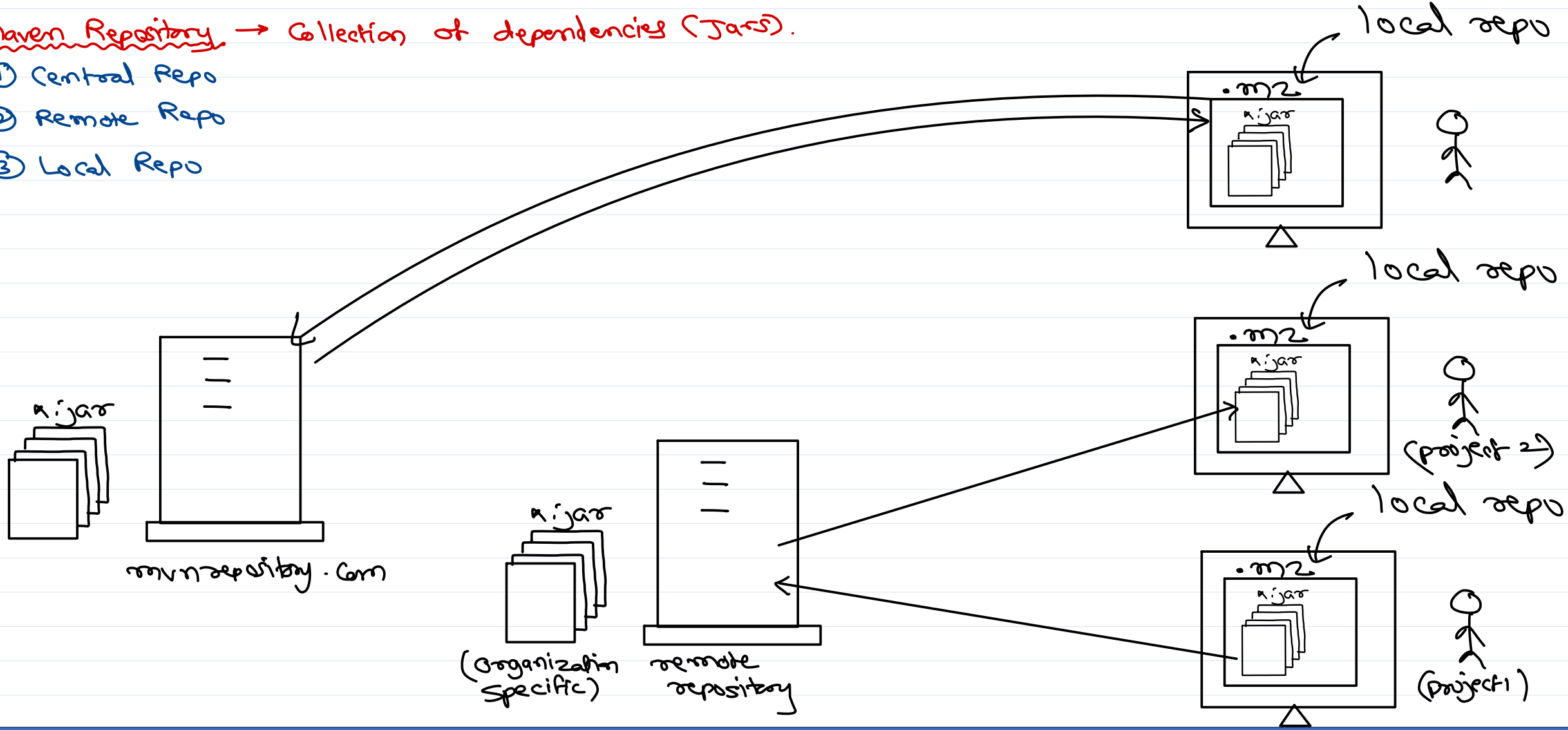
- Maven is Java Build Tool.
- Maven does following:
  - Download dependencies from central/remote repository into local repository.
  - Compile source code.
  - Package compiled code into JAR/WAR files.
  - Install the packaged code.
- POM.XML – Heart of Maven
  - Configurations, Dependencies, Build plugins.
- Maven build life cycles: default or clean
- Maven build phases: validate, compile, test, package, install, deploy

- Details: http://tutorials.jenkov.com/maven/maven-tutorial.html

Mac | Linux : $HOME/.m2
Windows : C:/Users/user/.m2

Maven Repository → Collection of dependencies (Jars).

1. Central Repo
2. Remote Repo
3. Local Repo

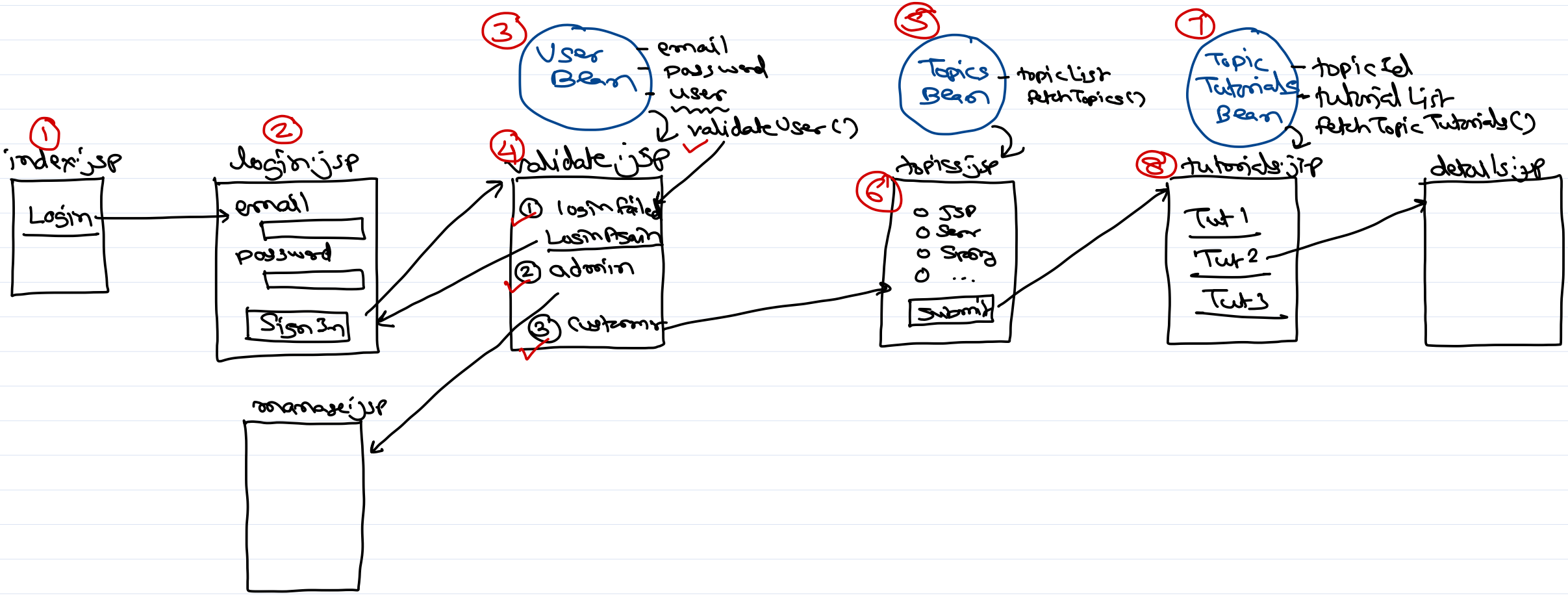


Maven Build Life Cycle

1. default
   - app build process
2. clean
   - delete all generated files (bin, .jar, .class)

App Build Process / Steps

1. validate (check pom.xml) (download dependency if not present in local repo)
2. compile (add dep in classpath & compile all src code files).
3. test (run all unit test cases).
4. package (package all output files into jar or war).
5. install (copy jar/war into local repo).
6. deploy (copy jar/war into server).


1
index.jsp
Login
2
login.jsp
email
password
Sign In
3
User Bean
email
password
user
validateUser()
4
validate.jsp
1 login failed
LoginFail
2 admin
3 customer
manage.jsp
5
Topics Bean
topicList
fetchTopics()
topics.jsp
JSP
Servlet
Spring
...
Submit
7
Topic Tutorials Bean
topicId
tutorialList
fetchTopicTutorials()
8
tutorials.jsp
Tut 1
Tut 2
Tut 3
details.jsp

Spring Introduction

Spring Framework

- Spring is **light-weight comprehensive** framework to **simplify** Java development.
- Developed by Rod Johnson. Spring 3 added annotations while Spring 5 added reactive.
- Spring pros
  - Inversion of control (Dependency injection)
  - Test driven development
  - Extensive but Flexible and modular
  - Smooth integration with existing technologies
  - Eliminate boilerplate code
  - Extendable
  - Maintainable
- Spring cons
  - Heavy configuration (XML, Java, Both)
  - Module versioning & compatibility
  - Application deployment

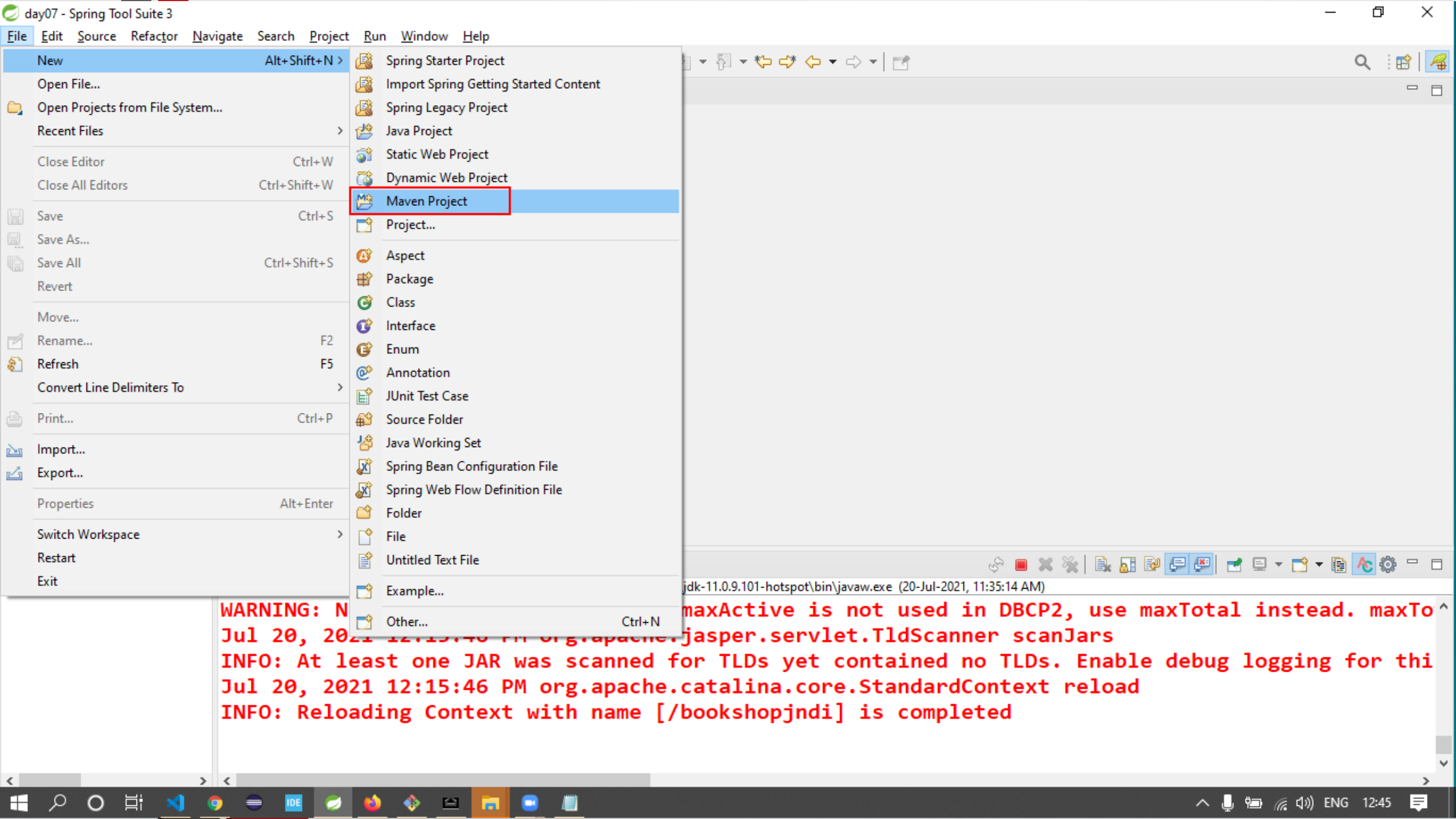
Thank you!

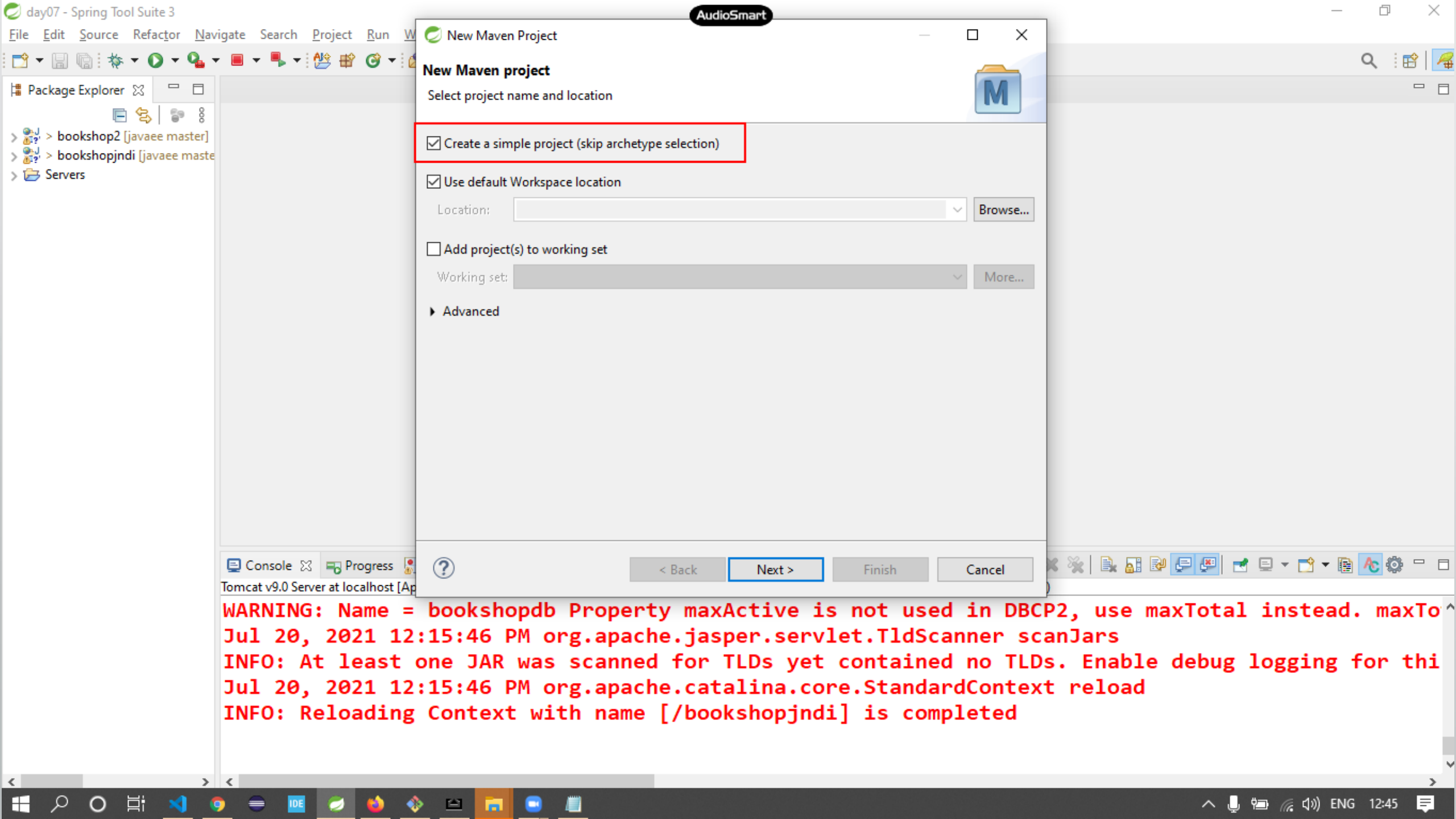
Nilesh Ghule <nilesh@sunbeaminfo.com>

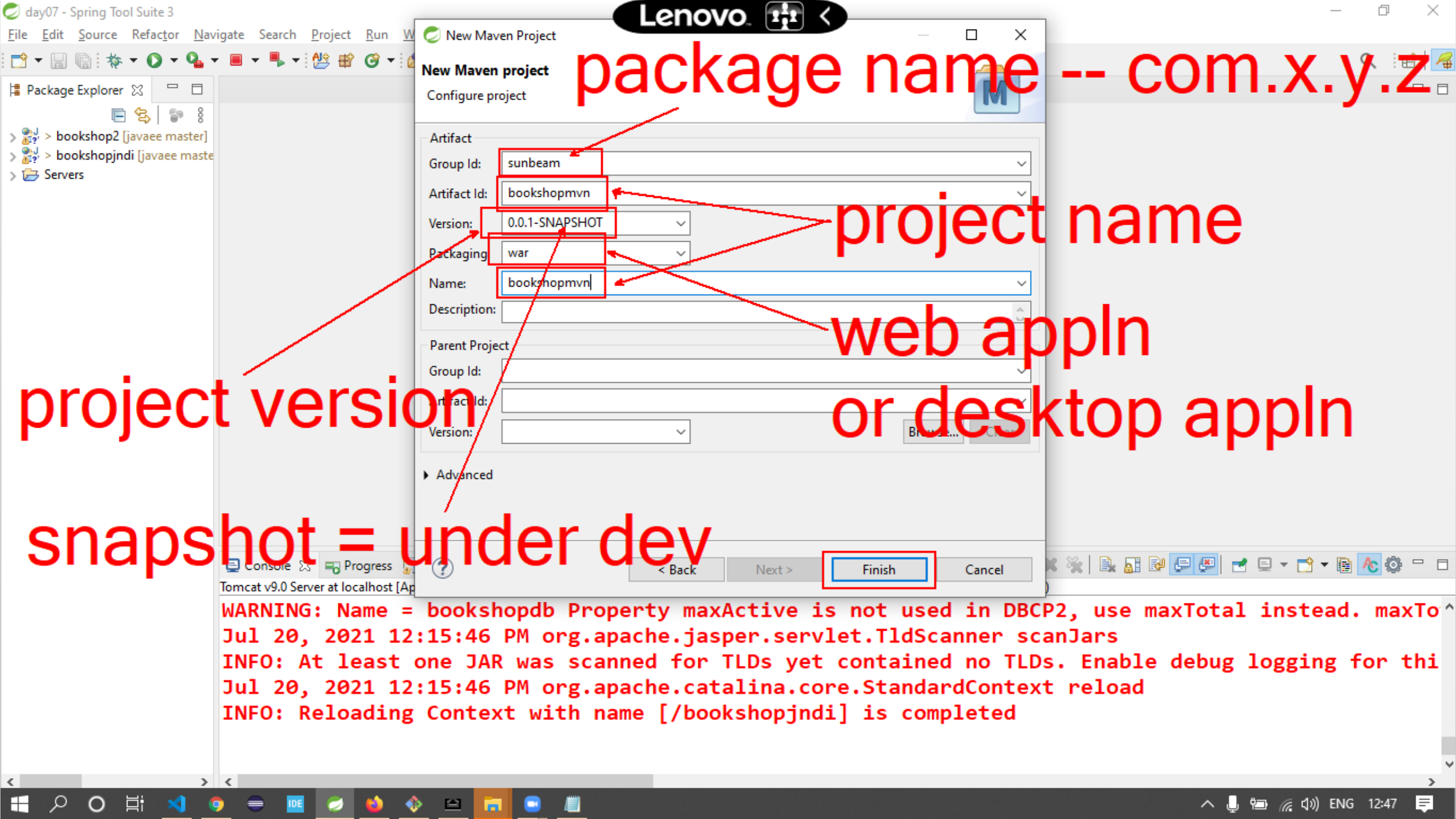
Java Web Application using Maven

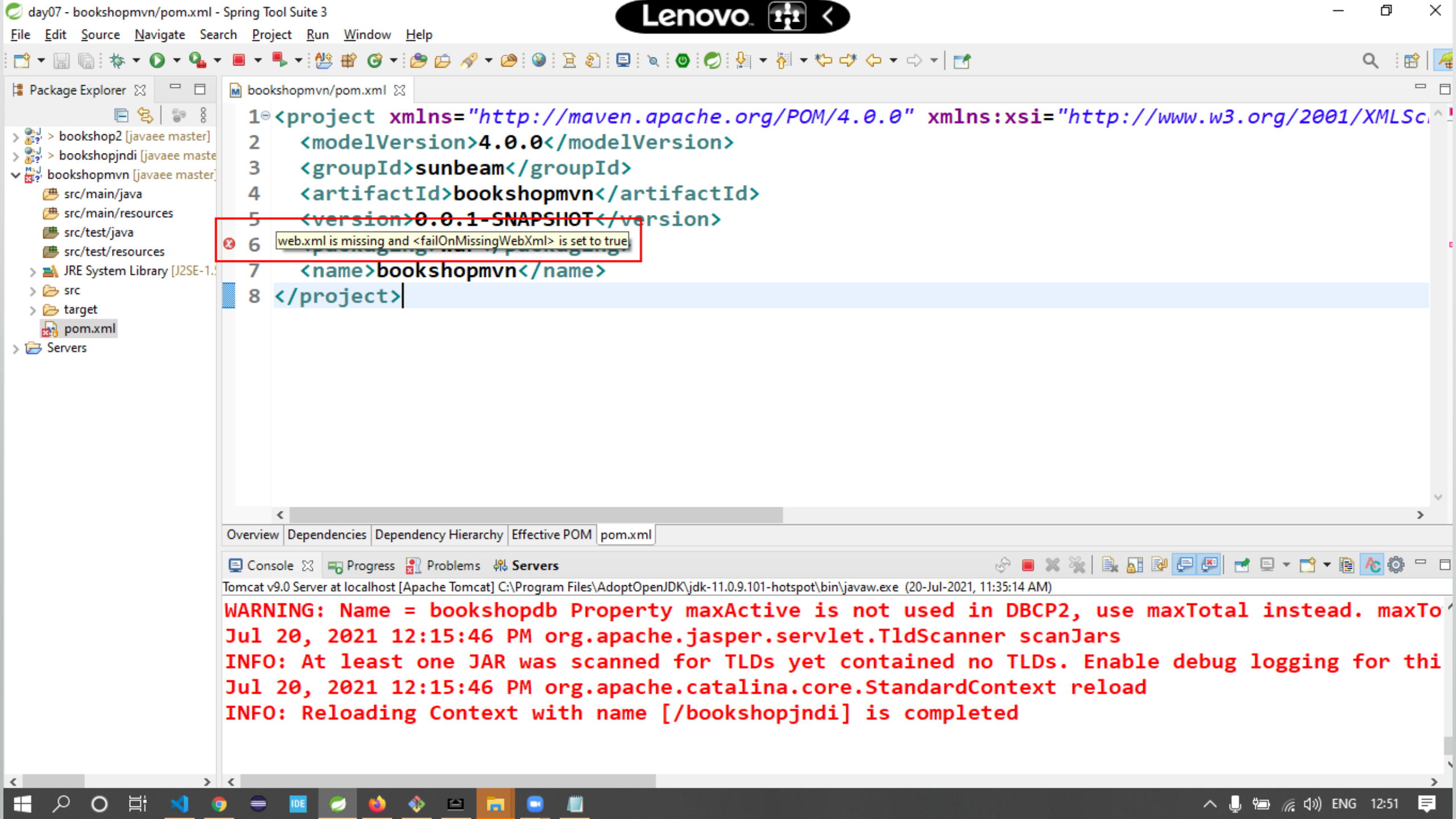
Nilesh Ghule @ Sunbeam Infotech

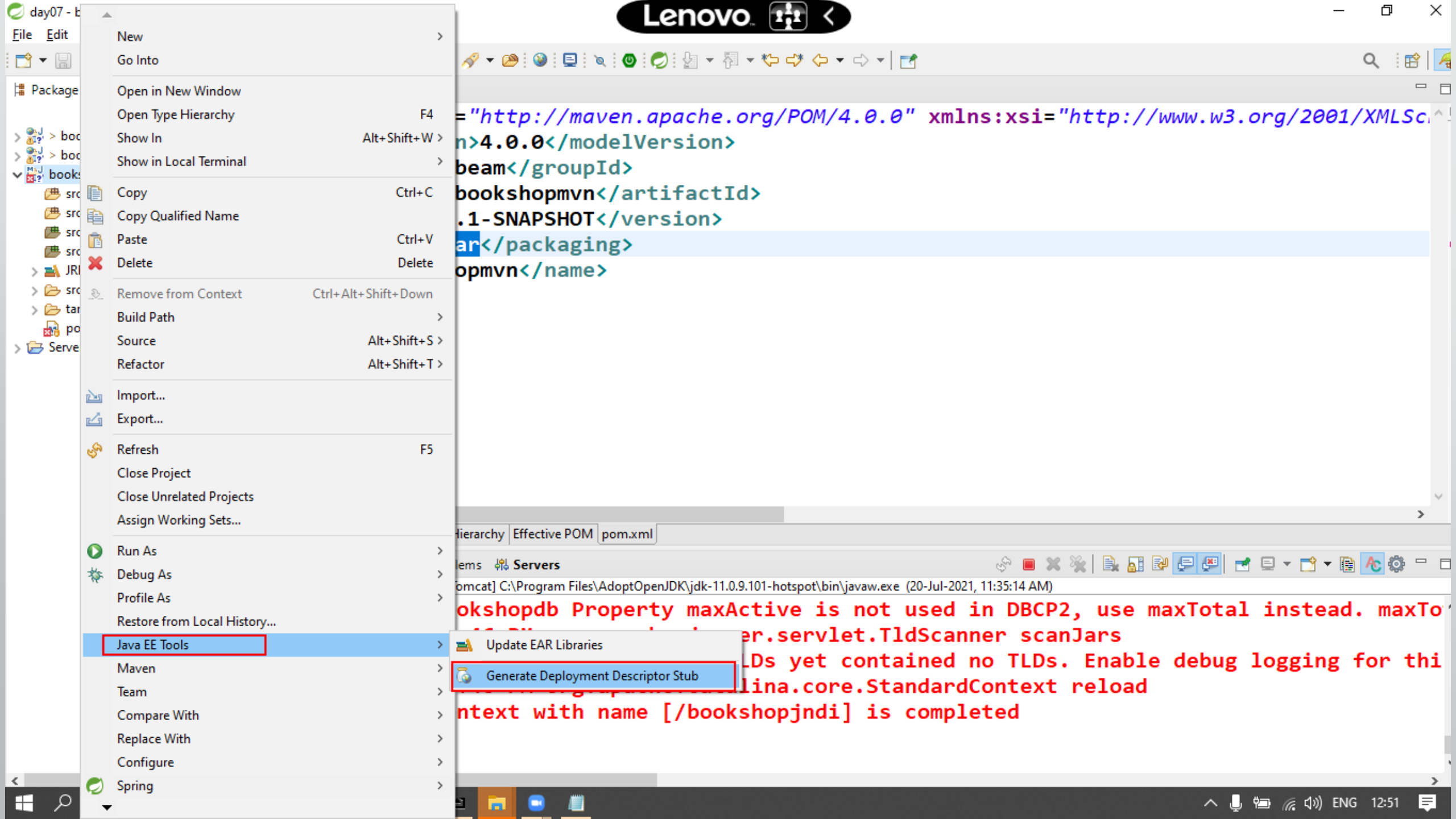
Develop Maven web application in Eclipse

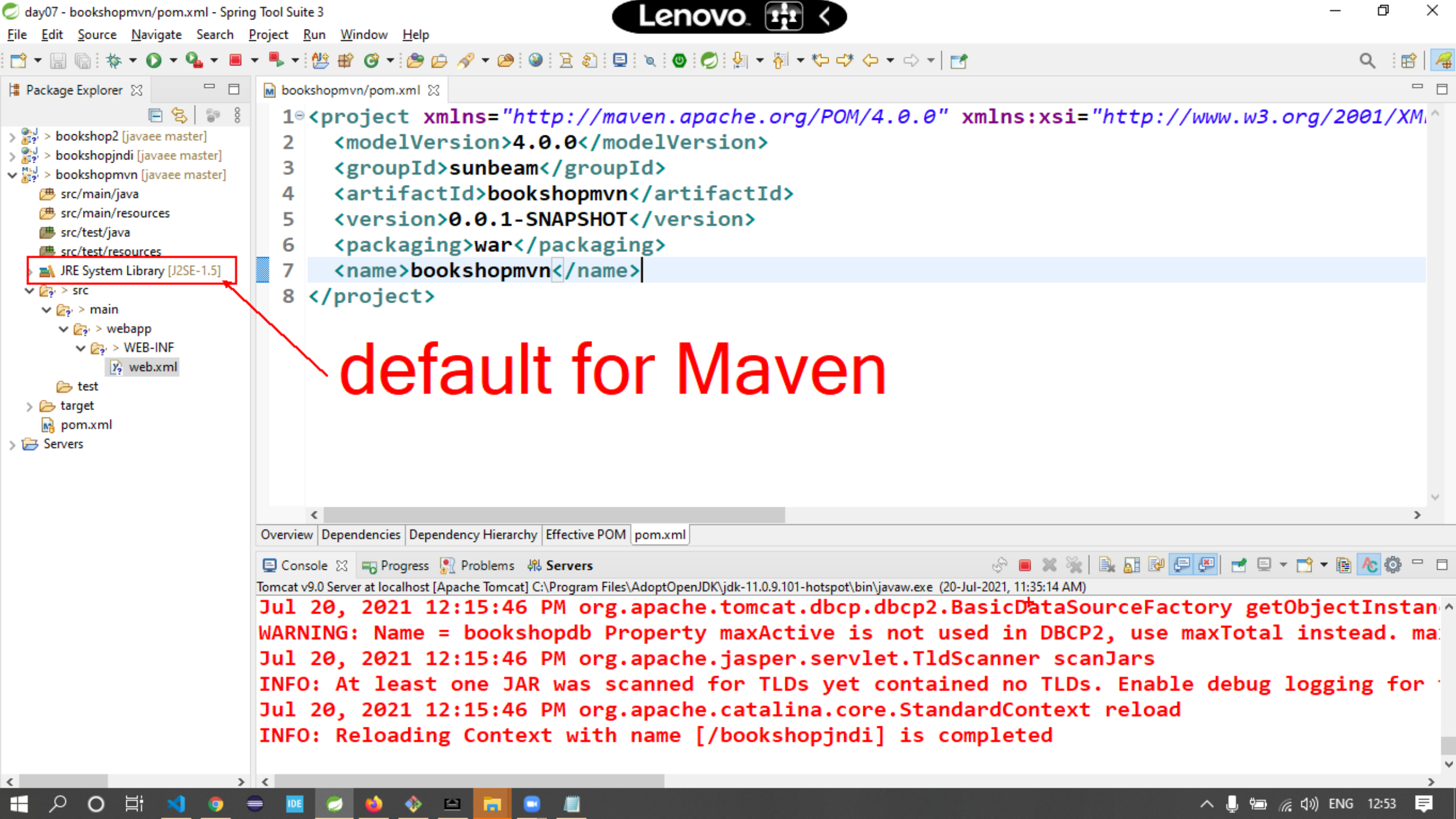

day07 - Spring Tool Suite 3
File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Spring Starter Project
Import Spring Getting Started Content
Spring Legacy Project
Java Project
Static Web Project
Dynamic Web Project
Maven Project
Project...
Aspect
Package
Class
Interface
Enum
Annotation
JUnit Test Case
Source Folder
Java Working Set
Spring Bean Configuration File
Spring Web Flow Definition File
Folder
File
Untitled Text File
Example...
Other... Ctrl+N
jdk-11.0.9.101-hotspot\bin\javaw.exe (20-Jul-2021, 11:35:14 AM)
WARNING: N
maxActive is not used in DBCP2, use maxTotal instead. maxTo
Jul 20, 202
jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for thi
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed
ENG 12:45

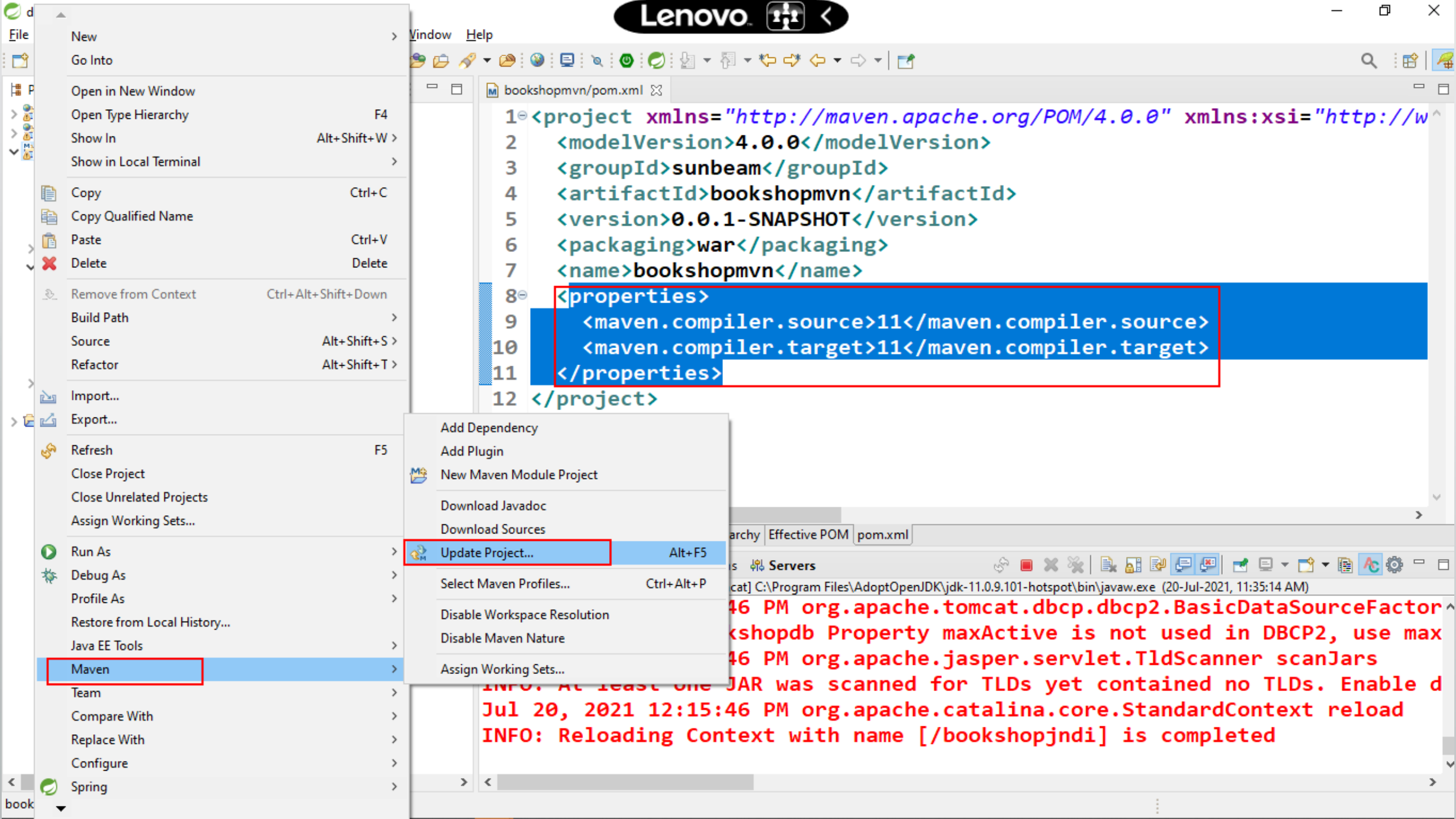

day07 - Spring Tool Suite 3
AudioSmart
File Edit Source Refactor Navigate Search Project Run
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee maste
Servers
New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel
Console
Progress
Tomcat v9.0 Server at localhost [Ap
WARNING: Name = bookshopdb Property maxActive is not used in DBCP2, use maxTotal instead. maxTo
Jul 20, 2021 12:15:46 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for thi
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed
ENG
12:45

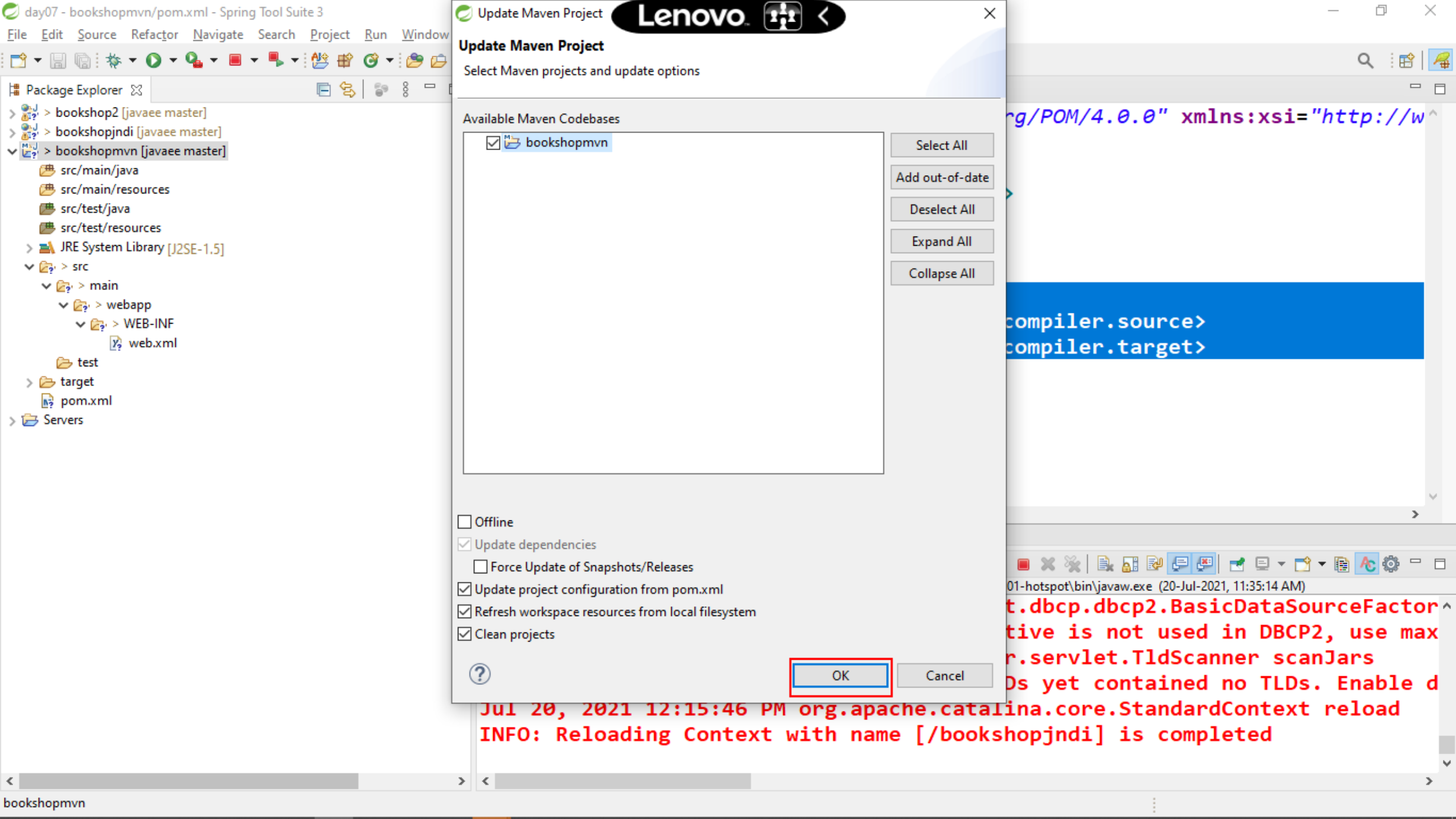

day07 - Spring Tool Suite 3
File Edit Source Refactor Navigate Search Project Run
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee maste
Servers
New Maven Project
New Maven project
Configure project
package name -- com.x.y.z
Artifact
Group Id: sunbeam
Artifact Id: bookshopmvn
Version: 0.0.1-SNAPSHOT
Packaging: war
Name: bookshopmvn
Description:
project name
web appln
or desktop appln
Parent Project
Group Id:
Version:
Advanced
project version
snapshot = under dev
< Back
Next >
Finish
Cancel
Console
Progress
Tomcat v9.0 Server at localhost [Ap
WARNING: Name = bookshopdb Property maxActive is not used in DBCP2, use maxTotal instead. maxTo
Jul 20, 2021 12:15:46 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for thi
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed
ENG 12:47

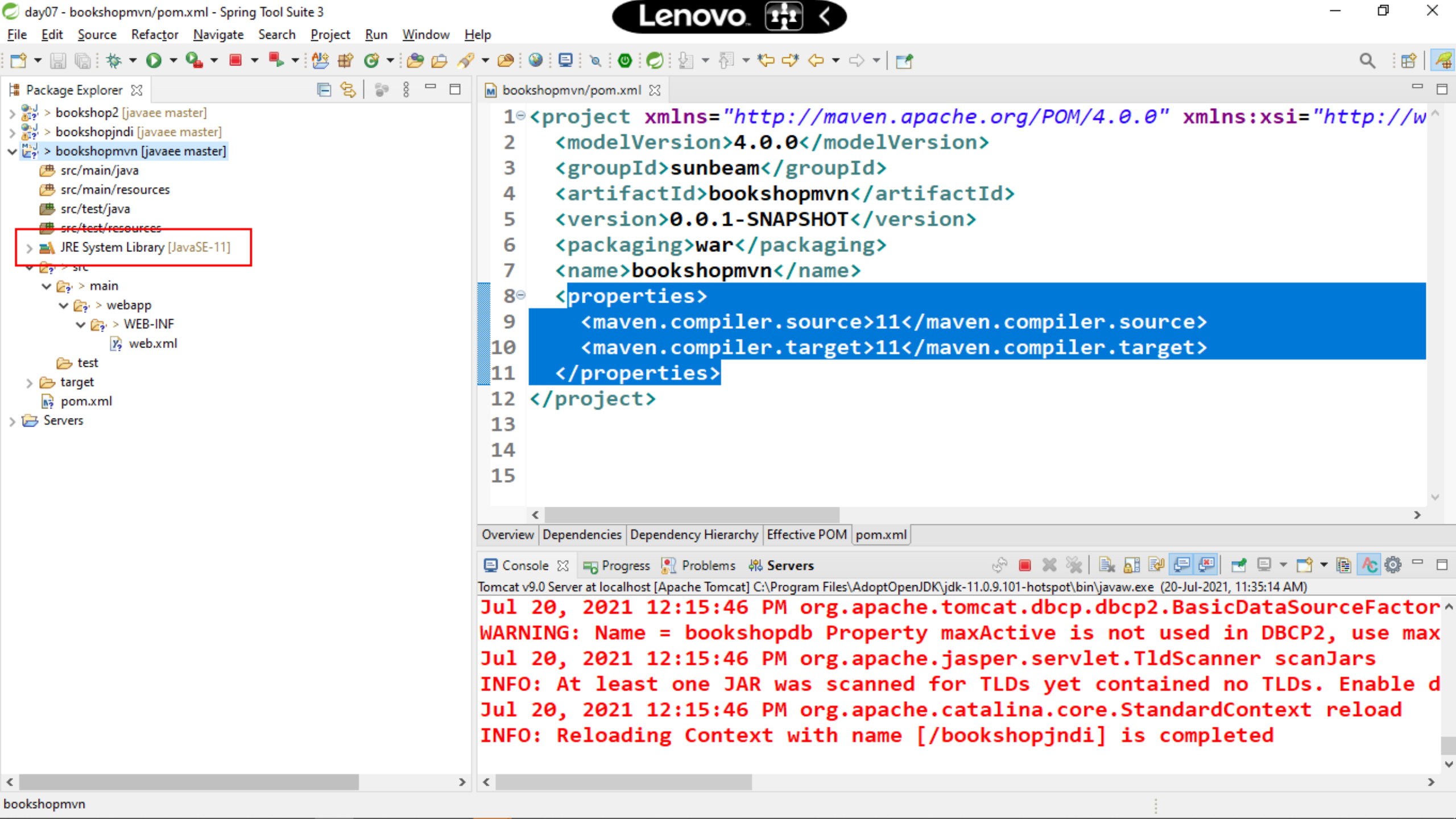

day07 - bookshopmvn/pom.xml - Spring Tool Suite 3
File Edit Source Navigate Search Project Run Window Help
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee maste
bookshopmvn [javaee master]
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.
src
target
pom.xml
Servers
bookshopmvn/pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSc
2 <modelVersion>4.0.0</modelVersion>
3 <groupId>sunbeam</groupId>
4 <artifactId>bookshopmvn</artifactId>
5 <version>0.0.1-SNAPSHOT</version>
6 web.xml is missing and <failOnMissingWebXml> is set to true
7 <name>bookshopmvn</name>
8 </project>
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Console Progress Problems Servers
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\javaw.exe (20-Jul-2021, 11:35:14 AM)
WARNING: Name = bookshopdb Property maxActive is not used in DBCP2, use maxTotal instead. maxTo
Jul 20, 2021 12:15:46 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for thi
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed
ENG 12:51


File Edit
Package
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Profile As
Restore from Local History...
Java EE Tools
Maven
Team
Compare With
Replace With
Configure
Spring
Update EAR Libraries
Generate Deployment Descriptor Stub
="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSc
n>4.0.0</modelVersion>
beam</groupId>
bookshopmvn</artifactId>
.1-SNAPSHOT</version>
ar</packaging>
opmvn</name>
Hierarchy Effective POM pom.xml
Servers
omcat] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\javaw.exe (20-Jul-2021, 11:35:14 AM)
okshopdb Property maxActive is not used in DBCP2, use maxTotal instead. maxTo
er.servlet.TldScanner scanJars
LDs yet contained no TLDs. Enable debug logging for thi
lina.core.StandardContext reload
ntext with name [/bookshopjndi] is completed
ENG 12:51


day07 - bookshopmvn/pom.xml - Spring Tool Suite 3
File Edit Source Navigate Search Project Run Window Help
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee master]
bookshopmvn [javaee master]
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
src
main
webapp
WEB-INF
web.xml
test
target
pom.xml
Servers
bookshopmvn/pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
2 <modelVersion>4.0.0</modelVersion>
3 <groupId>sunbeam</groupId>
4 <artifactId>bookshopmvn</artifactId>
5 <version>0.0.1-SNAPSHOT</version>
6 <packaging>war</packaging>
7 <name>bookshopmvn</name>
8 </project>
default for Maven
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Console Progress Problems Servers
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\javaw.exe (20-Jul-2021, 11:35:14 AM)
Jul 20, 2021 12:15:46 PM org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory getObjectInstan
WARNING: Name = bookshopdb Property maxActive is not used in DBCP2, use maxTotal instead. ma
Jul 20, 2021 12:15:46 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed
ENG 12:53


File
Window
Help
bookshopmvn/pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://w
<modelVersion>4.0.0</modelVersion>
<groupId>sunbeam</groupId>
<artifactId>bookshopmvn</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
<name>bookshopmvn</name>
<properties>
<maven.compiler.source>11</maven.compiler.source>
<maven.compiler.target>11</maven.compiler.target>
</properties>
</project>
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Profile As
Restore from Local History...
Java EE Tools
Maven
Team
Compare With
Replace With
Configure
Spring
Add Dependency
Add Plugin
New Maven Module Project
Download Javadoc
Download Sources
Update Project... Alt+F5
Select Maven Profiles... Ctrl+Alt+P
Disable Workspace Resolution
Disable Maven Nature
Assign Working Sets...
Effective POM
pom.xml
Servers
cat] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\javaw.exe (20-Jul-2021, 11:35:14 AM)
46 PM org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactor
kshopdb Property maxActive is not used in DBCP2, use max
46 PM org.apache.jasper.servlet.TldScanner scanJars
JAR was scanned for TLDs yet contained no TLDs. Enable d
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed


day07 - bookshopmvn/pom.xml - Spring Tool Suite 3
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee master]
bookshopmvn [javaee master]
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
src
main
webapp
WEB-INF
web.xml
test
target
pom.xml
Servers
Update Maven Project
Lenovo
Update Maven Project
Select Maven projects and update options
Available Maven Codebases
bookshopmvn
Select All
Add out-of-date
Deselect All
Expand All
Collapse All
Offline
Update dependencies
Force Update of Snapshots/Releases
Update project configuration from pom.xml
Refresh workspace resources from local filesystem
Clean projects
OK
Cancel
g/POM/4.0.0" xmlns:xsi="http://w
compiler.source>
compiler.target>
01-hotspot\bin\javaw.exe (20-Jul-2021, 11:35:14 AM)
t.dbcp.dbcp2.BasicDataSourceFactor
tive is not used in DBCP2, use max
r.servlet.TldScanner scanJars
Ds yet contained no TLDs. Enable d
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed
bookshopmvn


day07 - bookshopmvn/pom.xml - Spring Tool Suite 3
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee master]
bookshopmvn [javaee master]
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
src
main
webapp
WEB-INF
web.xml
test
target
pom.xml
Servers
bookshopmvn/pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://w
2 <modelVersion>4.0.0</modelVersion>
3 <groupId>sunbeam</groupId>
4 <artifactId>bookshopmvn</artifactId>
5 <version>0.0.1-SNAPSHOT</version>
6 <packaging>war</packaging>
7 <name>bookshopmvn</name>
8 <properties>
9 <maven.compiler.source>11</maven.compiler.source>
10 <maven.compiler.target>11</maven.compiler.target>
11 </properties>
12 </project>
13
14
15
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Console Progress Problems Servers
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\AdoptOpenJDK\jdk-11.0.9.101-hotspot\bin\javaw.exe (20-Jul-2021, 11:35:14 AM)
Jul 20, 2021 12:15:46 PM org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactor
WARNING: Name = bookshopdb Property maxActive is not used in DBCP2, use max
Jul 20, 2021 12:15:46 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable d
Jul 20, 2021 12:15:46 PM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/bookshopjndi] is completed
bookshopmvn

Maven pom.xml

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.27</version>
</dependency>

<dependency>
  <groupId>javax</groupId>
  <artifactId>javaee-api</artifactId>
  <version>8.0</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
  <scope>provided</scope>
</dependency>
```

day07 - bookshopmvn/src/main/webapp/WEB-INF/web.xml - Spring Tool Suite 3

File Edit Source Refactor Navigate Search Project Run Design Window Help

Package Explorer

- bookshop2 [javaee master]
- bookshopjndi [javaee master]
- bookshopmvn [javaee master]
  - src/main/java
    - sunbeam.beans
    - sunbeam.controllers
    - sunbeam.dao
    - sunbeam.pojo
    - sunbeam.util
    - jdbc.properties
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [JavaSE-11]
  - Maven Dependencies
  - src
    - main
      - webapp
        - WEB-INF
          - web.xml
        - addcart.jsp
        - auth.jsp
        - books.jsp
        - doregister.jsp
        - login.jsp
        - logout.jsp
        - register.jsp
        - showcart.jsp
        - subjects2.jsp
      - test
  - target
    - m2e-wtp
  - pom.xml
- Servers

bookshopmvn/pom.xml | web.xml

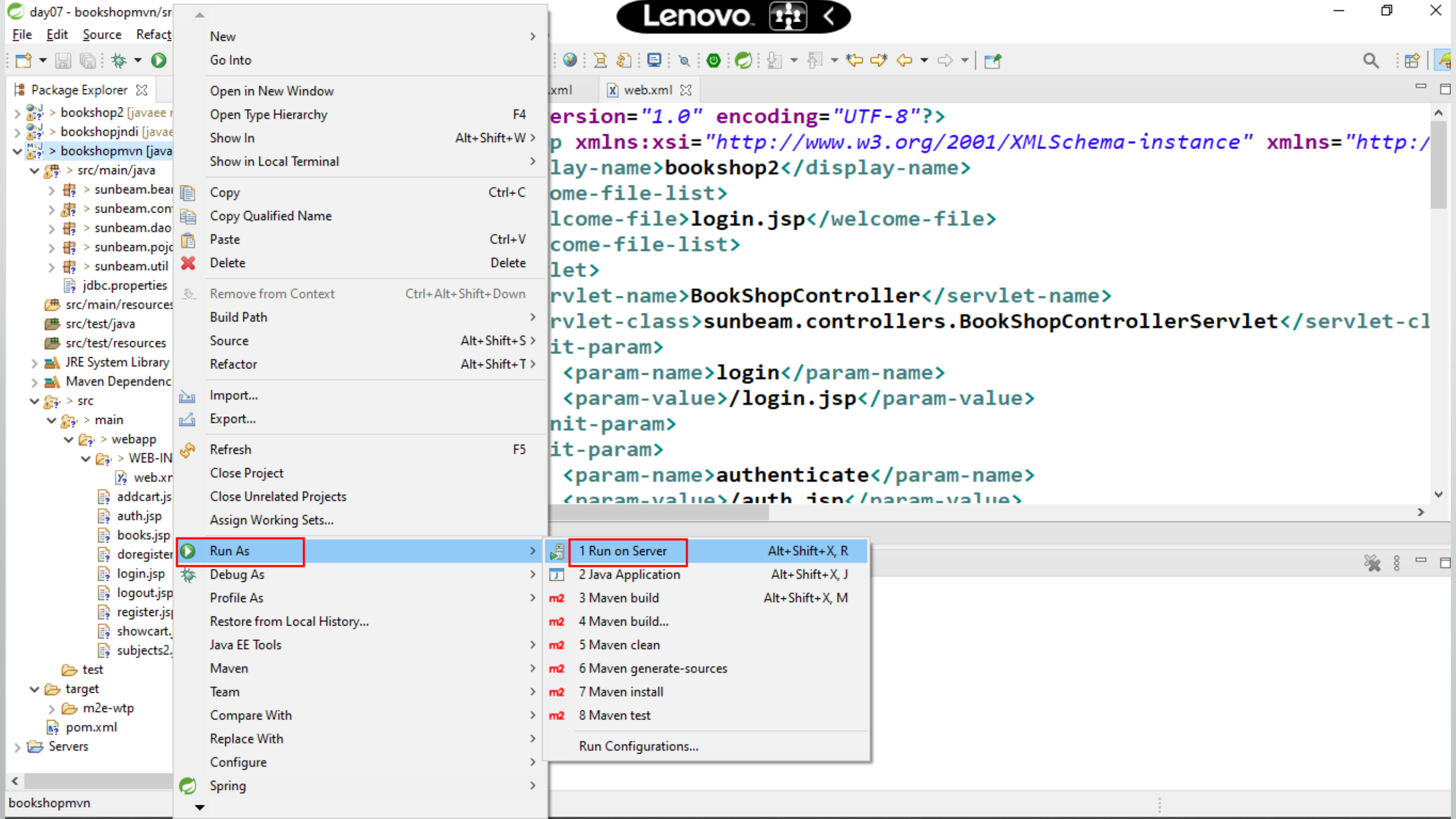
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:/
3    <display-name>bookshop2</display-name>
4    <welcome-file-list>
5      <welcome-file>login.jsp</welcome-file>
6    </welcome-file-list>
7    <servlet>
8      <servlet-name>BookShopController</servlet-name>
9      <servlet-class>sunbeam.controllers.BookShopControllerServlet</servlet-cl
10     <init-param>
11         <param-name>login</param-name>
12         <param-value>/login.jsp</param-value>
13     </init-param>
14     <init-param>
15         <param-name>authenticate</param-name>
16         <param-value>/auth.jsp</param-value>
```

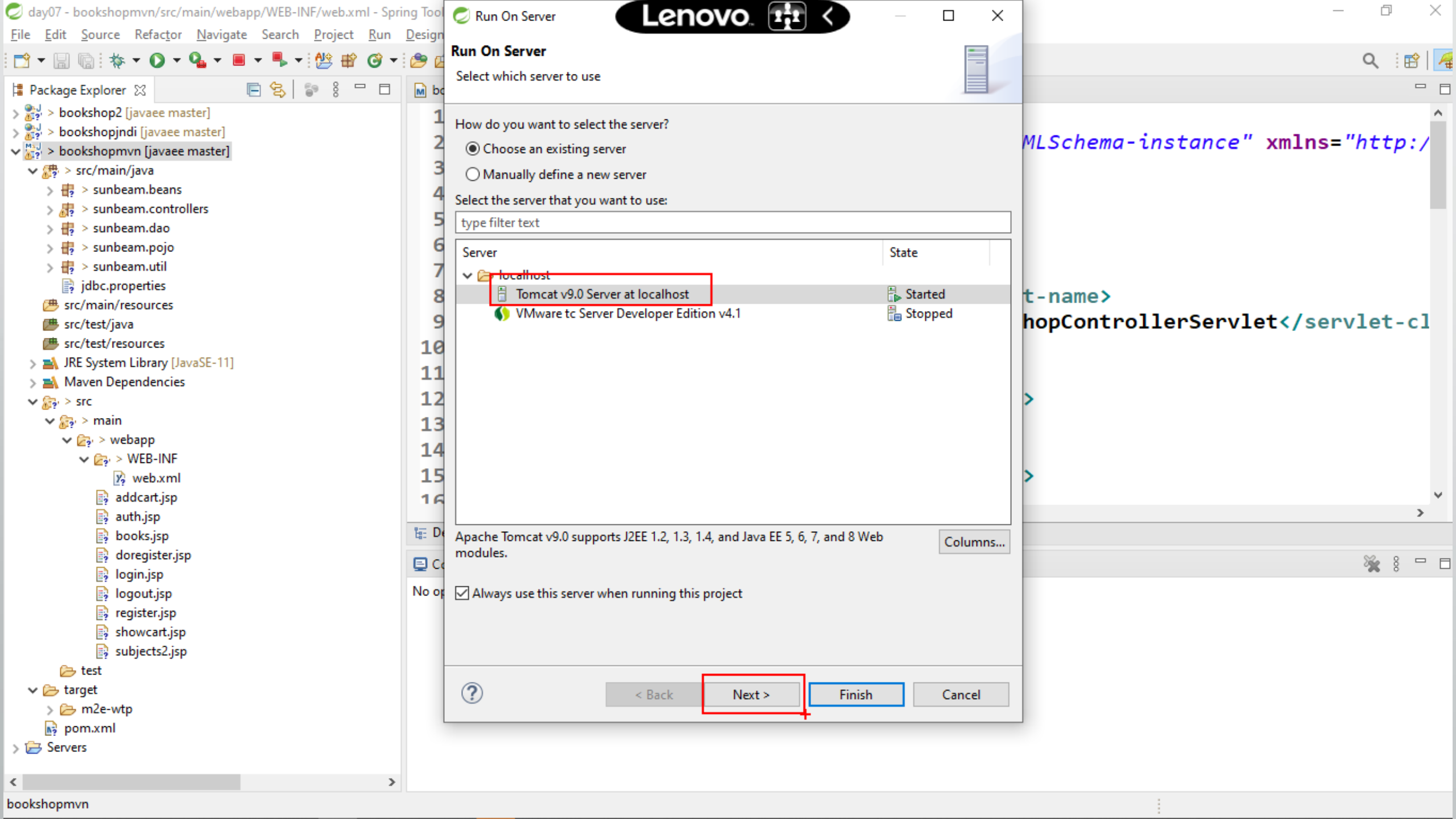
Design | Source

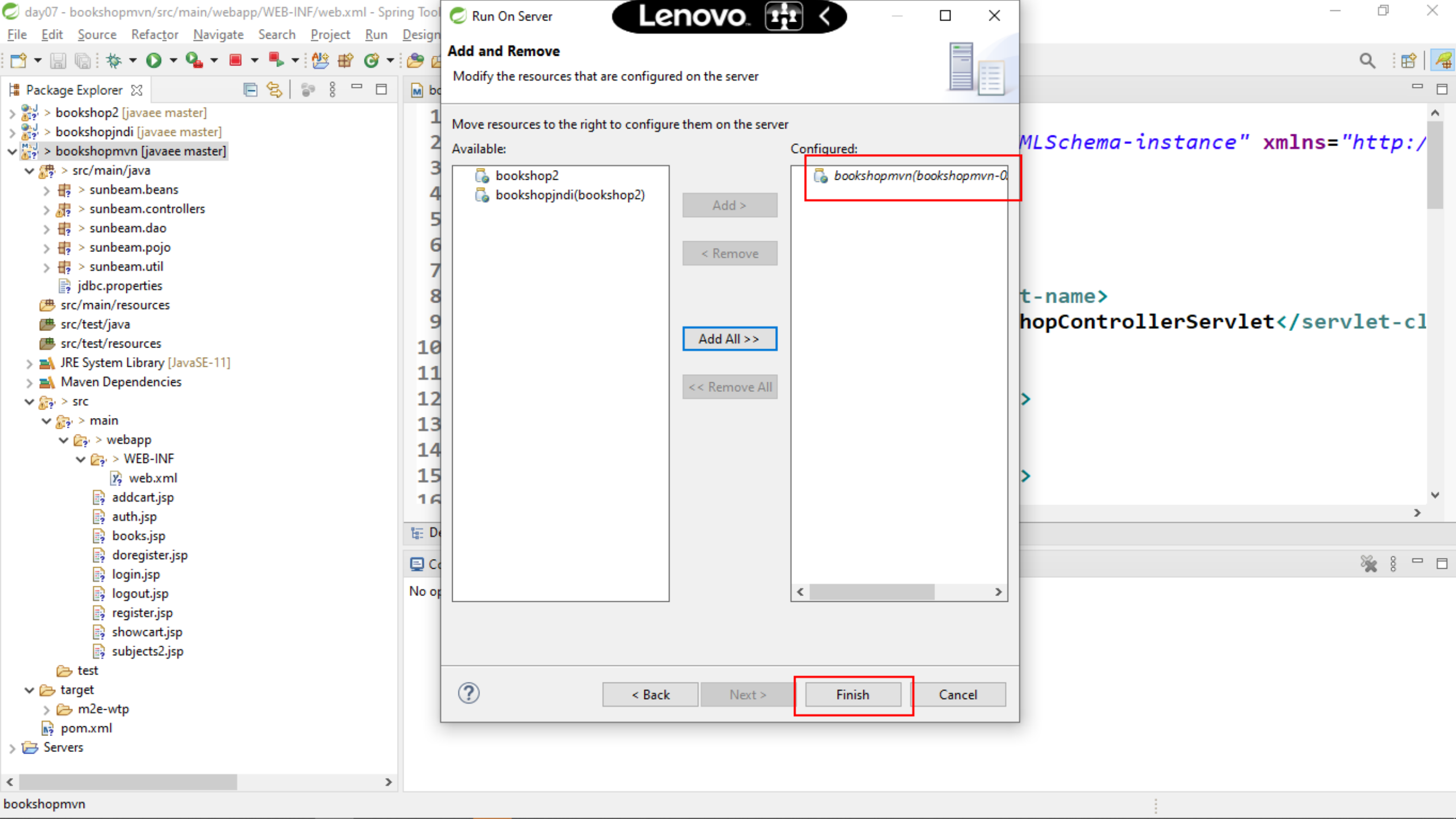
Console | Progress | Problems | Servers

No operations to display at this time.

Run Maven web application from Eclipse


day07 - bookshopmvn/sr
File Edit Source Refact
Package Explorer
bookshop2 [javaee
bookshopjndi [javae
bookshopmvn [java
src/main/java
sunbeam.bea
sunbeam.con
sunbeam.dao
sunbeam.pojo
sunbeam.util
jdbc.properties
src/main/resources
src/test/java
src/test/resources
JRE System Library
Maven Dependenc
src
main
webapp
WEB-IN
web.xr
addcart.js
auth.jsp
books.jsp
doregister
login.jsp
logout.jsp
register.js
showcart.
subjects2.
test
target
m2e-wtp
pom.xml
Servers
bookshopmvn
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Profile As
Restore from Local History...
Java EE Tools
Maven
Team
Compare With
Replace With
Configure
Spring
1 Run on Server Alt+Shift+X, R
2 Java Application Alt+Shift+X, J
3 Maven build Alt+Shift+X, M
4 Maven build...
5 Maven clean
6 Maven generate-sources
7 Maven install
8 Maven test
Run Configurations...
.xml
web.xml
ersion="1.0" encoding="UTF-8"?>
p xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:/
lay-name>bookshop2</display-name>
ome-file-list>
lcome-file>login.jsp</welcome-file>
come-file-list>
let>
rvlet-name>BookShopController</servlet-name>
rvlet-class>sunbeam.controllers.BookShopControllerServlet</servlet-cl
it-param>
<param-name>login</param-name>
<param-value>/login.jsp</param-value>
nit-param>
it-param>
<param-name>authenticate</param-name>

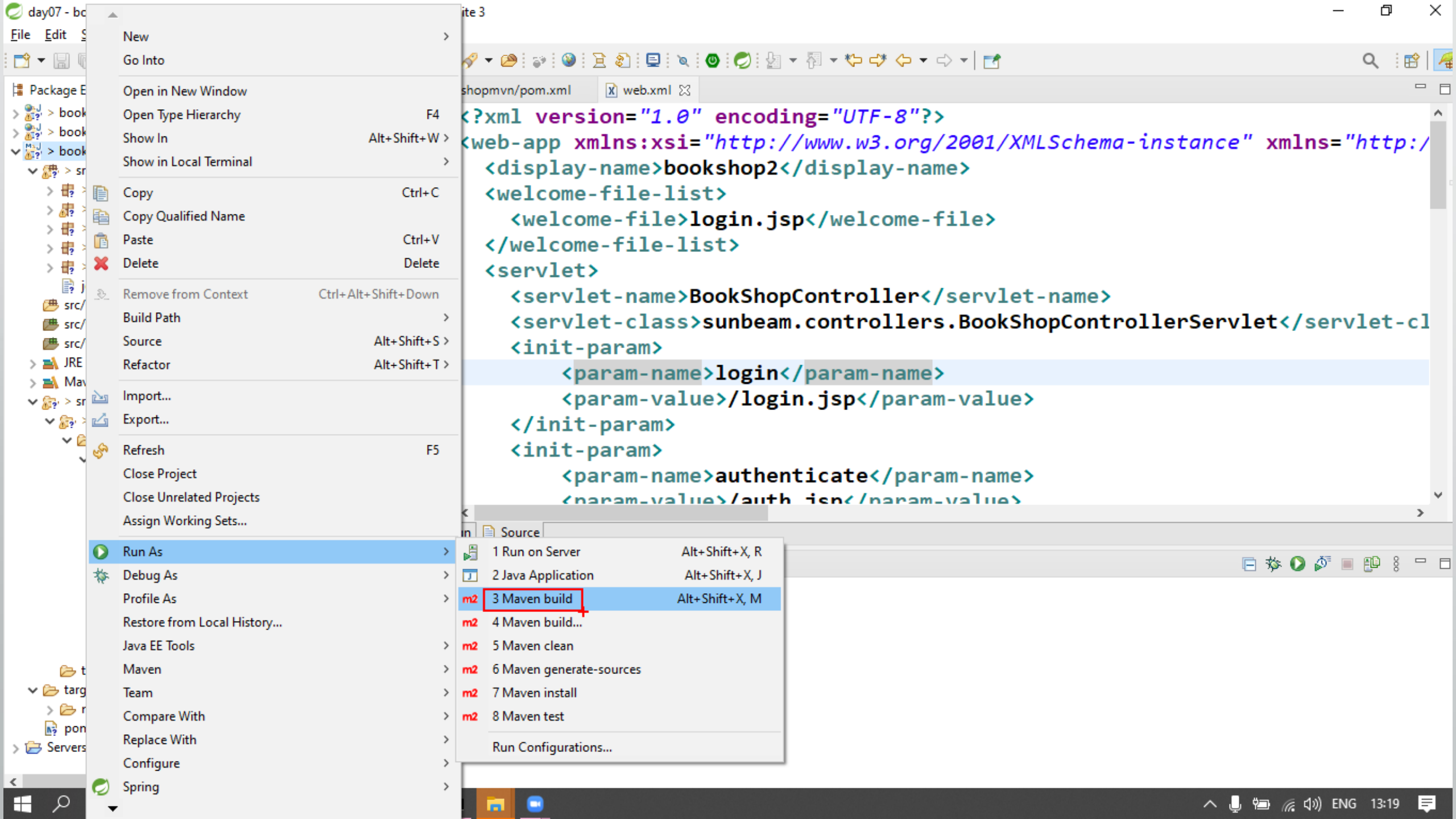

day07 - bookshopmvn/src/main/webapp/WEB-INF/web.xml - Spring Tool
File Edit Source Refactor Navigate Search Project Run Design
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee master]
bookshopmvn [javaee master]
src/main/java
sunbeam.beans
sunbeam.controllers
sunbeam.dao
sunbeam.pojo
sunbeam.util
jdbc.properties
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
src
main
webapp
WEB-INF
web.xml
addcart.jsp
auth.jsp
books.jsp
doregister.jsp
login.jsp
logout.jsp
register.jsp
showcart.jsp
subjects2.jsp
test
target
m2e-wtp
pom.xml
Servers
bookshopmvn
Run On Server
Lenovo
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Tomcat v9.0 Server at localhost
Started
VMware tc Server Developer Edition v4.1
Stopped
Apache Tomcat v9.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, 7, and 8 Web modules.
Columns...
Always use this server when running this project
< Back
Next >
Finish
Cancel
MLSchema-instance" xmlns="http:/
t-name>
hopControllerServlet</servlet-cl
No op

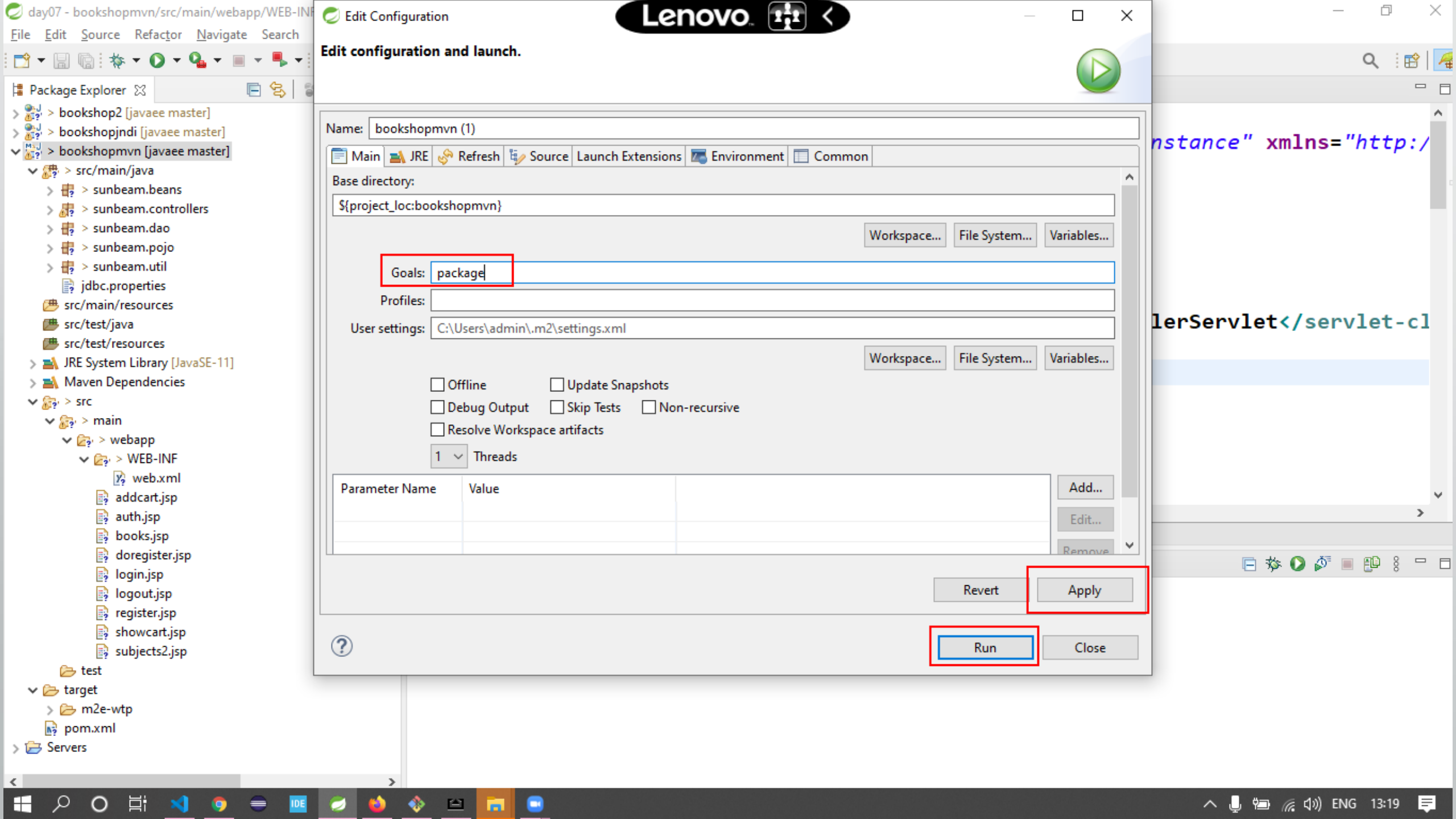

day07 - bookshopmvn/src/main/webapp/WEB-INF/web.xml - Spring Tool
File Edit Source Refactor Navigate Search Project Run Design
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee master]
bookshopmvn [javaee master]
src/main/java
sunbeam.beans
sunbeam.controllers
sunbeam.dao
sunbeam.pojo
sunbeam.util
jdbc.properties
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
src
main
webapp
WEB-INF
web.xml
addcart.jsp
auth.jsp
books.jsp
doregister.jsp
login.jsp
logout.jsp
register.jsp
showcart.jsp
subjects2.jsp
test
target
m2e-wtp
pom.xml
Servers
bookshopmvn
Run On Server
Lenovo
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
bookshop2
bookshopjndi(bookshop2)
Add >
< Remove
Add All >>
<< Remove All
Configured:
bookshopmvn(bookshopmvn-0
MLSchema-instance" xmlns="http:/
t-name>
hopControllerServlet</servlet-cl
< Back
Next >
Finish
Cancel

Run application
(Same as Dynamic Web Project execution)

- Open the browser
- Enter application login page URL
- Login and test all features.

Deploy Maven web application in Tomcat


day07 - bo
File Edit
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Profile As
Restore from Local History...
Java EE Tools
Maven
Team
Compare With
Replace With
Configure
Spring
1 Run on Server Alt+Shift+X, R
2 Java Application Alt+Shift+X, J
3 Maven build Alt+Shift+X, M
4 Maven build...
5 Maven clean
6 Maven generate-sources
7 Maven install
8 Maven test
Run Configurations...
shopmvn/pom.xml
web.xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:/
<display-name>bookshop2</display-name>
<welcome-file-list>
<welcome-file>login.jsp</welcome-file>
</welcome-file-list>
<servlet>
<servlet-name>BookShopController</servlet-name>
<servlet-class>sunbeam.controllers.BookShopControllerServlet</servlet-cl
<init-param>
<param-name>login</param-name>
<param-value>/login.jsp</param-value>
</init-param>
<init-param>
<param-name>authenticate</param-name>
Source
ENG 13:19


day07 - bookshopmvn/src/main/webapp/WEB-INF
File Edit Source Refactor Navigate Search
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee master]
bookshopmvn [javaee master]
src/main/java
sunbeam.beans
sunbeam.controllers
sunbeam.dao
sunbeam.pojo
sunbeam.util
jdbc.properties
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
src
main
webapp
WEB-INF
web.xml
addcart.jsp
auth.jsp
books.jsp
doregister.jsp
login.jsp
logout.jsp
register.jsp
showcart.jsp
subjects2.jsp
test
target
m2e-wtp
pom.xml
Servers
Edit Configuration
Edit configuration and launch.
Name: bookshopmvn (1)
Main JRE Refresh Source Launch Extensions Environment Common
Base directory:
${project_loc:bookshopmvn}
Workspace... File System... Variables...
Goals: package
Profiles:
User settings: C:\Users\admin\.m2\settings.xml
Workspace... File System... Variables...
Offline
Update Snapshots
Debug Output
Skip Tests
Non-recursive
Resolve Workspace artifacts
1 Threads
Parameter Name
Value
Add...
Edit...
Remove
Revert
Apply
Run
Close
nstance" xmlns="http:/
lerServlet</servlet-cl
ENG
13:19

Run application in Tomcat

- Stop tomcat from Eclipse (if already running).
- Run tomcat. <tomcat>/bin/startup.bat.
- Check if tomcat running well. Browser http://localhost:8080
- Copy application war file (created while "package") into <tomcat>/webapps
- Check if appln running well. Browser http://localhost:8080/appln_dir_name

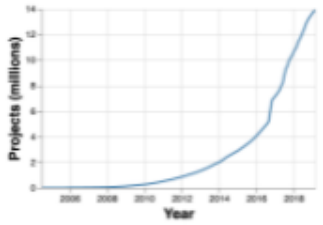
Understanding Maven Central & Local Repository

Thanks | Subjects | Thanks | Repository: mysql » mysq

mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.24

Maven Central

Indexed Artifacts (21.4M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools

Home » mysql » mysql-connector-java » 8.0.24

MySQL Connector/J » 8.0.24

JDBC Type 4 driver for MySQL

License	GPL 2.0
Categories	MySQL Drivers
Organization	Oracle Corporation
HomePage	http://dev.mysql.com/doc/connector-j/en/
Date	(Apr 19, 2021)
Files	jar (2.3 MB) View All
Repositories	Central
Used By	5,417 artifacts

Note: There is a new version for this artifact

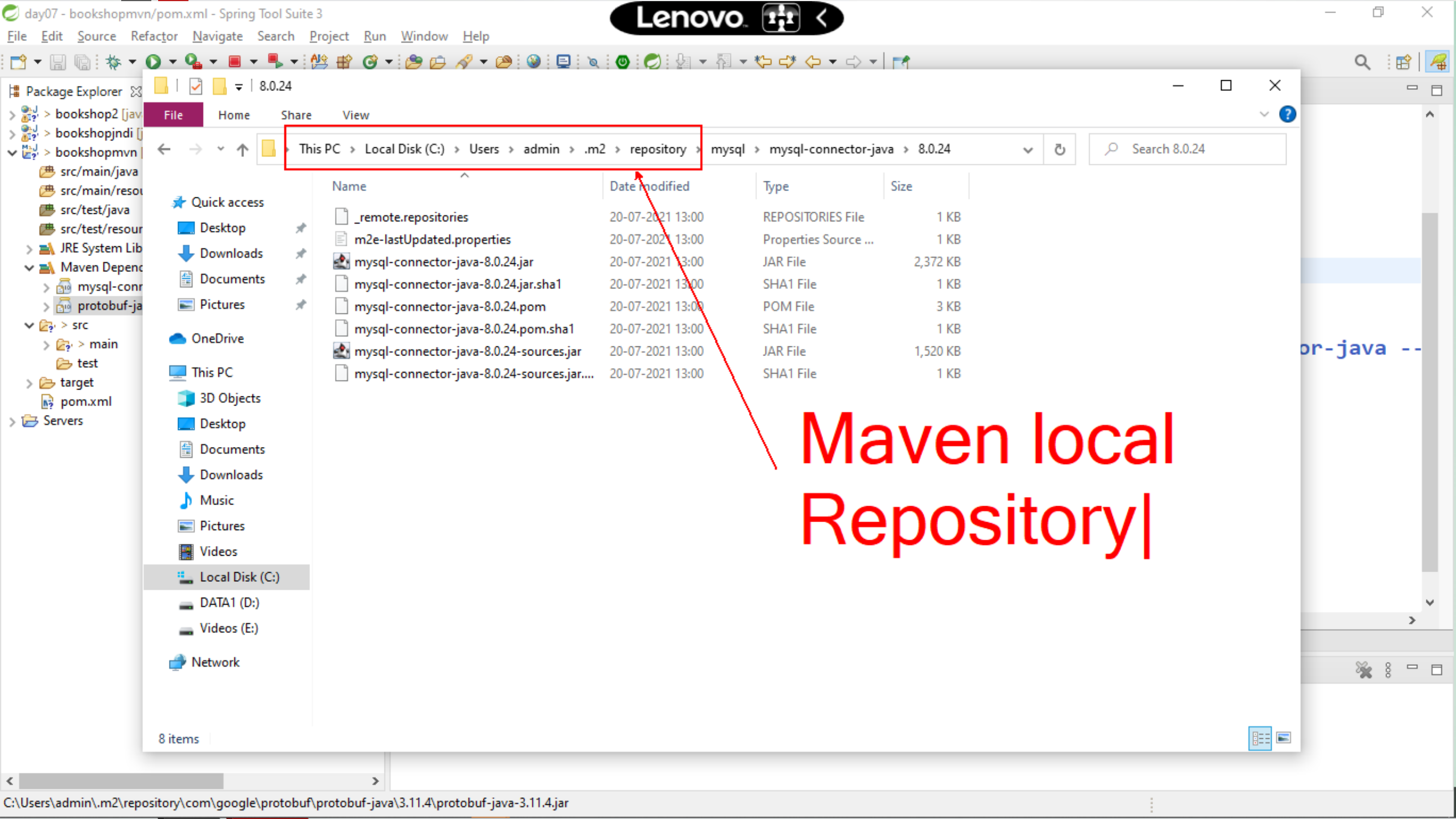
New Version	8.0.25

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

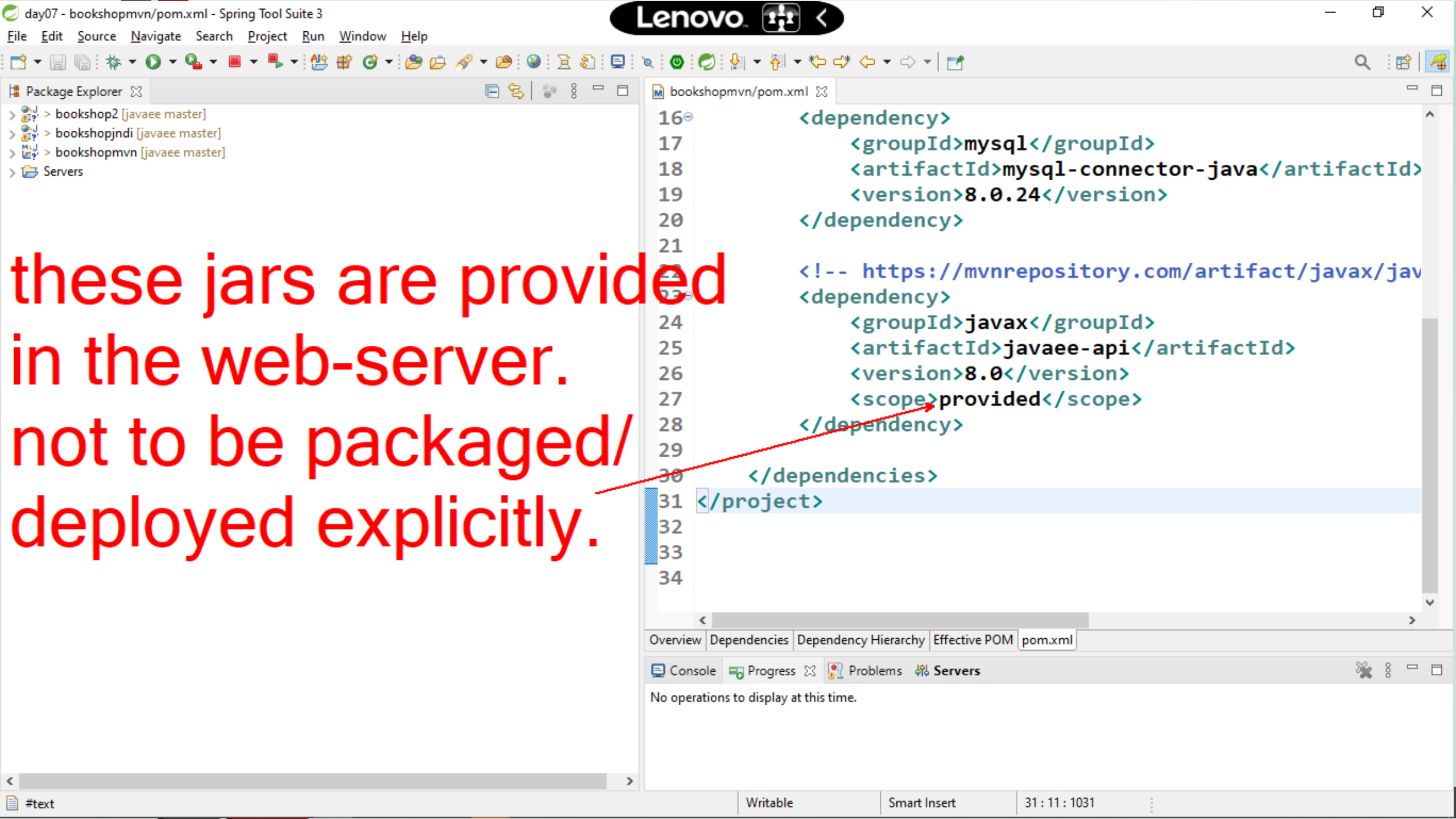
```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.24</version>
</dependency>
```

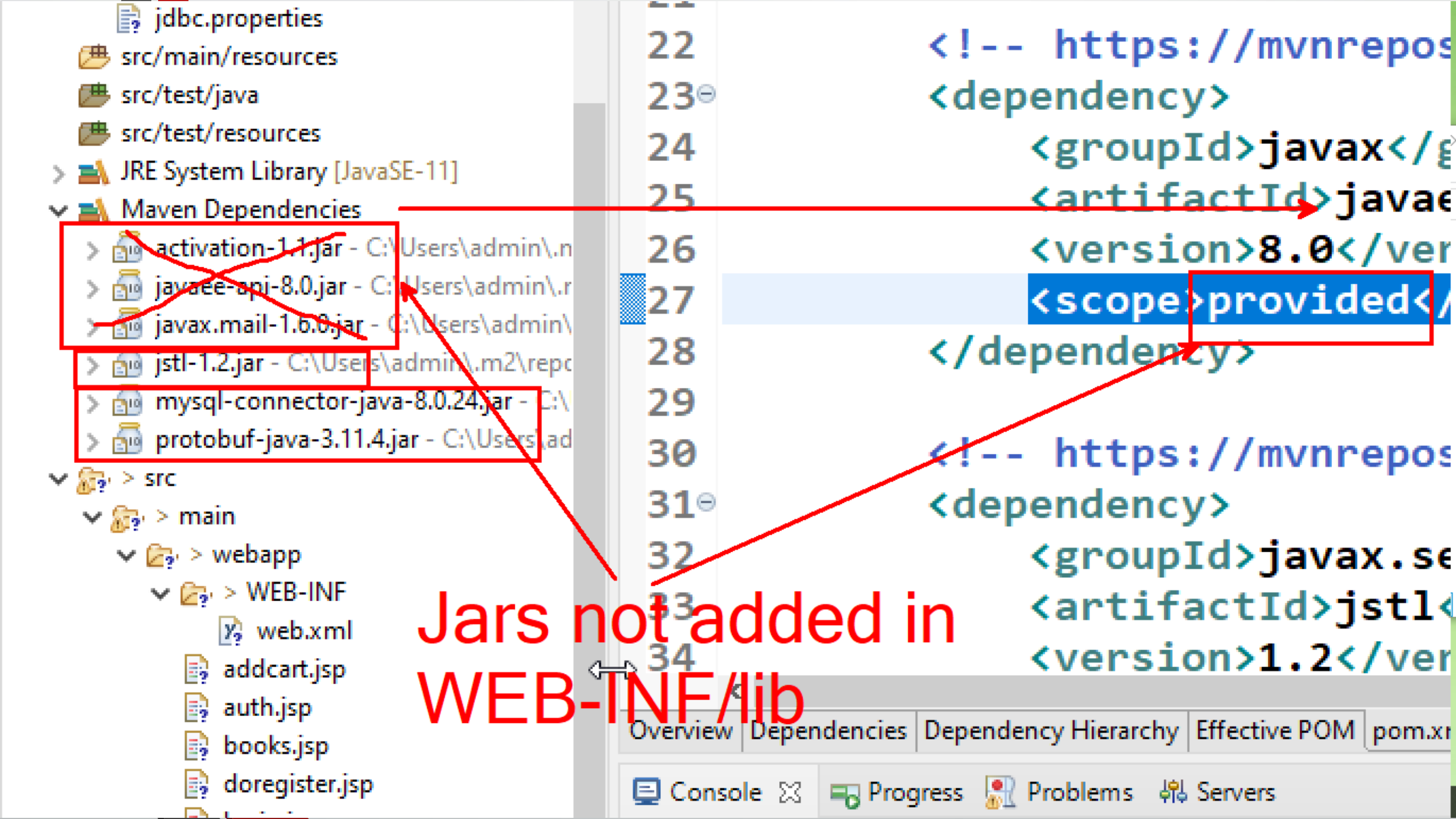
Include comment with link to declaration

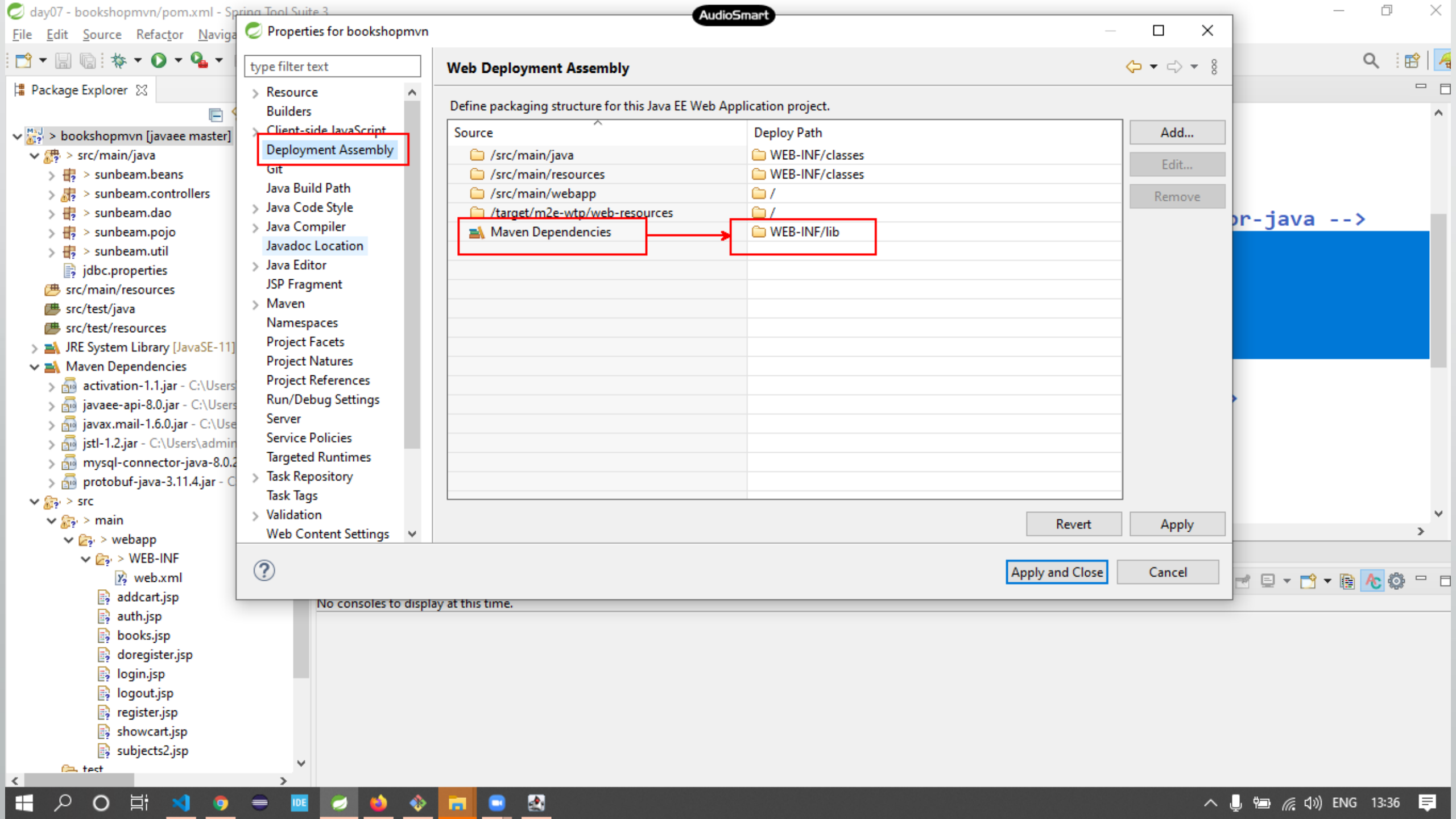
Copied to clipboard!


day07 - bookshopmvn/pom.xml - Spring Tool Suite 3
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
bookshop2 [jav
bookshopjndi [j
bookshopmvn
src/main/java
src/main/resou
src/test/java
src/test/resour
JRE System Lib
Maven Depend
mysql-conr
protobuf-ja
src
main
test
target
pom.xml
Servers
8.0.24
File Home Share View
This PC › Local Disk (C:) › Users › admin › .m2 › repository › mysql › mysql-connector-java › 8.0.24
Search 8.0.24
Quick access
Desktop
Downloads
Documents
Pictures
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
DATA1 (D:)
Videos (E:)
Network
Name
Date modified
Type
Size
_remote.repositories 20-07-2021 13:00 REPOSITORIES File 1 KB
m2e-lastUpdated.properties 20-07-2021 13:00 Properties Source ... 1 KB
mysql-connector-java-8.0.24.jar 20-07-2021 13:00 JAR File 2,372 KB
mysql-connector-java-8.0.24.jar.sha1 20-07-2021 13:00 SHA1 File 1 KB
mysql-connector-java-8.0.24.pom 20-07-2021 13:00 POM File 3 KB
mysql-connector-java-8.0.24.pom.sha1 20-07-2021 13:00 SHA1 File 1 KB
mysql-connector-java-8.0.24-sources.jar 20-07-2021 13:00 JAR File 1,520 KB
mysql-connector-java-8.0.24-sources.jar.... 20-07-2021 13:00 SHA1 File 1 KB
8 items
Maven local
Repository
C:\Users\admin\.m2\repository\com\google\protobuf\protobuf-java\3.11.4\protobuf-java-3.11.4.jar

Understanding Maven dependency scope = provided


day07 - bookshopmvn/pom.xml - Spring Tool Suite 3
File Edit Source Navigate Search Project Run Window Help
Package Explorer
bookshop2 [javaee master]
bookshopjndi [javaee master]
bookshopmvn [javaee master]
Servers
these jars are provided
in the web-server.
not to be packaged/
deployed explicitly.
bookshopmvn/pom.xml
16 <dependency>
17 <groupId>mysql</groupId>
18 <artifactId>mysql-connector-java</artifactId>
19 <version>8.0.24</version>
20 </dependency>
21
22 <!-- https://mvnrepository.com/artifact/javax/jav
23 <dependency>
24 <groupId>javax</groupId>
25 <artifactId>javaee-api</artifactId>
26 <version>8.0</version>
27 <scope>provided</scope>
28 </dependency>
29
30 </dependencies>
31 </project>
32
33
34
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Console Progress Problems Servers
No operations to display at this time.
#text
Writable
Smart Insert
31 : 11 : 1031


jdbc.properties
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
activation-1.1.jar - C:\Users\admin\.n
javaee-api-8.0.jar - C:\Users\admin\.r
javax.mail-1.6.0.jar - C:\Users\admin\
jstl-1.2.jar - C:\Users\admin\.m2\repo
mysql-connector-java-8.0.24.jar - C:\
protobuf-java-3.11.4.jar - C:\Users\ad
src
main
webapp
WEB-INF
web.xml
addcart.jsp
auth.jsp
books.jsp
doregister.jsp
22 <!-- https://mvnrepos
23 <dependency>
24 <groupId>javax</g
25 <artifactId>javae
26 <version>8.0</ver
27 <scope>provided</
28 </dependency>
29
30 <!-- https://mvnrepos
31 <dependency>
32 <groupId>javax.se
33 <artifactId>jstl<
34 <version>1.2</ver
Jars not added in
WEB-INF/lib
Overview
Dependencies
Dependency Hierarchy
Effective POM
pom.x
Console
Progress
Problems
Servers


AudioSmart
day07 - bookshopmvn/pom.xml - Spring Tool Suite 3
File Edit Source Refactor Naviga
Package Explorer
> bookshopmvn [javaee master]
> src/main/java
> sunbeam.beans
> sunbeam.controllers
> sunbeam.dao
> sunbeam.pojo
> sunbeam.util
jdbc.properties
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
activation-1.1.jar - C:\Users
javaee-api-8.0.jar - C:\Users
javax.mail-1.6.0.jar - C:\Use
jstl-1.2.jar - C:\Users\admin
mysql-connector-java-8.0.2
protobuf-java-3.11.4.jar - C
> src
> main
> webapp
> WEB-INF
web.xml
addcart.jsp
auth.jsp
books.jsp
doregister.jsp
login.jsp
logout.jsp
register.jsp
showcart.jsp
subjects2.jsp
test
Properties for bookshopmvn
type filter text
Resource
Builders
Client-side JavaScript
Deployment Assembly
Git
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Java Editor
JSP Fragment
Maven
Namespaces
Project Facets
Project Natures
Project References
Run/Debug Settings
Server
Service Policies
Targeted Runtimes
Task Repository
Task Tags
Validation
Web Content Settings
Web Deployment Assembly
Define packaging structure for this Java EE Web Application project.
Source
Deploy Path
/src/main/java
WEB-INF/classes
/src/main/resources
WEB-INF/classes
/src/main/webapp
/
/target/m2e-wtp/web-resources
/
Maven Dependencies
WEB-INF/lib
Add...
Edit...
Remove
Revert
Apply
Apply and Close
Cancel
No consoles to display at this time.
r-java -->
ENG
13:36

Thank you

Nilesh Ghule <nilesh@sunbeaminfo.com>

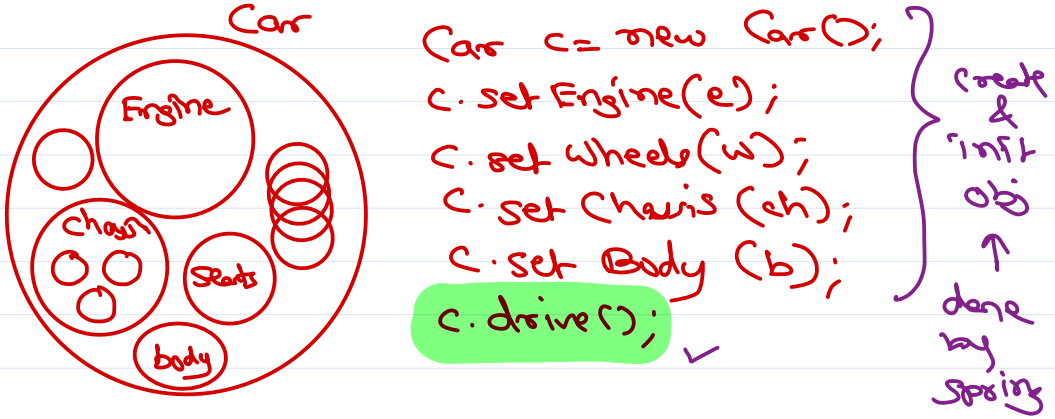
Spring and Hibernate

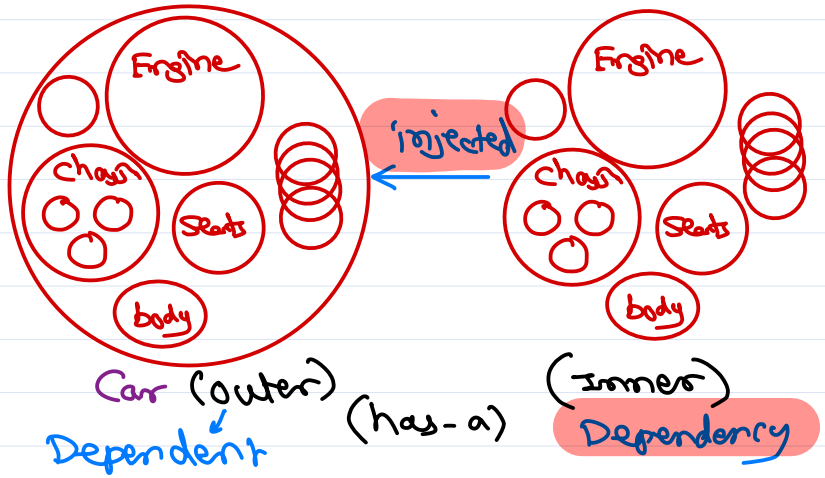
Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring Framework

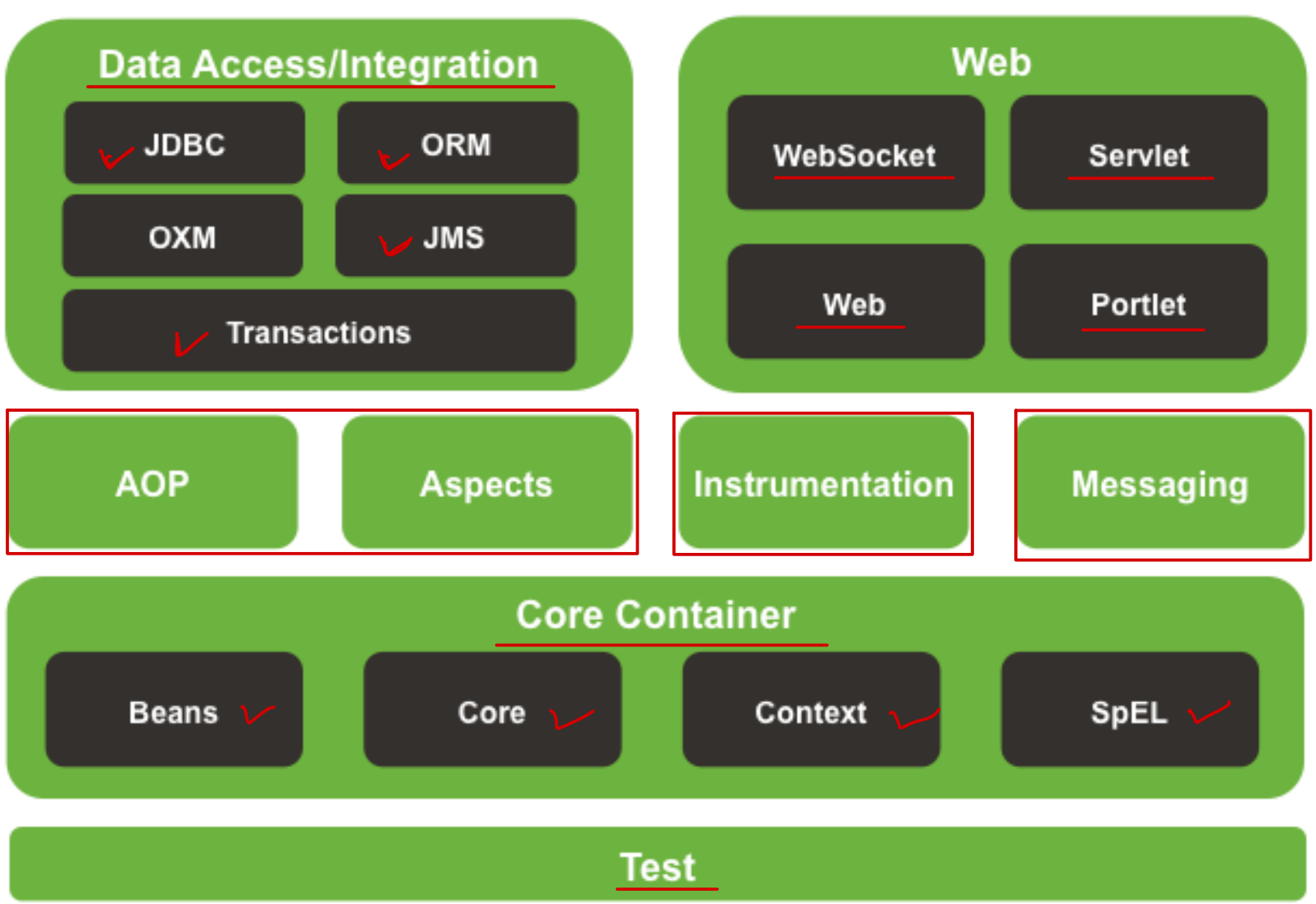
coding
unit testing
deployment
...

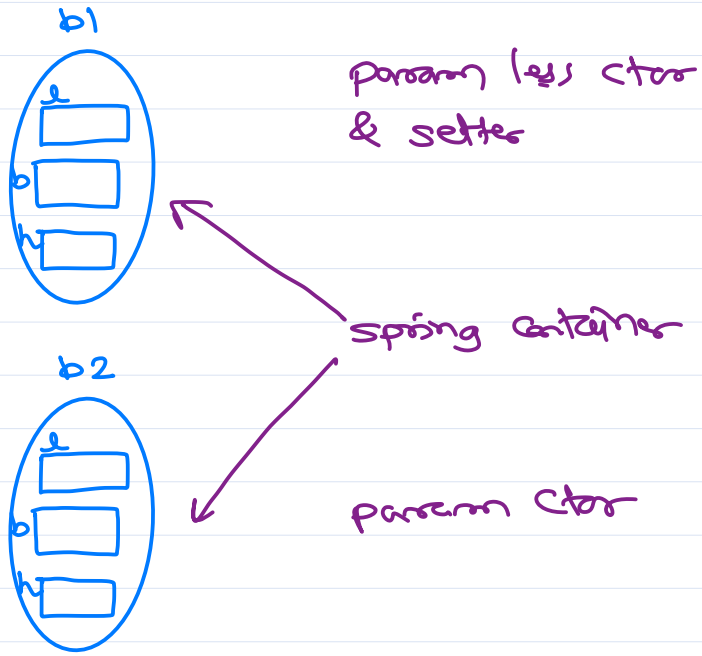
- Spring is **light-weight comprehensive** framework to **simplify** Java development.
- Developed by Rod Johnson. Spring 3 added annotations while Spring 5 added reactive.
  2003
- Spring pros
  - Inversion of control (Dependency injection)
  - Test driven development
  - Extensive but Flexible and modular
  - Smooth integration with existing technologies
  - Eliminate boilerplate code
  - Extendable
  - Maintainable → readable, exceptions unchecked
- Spring cons
  - Heavy configuration (XML, Java, Both)
  - Module versioning & compatibility
  - Application deployment





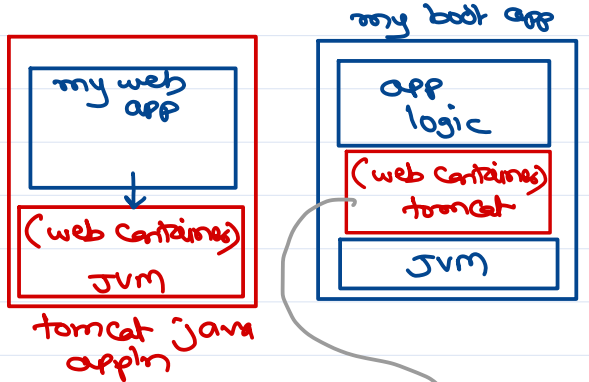
Spring architecture (3.x)



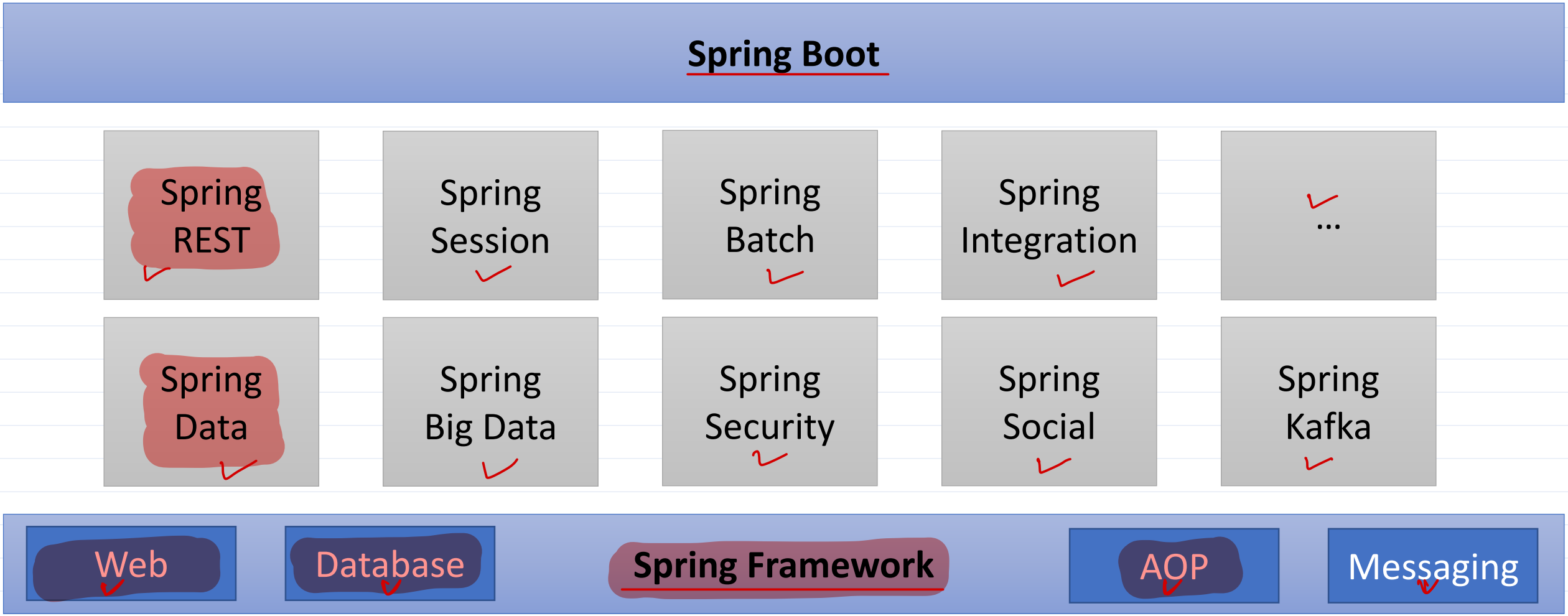

b1
b2
param less ctor
& setter
spring container
param ctor

Spring Boot

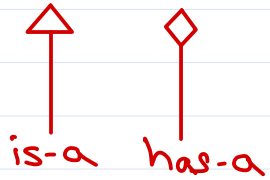
- Spring Boot is NOT a new standalone framework. It is combination of various frameworks on spring platform i.e. spring-core, spring-webmvc, spring-data, …
- Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can “just run”.
- Spring Boot take an "opinionated view" of the Spring platform and third-party libraries. So most of applications need minimum configuration.
- Primary Goals/Features
  - Provide a radically faster and widely accessible "Quick Start".
  - Opinionated config, yet quickly modifiable for different requirements.
  - Managing versions of dependencies with starter projects.
  - Provide lot of non-functional common features (e.g. security, servers, health checks, ...).
  - No extra code generation (provide lot of boilerplate code) and XML config.
  - Easy deployment and containerisation with embedded Web Server.
- Recommended to use for new apps and not to port legacy Spring apps.

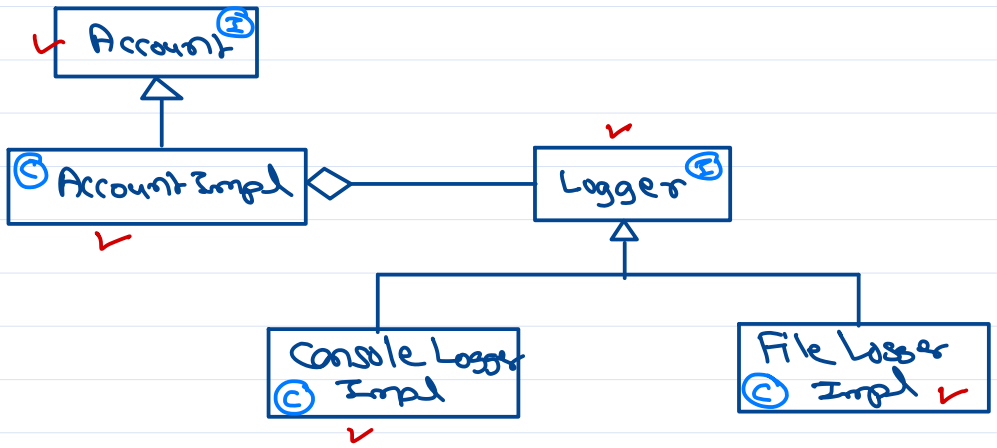


Spring Boot



Spring Boot = Spring framework + Embedded web-server + Auto-configuration - XML config - Jar conflicts


is-a
has-a


Account
AccountImpl
Logger
ConsoleLogger Impl
FileLogger Impl

Spring Dependency Injection

Dependency Injection

- Spring container injects dependency beans into dependent beans.
- Spring container is also called as IoC container.
- Dependency Injection
  - Setter based DI
  - Constructor based DI
  - Field based DI

XML based DI

- Spring beans are declared into *Spring bean configuration file*.
- Bean factory or application context reads the file. It instantiate and initialize bean objects at runtime.
- XML configuration
  - Initializing properties
  - Initializing dependency beans
  - Initializing collections

Dependency Injection

- One interface may have multiple implementations.
- This makes application more extendable and maintainable.
- Desired implementation can be plugged in using DI.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

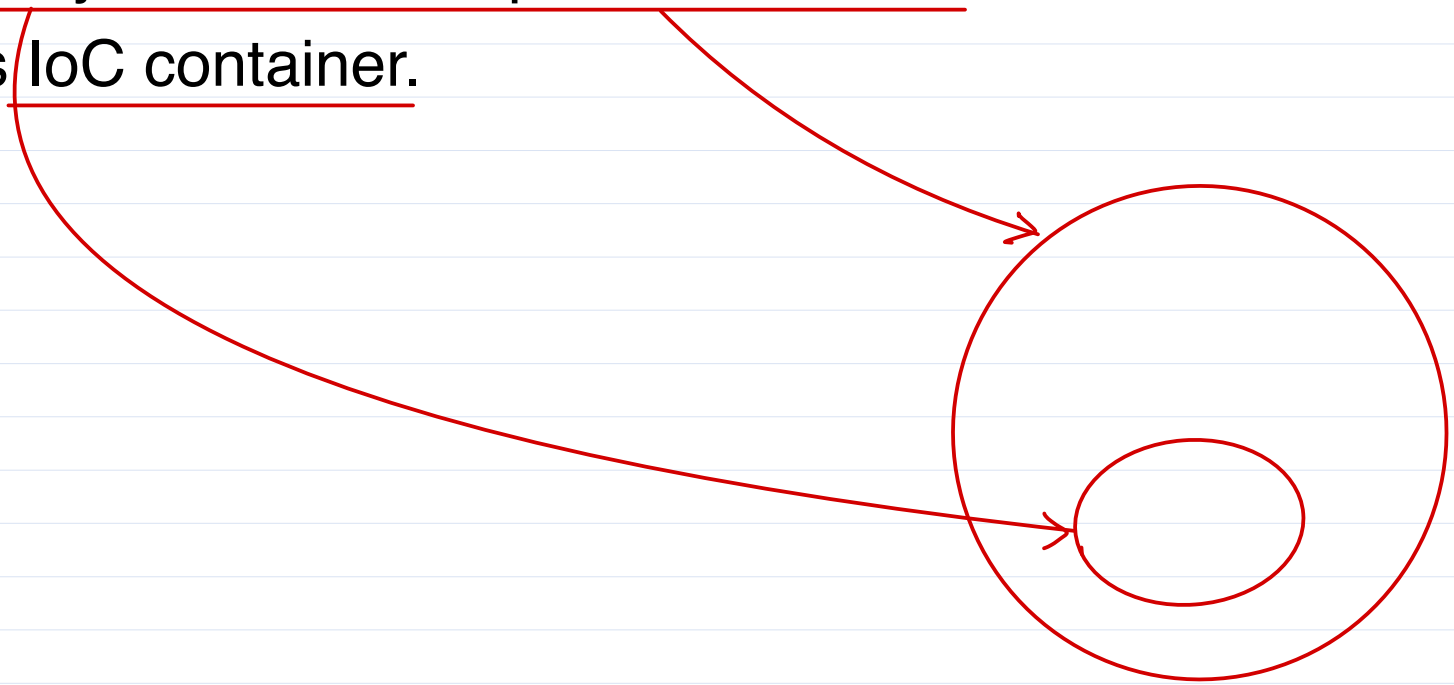
Spring and Hibernate

Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring Dependency Injection

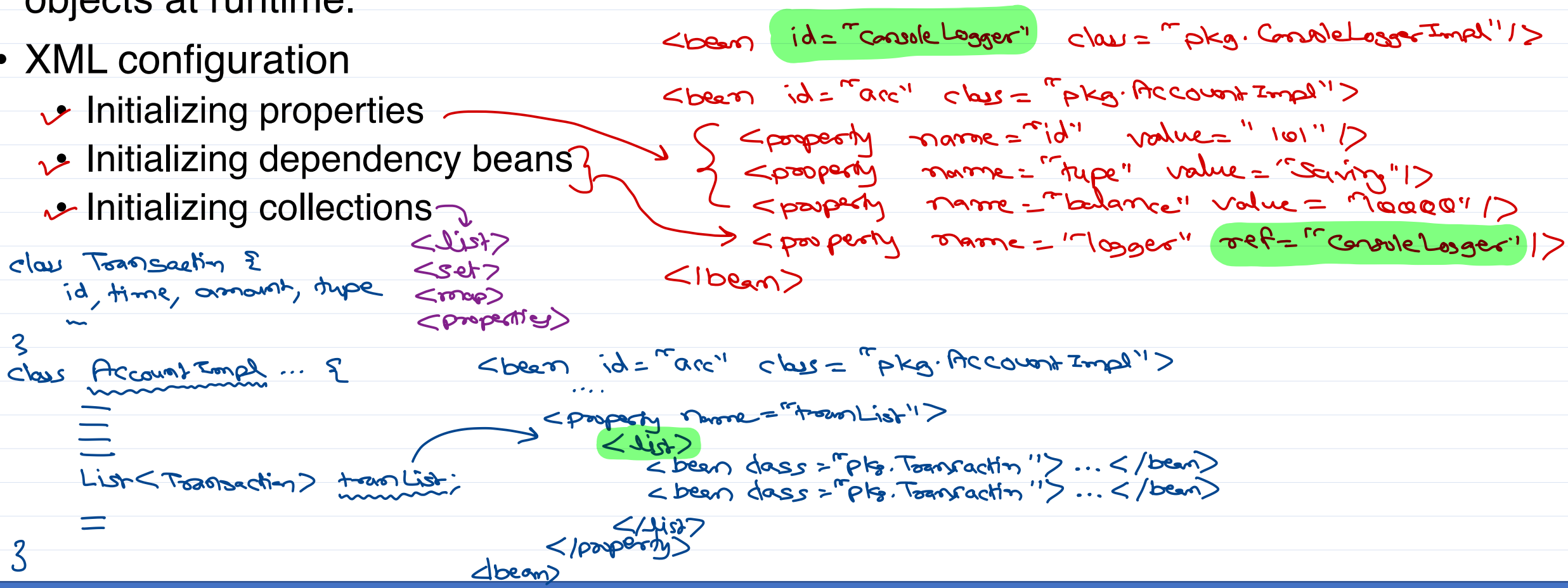
Dependency Injection

- Spring container injects dependency beans into dependent beans.
- Spring container is also called as IoC container.
- Dependency Injection
  - Setter based DI
  - Constructor based DI
  - Field based DI



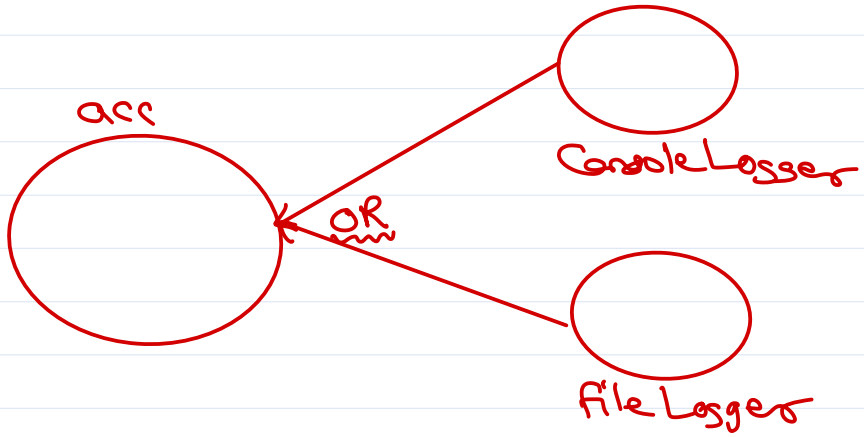
XML based DI

- Spring beans are declared into *Spring bean configuration file*.
- Bean factory or application context reads the file. It instantiate and initialize bean objects at runtime.
- XML configuration
  - Initializing properties
  - Initializing dependency beans
  - Initializing collections



Dependency Injection

- One interface may have multiple implementations.
- This makes application more extendable and maintainable.
- Desired implementation can be plugged in using DI.



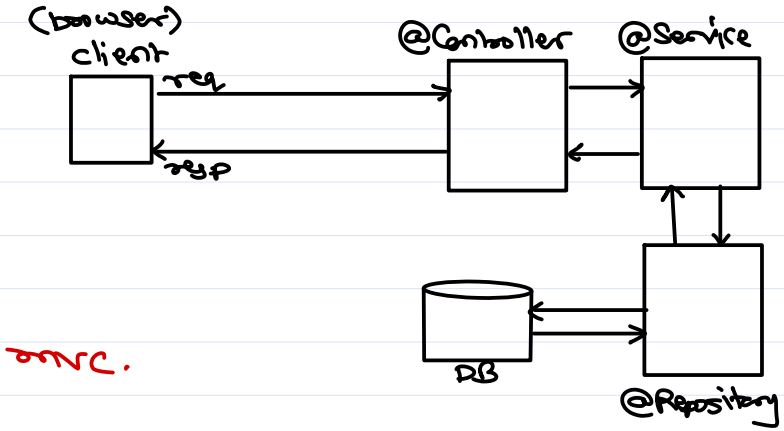
Annotation config

- Does spring configuration without using any XML file.
- The bean creation is encapsulated in @Configuration class and beans are represented as @Bean methods. method name = bean id
- Annotation configuration is processed using AnnotationConfigApplicationContext.

Stereo-type annoatations

Stereo type annotations

- Auto-detecting beans (avoid manual config of beans).
  - @Component → generic spring bean
  - @Service → business logic spring bean
  - @Repository → spring bean persisting data of app i.e. Dao.
  - @Controller and @RestController → Controller beans in spring mvc.
- In XML config file → mixed config
  - <context:component-scan basePackages="____"/>
- Annotation based config
  - @ComponentScan(basePackages = "pkg")
  - includeFilters and excludeFilters can be used to control bean detection.

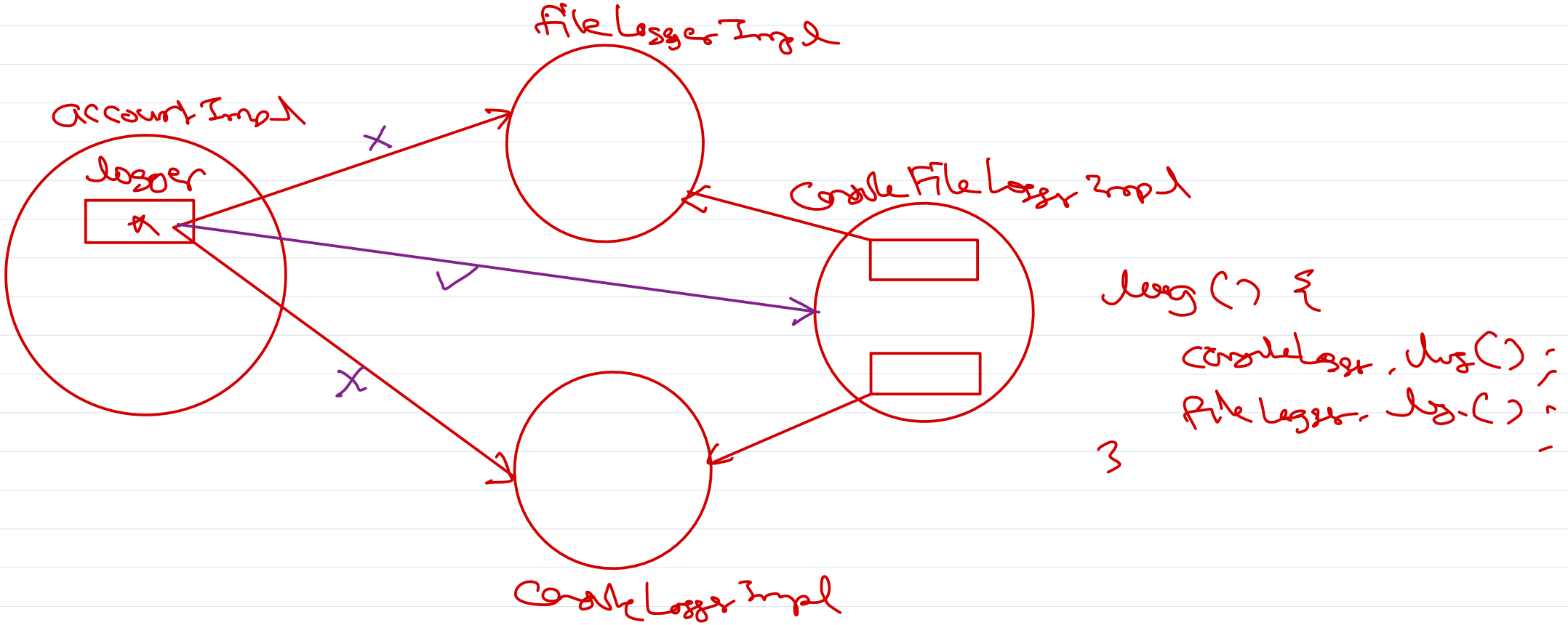


Spring Boot @SpringBootApplication implicitly does @ComponentScan on current package & its sub packages.

Auto-wiring

Auto-wiring

- Automatically injecting appropriate dependency beans into the dependent beans.



Auto-wiring (Annotation config)

- @Autowired (Spring Framework)
  - Setter level: setter based DI
  - Constructor level: constructor based DI
  - Field level: field based DI *
- @Autowired: Find bean of corresponding field type and assign it.
  - If no bean is found of given type, it throws exception.
    - @Autowired(required=false): no exception is thrown, auto-wiring skipped.
  - If multiple beans are found of given type, it try to attach bean of same name. and if such bean is not found, then throw exception.
  - If multiple beans are found of given type, programmer can use @Qualifier to choose expected bean. @Qualifier can only be used to resolve conflict in case of @Autowired.
- @Resource (JSR 250)
  - @Resource: DI byName, byType, byQualifier
- @Inject (JSR 330) – same as @Autowired
  - @Inject: DI byType, byQualifier (@Named), byName

@Autowired on param constructor
Spring docs recommend this. In same step obj creation & DI is performed. Execute faster.

Spring 5 - implicitly consider @Autowired on single arg constructor.

Test 3

Test 1

field will be null.

Test 2

Test 4

Test 7

Test 5

@Autowired
low priority
① by Type
② by Name
③ by Qualifier
high priority

Auto-wiring (XML config)

- <bean id="..." class="..." autowire="default|no|byType|byName|constructor" .../>
  - default / no: Auto-wiring is disabled.
  - byType: Dependency bean of property type will be assigned (via setter).
    - If multiple beans of required type are available, then exception is thrown.
    - If no bean of required type is available, auto-wiring is not done.
  - byName: Dependency bean of property name will be assigned (via setter).
    - If no bean of required name is available, auto-wiring is not done.
  - constructor: Dependency bean of property type will be assigned via single argument constructor (of bean type).
- <bean id="..." class="..." autowire-candidate="true|false" .../>
  - false -- do not consider this dependency bean for auto-wiring.

Mixed configuration

- XML config + Annotations
  - XML config file is loaded using ClassPathXmlApplicationContext.
  - <context:annotation-config/> activate annotation processing.
    - @PostConstruct
    - @PreDestroy
    - @Autowired
    - @Qualifier

Spring EL

Spring Expression Language

Like JSP EL ${ — }

#{ — }

- SPEL can be used with XML config or @Value annotation.
- Using XML config <property name="..." value="..." /> (value: SpEL)
  - value="#{bean.field}"
  - value="${property-key}" (properties file)
    - In beans.xml: <context:property-placeholder location="classpath:app.properties" />
- Using Annotation config @Value (field level, setter)
  - Auto-wire dependency beans: #{bean}
  - Values using SPEL expressions: #{bean.field} (getter()) or #{bean.method()}
  - Values using SPEL expressions from properties files: ${property-key}
    - In @Configuration class: @PropertySource("classpath:app.properties")

Spring Expression Language

- SpEL is a powerful expression language that supports querying and manipulating an object graph at runtime. Syntactically it is similar to EL.
- SpEL can be used in all spring framework components/products.
- SpEL supports Literal expressions, Regular expressions., Class expressions, Accessing properties, Collections, Method invocation, Relational operators, Assignment, Bean references, Inline lists/maps, Ternary operator, etc.
- SpEL expressions are internally evaluated using SpELExpressionParser.
  - ExpressionParser parser = new SpELExpressionParser(); → most commonly used.
  - value = parser.parseExpression("'Hello World'.concat('!')");
  - value = parser.parseExpression("new String('Hello World').toUpperCase()");
  - value = parser.parseExpression("bean.list[0]");
- SpEL expressions are slower in execution due parsing. Spring 4.1 added SpEL compiler to speed-up execution by creating a class for expression behaviour at runtime.

Spring beans

Spring bean life cycle

- Spring container creates (singleton) spring bean objects when container is started.
- Spring container controls creation and destruction of bean objects.
- There are four options to control bean life cycle.
  - InitializingBean and DisposableBean callback interfaces
  - Aware interfaces for specific behaviours
    - BeanNameAware, BeanFactoryAware, ApplicationContextAware, etc.
  - BeanPostProcessor callback interface
  - Custom init() / @PostConstruct and destroy() / @PreDestroy methods

Spring bean life cycle

- Bean creation
  1. Instantiation (Constructor)
  2. Set properties (Field/setter based DI)
  3. BeanNameAware.setBeanName()
  4. ApplicationContextAware.setApplicationContext()
  5. Custom BeanPostProcessor.postProcessBeforeInitialization()
  6. Custom init() (@PostConstructor)
  7. InitializingBean.afterPropertiesSet()
  8. Custom BeanPostProcessor.postProcessAfterInitialization()
- Bean destruction
  1. Custom destroy() (@PreDestroy)
  2. DisposableBean.destroy()
  3. Object.finalize()

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring and Hibernate

Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring beans

XML Bean Factory

- Deprecated
- Minimal Dependency injection features
  - Beans loading
  - Auto-wiring
- Create a singleton bean when it is accessed first time in the application.
- No bean life cycle management

demo01

b1 & b2 beans are created here.

```
ApplicationContext ctx = new
    ClassPathXmlAppContext("beans.xml");
Box b1 = (Box) ctx.getBean("b1");
```

return ptr to b1 bean (which is created earlier).

demo01

```
ClasspathXmlResource rs = new ClasspathXmlResource("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory(rs);

Box b1 = (Box) factory.getBean("b1");
sop(b1.calcVolume());

Box b2 = (Box) factory.getBean("b1");
```

b1 bean obj is created.

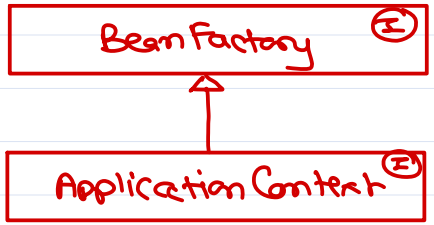
return bean ref of already created bean

Application Context

- Features
  - Dependency injection - Beans loading and Auto-wiring
  - Automatic BeanPostProcessor registration i.e. Bean life cycle management
  - Convenient MessageSource access (Internationalization)
  - ApplicationEvent publication
- Create all singleton beans when application context is loaded.

Application Context is incharge of spring container.

- ApplicationContext (interface)
  - ClassPathXmlApplicationContext (absolute path of xml file / project resources dir)
  - FileSystemXmlApplicationContext
  - AnnotationConfigApplicationContext (configs given in @Configuration marked class - AppConfig.class)
  - WebApplicationContext (interface)
    - XmlWebApplicationContext
    - AnnotationConfigWebApplicationContext

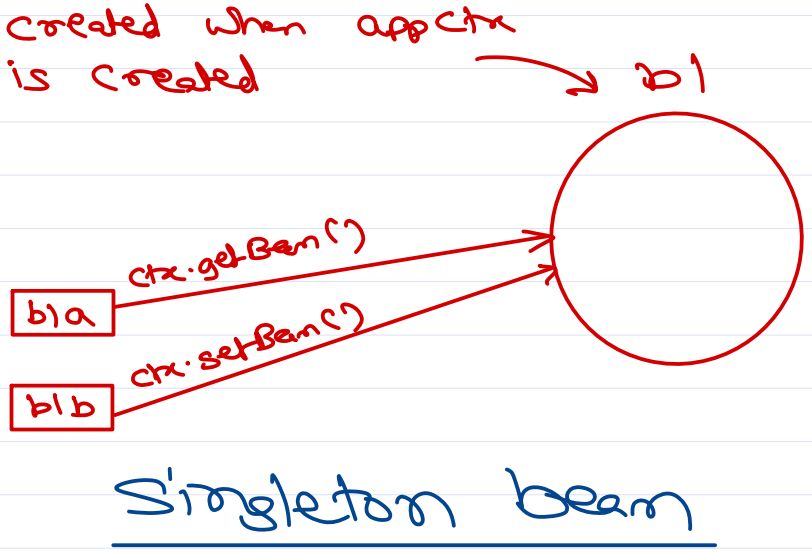


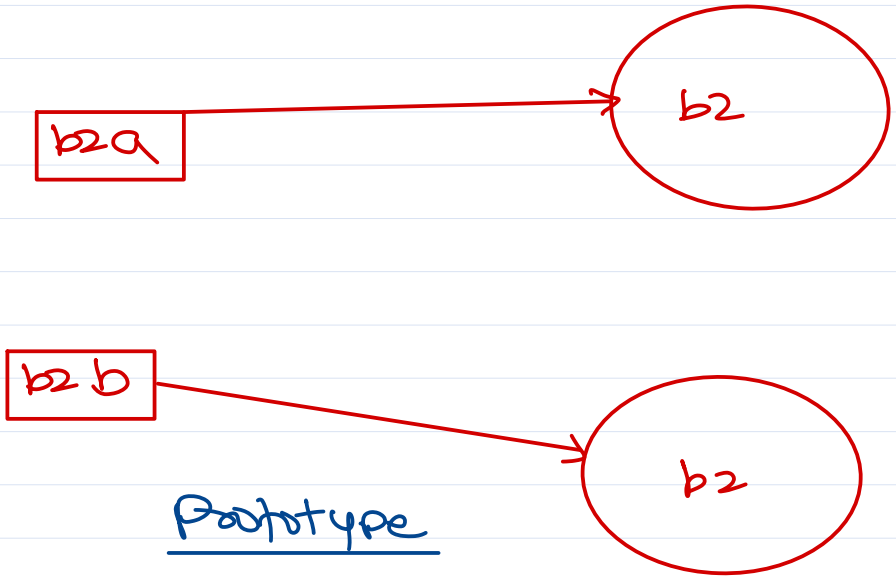
Spring bean scopes

Bean scopes

- Bean scope can be set in XML or annotation.
  - <bean id="___" class="___" scope="singleton|prototype|request|session" />
  - @Scope("singleton|prototype|request|session") → class level with stereo-type anno / method level for @Bean method.
- singleton → default scope.
  - Single bean object is created and accessed throughout the application.
  - XMLBeanFactory creates object when getBean() is called for first time for that bean.
  - ApplicationContext creates object when ApplicationContext is created.
  - For each sub-sequent call to getBean() returns same object reference.
  - Reference of all singleton beans is managed by spring container.
  - During shutdown, all singleton beans are destroyed (@PreDestroy will be called).
- prototype
  - No bean is created during startup.
  - Reference of bean is not maintained by ApplicationContext.
  - Beans are not destroyed automatically during shutdown.
  - Bean object is created each time getBean() is called.
- request and session: scope limited to current request and session. } same as java bean scopes. Possible only for web appln.
- application

Bean scopes


created when appctx is created
b1
b1a
ctx.getBean()
b1b
ctx.setBean()
Singleton bean


b2a
b2
b2b
b2
Prototype

Bean scopes

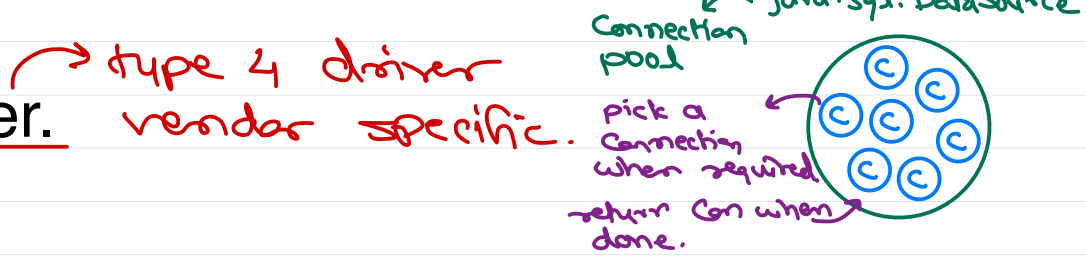
- Singleton bean inside prototype bean
  - Single singleton bean object is created.
  - Each call to getBean() create new prototype bean. But same singleton bean is autowired with them.
- Prototype bean inside singleton bean
  - Single singleton bean object is created.
  - While auto-wiring singleton bean, prototype bean is created and is injected in singleton bean.
  - Since there is single singleton bean, there is a single prototype bean.
- Need multiple prototype beans from singleton bean (solution 1)
  - Using ApplicationContextAware
  - The singleton bean class can be inherited from ApplicationContextAware interface.
  - When its object is created, container call its setApplicationContext() method and give current ApplicationContext object. This object can be used to create new prototype bean each time (as per requirement).
- Need multiple prototype beans from singleton bean (solution 2)
  - using @Lookup method
  - The singleton bean class contains method returning prototype bean.
  - If method is annotated with @Lookup, each call to the method will internally call ctx.getBean(). Hence for prototype beans, it returns new bean each time.

Spring JDBC

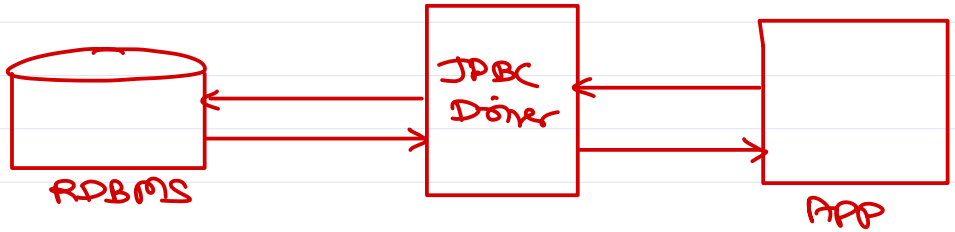
JDBC Quick Revision

tomcat : BasicDataSource
spring : HikariCP
DriverManagerDataSource

- JDBC is specification given by Sun/Oracle.
- Specification interfaces are implemented by driver.
  - Driver, Connection, Statement, ResultSet
- JDBC driver convert Java request to RDBMS understandable form and RDBMS response to Java understandable form.
- JDBC programming steps
  - Add JDBC driver into project CLASSPATH.
  - Load and register JDBC driver.
  - Create JDBC connection.
  - Prepare JDBC statement.
  - Execute query and process result.
  - Close all.



type 4 driver
vendor specific.



Classpath → add external jar
Maven / Gradle

Class.forName()
↳ optional (JDBC 4+)

DriverManager.getConnection()

Con.prepareStatement()

Select: stmt.executeQuery() → ResultSet → while(rs.next()) { ... }

non-select: stmt.executeUpdate().

Put data into Java object ← get data from rs

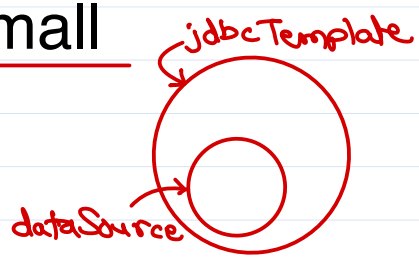
JDBC Quick Revision

- Transaction is a set of DML queries executed as a single unit. If any query fails, other queries are discarded.
- Transaction is feature of RDBMS.
- RDBMS commands: START TRANSACTION, SAVEPOINT, COMMIT, ROLLBACK.
- It follows ACID properties: Atomicity, Consistency, Isolation and Durability.
- JDBC functions
  - con.setAutocommit(false);
  - con.commit();
  - con.rollback();

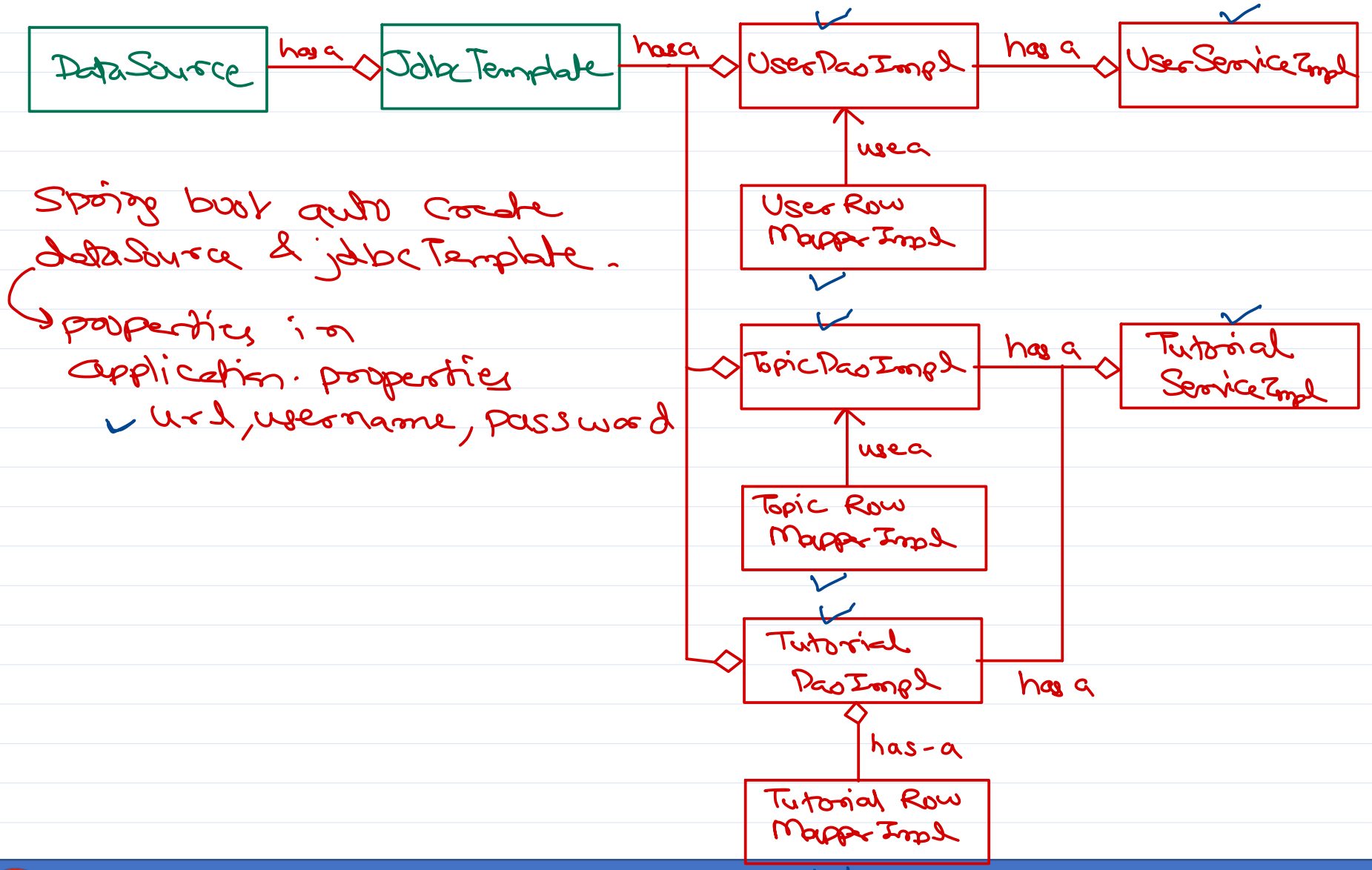
```
try {
    con.setAutocommit(false);
    // exec DML statements
    con.commit();
} catch(Exception e) {
    con.rollback();
}
```

Spring JDBC Integration

- Spring DI simplifies JDBC programming.
- Using JDBC we can avoid overheads of ORM tools. This is helpful in small applications, report generation tools or running ad-hoc SQL queries.
- Steps:
  - In pom.xml, add spring-jdbc and mysql-connector-java
  - Create dataSource bean (XML or annotation config)
  - Create JdbcTemplate bean (XML or annotation config) and attach dataSource,
  - Implement RowMapper interface in a class (for dealing with SELECT queries)
    - mapRow() convert resultset row to Java object.
  - Create Spring @Repository bean and auto-wire JdbcTemplate in it.
  - Invoke JdbcTemplate query() and/or update() for appropriate operations.
  - To use transaction management, create TransactionManager bean and use @Transactional on service layer (common practice).



Spring JDBC Integration

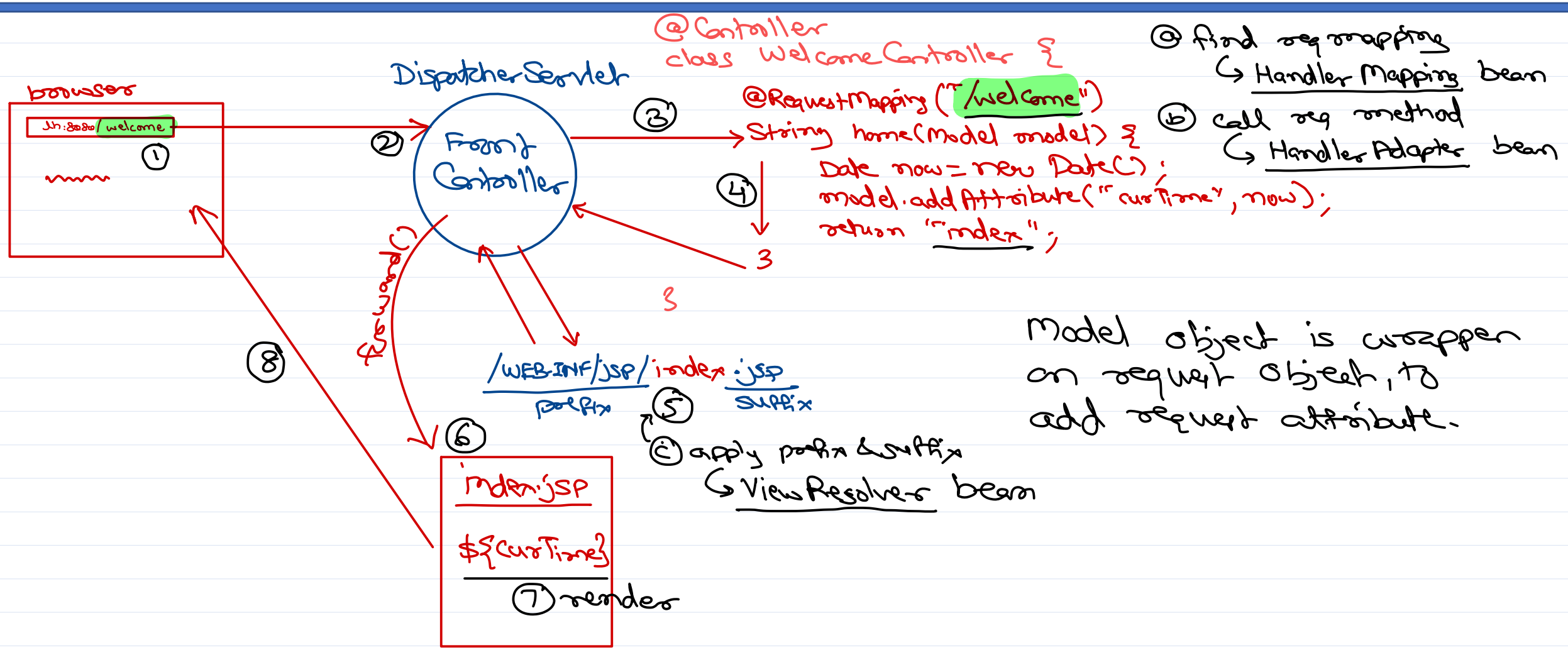


Spring Web MVC

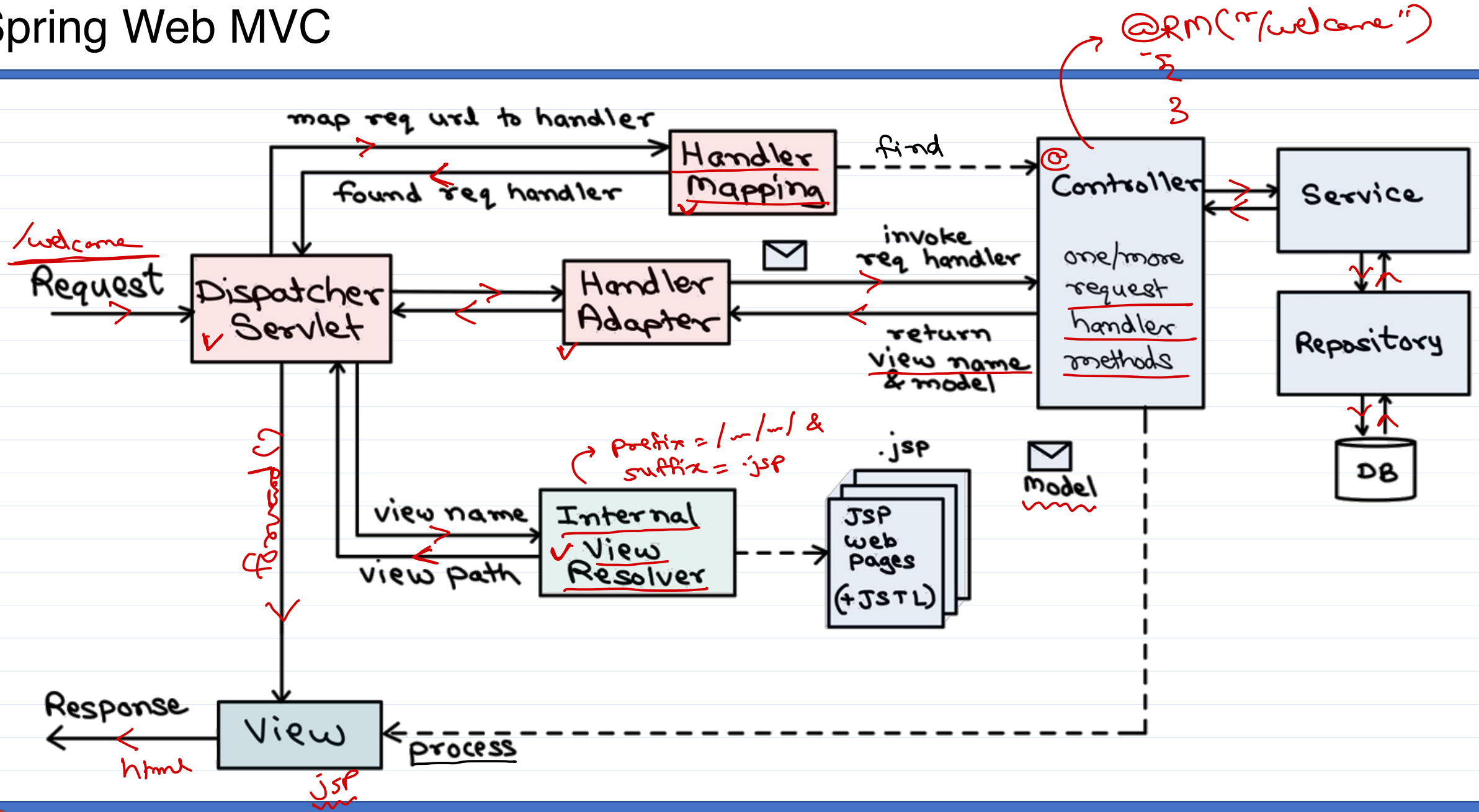
Spring Web MVC

- MVC is design-pattern.
  - Divide application code into multiple relevant components to make application maintainable and extendable.
  - M: Model: Data of the application.
  - V: View: Appearance of data.
  - C: Controller: Interaction between models & views.
- Typical MVC implementation using Servlets & JSP.
  - Model: Java beans
  - View: JSP pages
  - Controller: Servlet dispatching requests → forward()
- Spring MVC components
  - Model: POJO classes holding data between view & controller.
  - View: JSP pages / Thymeleaf / Velocity.
  - Controller: Spring Front Controller i.e. DispatcherServlet ← pre-defined by spring
  - User defined controller: Interact with front controller to collect/send data to appropriate view, process with service layer. @Controller.

Spring Web MVC



Spring Web MVC



Spring Web MVC

- DispatcherServlet receives the request.
- DispatcherServlet dispatches the task of selecting an appropriate controller to HandlerMapping. HandlerMapping selects the controller which is mapped to the incoming request URL and returns the (selected Handler) and Controller to DispatcherServlet.
- DispatcherServlet dispatches the task of executing of business logic of Controller to HandlerAdapter.
- HandlerAdapter calls the business logic process of Controller.
- Controller executes the business logic, sets the processing result in Model and returns the logical name of view to HandlerAdapter.
- DispatcherServlet dispatches the task of resolving the View corresponding to the View name to ViewResolver. ViewResolver returns the View mapped to View name.
- DispatcherServlet dispatches the rendering process to returned View.
- View renders Model data and returns the response.

- https://terasolunaorg.github.io/guideline/1.0.x/en/Overview/SpringMVCOverview.html

Spring Boot Web MVC

- Create new Spring Boot project with Spring Web starter.
- In pom.xml add JSP & JSTL support
  - org.apache.tomcat.embed - tomcat-embed-jasper: provided
  - javax.servlet - jstl
- Configure viewResolver in application.properties
  - spring.mvc.view.prefix=/WEB-INF/jsp/
  - spring.mvc.view.suffix=.jsp
- Implement a controller with a request handler method returning view name (input).
- Under src/main create directory structure webapp/WEB-INF/jsp.
- Create input.jsp under webapp/WEB-INF/jsp to input user name and submit to server.
- Add another request handler method in controller to accept request param and send result to output page.
- Create output.jsp under webapp/WEB-INF/jsp to display the result.

Thank you!

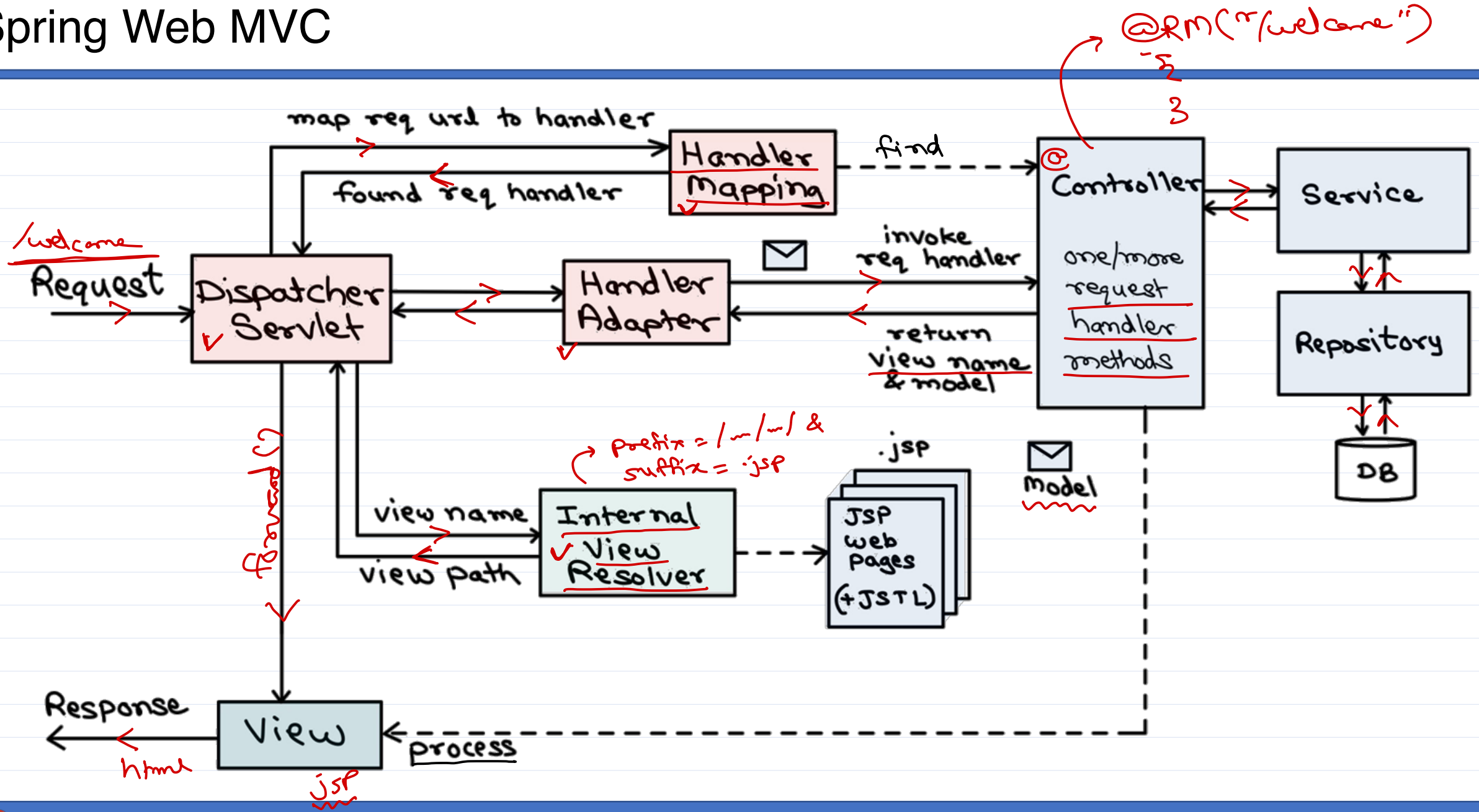
Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring and Hibernate

Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring Web MVC

Spring Web MVC



Spring Boot Web MVC

- Create new Spring Boot project with Spring Web starter.
- In pom.xml add JSP & JSTL support
  - org.apache.tomcat.embed - tomcat-embed-jasper: provided
  - javax.servlet - jstl
- Configure viewResolver in application.properties
  - spring.mvc.view.prefix=/WEB-INF/jsp/
  - spring.mvc.view.suffix=.jsp
- Implement a controller with a request handler method returning view name (input).
- Under src/main create directory structure webapp/WEB-INF/jsp.
- Create input.jsp under webapp/WEB-INF/jsp to input user name and submit to server.
- Add another request handler method in controller to accept request param and send result to output page.
- Create output.jsp under webapp/WEB-INF/jsp to display the result.

Request handler method

- @RequestMapping attributes
  - value/path = “url-pattern”
  - method = GET | POST | PUT | DELETE
  - params/header = … (map request only if given param or header is present).
  - consumes = … (map request only if given request body type is available).
  - produces = … (produce given response type from handler method)
- One request handler method can be mapped to multiple request methods.
- One request handler method can be restrict to set of request methods.
- To restrict handler method to single request method, shorthand annotations available.
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping

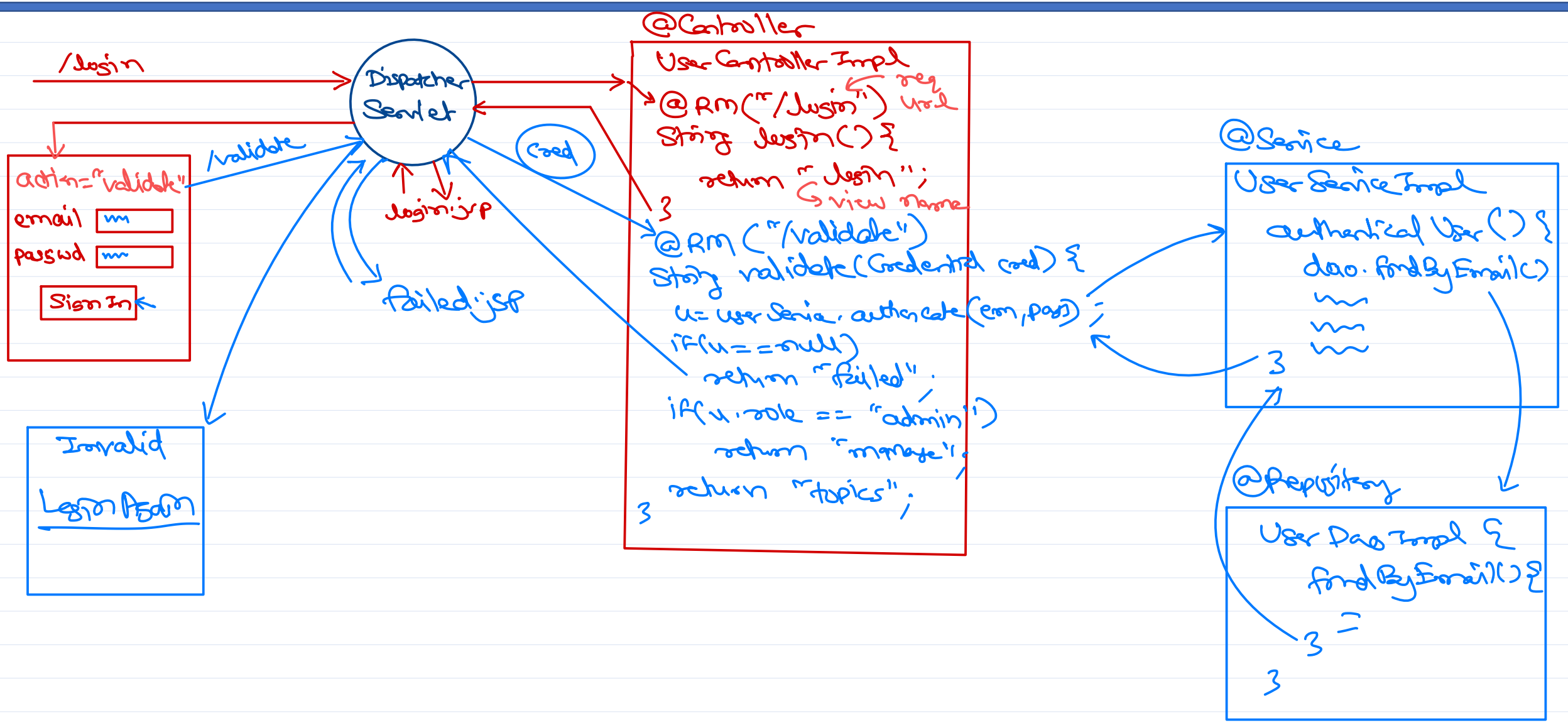
value = {"/welcome", "/"}

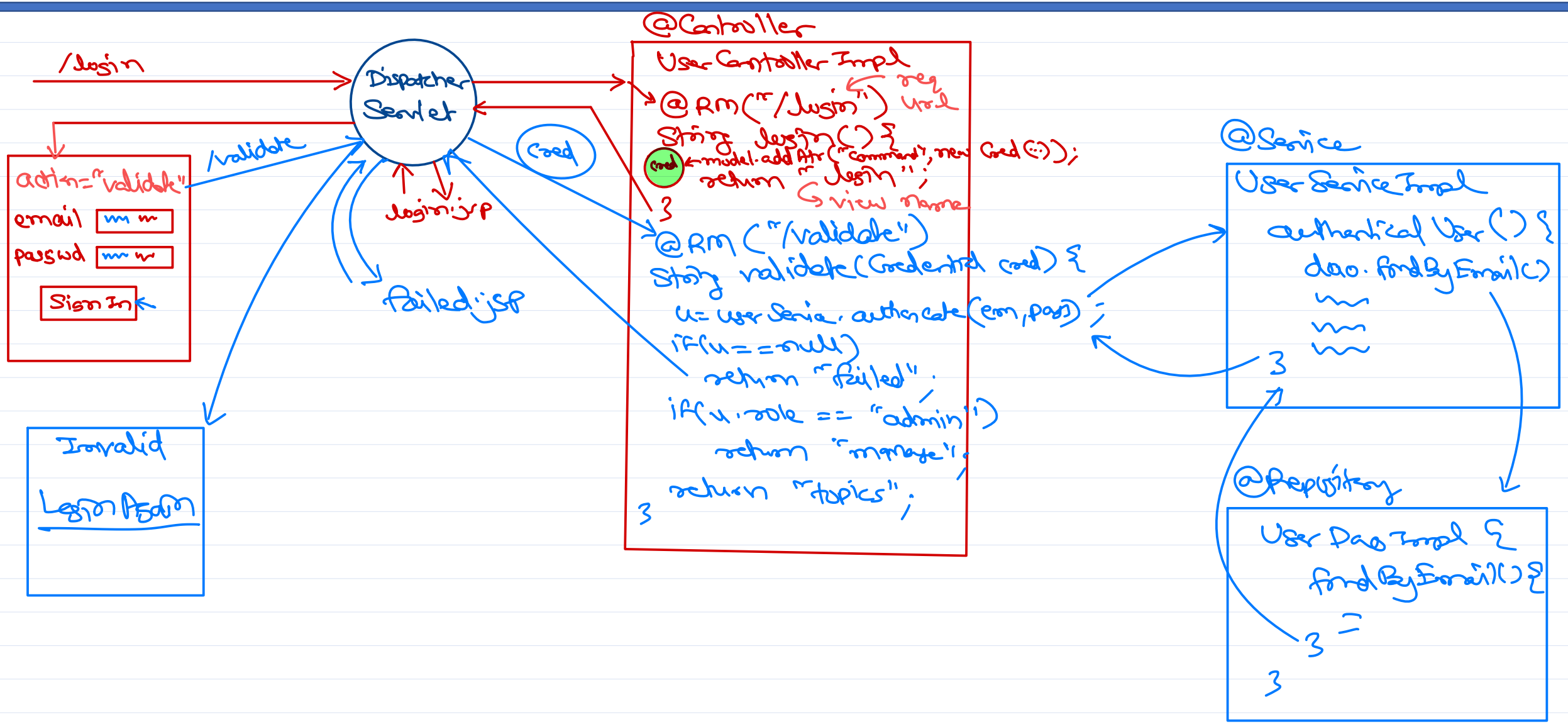
@RequestMethod(method = {GET, POST})

Request handler method

- An @RequestMapping handler method can have a very flexible signatures.
- Supported method argument types
  - HttpServletRequest, HttpServletResponse
  - @RequestParam, @RequestHeader, @PathVariable
  - Map or Model or ModelMap
  - Command object, @ModelAttribute, Errors or BindingResult
  - InputStream, OutputStream
  - HttpSession, @CookieValue, Locale
  - HttpEntity<>, @RequestBody.
- Supported method return types
  - String, View, Model or Map, ModelAndView
  - HttpHeaders, void
  - HttpEntity<>, ResponseEntity<>, @ResponseBody.

view name


/login
Dispatcher Servlet
@Controller
UserControllerImpl
@RM("/login")
String login(){
return "login";
}
@RM("/validate")
String validate(Credentials cred){
u = userServ.authenticate(em,pass);
if(u==null)
return "failed";
if(u.role == "admin")
return "manage";
return "topics";
}
action="validate"
email
passwd
Sign In
/validate
login.jsp
failed.jsp
cred
Invalid LoginPage
@Service
UserServiceImpl
authenticateUser(){
dao.findByEmail()
@Repository
UserDaoImpl{
findByEmail(){


/login
Dispatcher Servlet
@Controller
UserControllerImpl
@RM("/login")
String login(){
model.addAttr("command", new Cred());
return "login";
}
@RM("/validate")
String validate(Credentials cred){
u = userServ.authenticate(em,pass);
if(u==null)
return "failed";
if(u.role == "admin")
return "manage";
return "topics";
}
action="validate"
email
passwd
Sign In
/validate
login.jsp
failed.jsp
Cred
Invalid
LoginBean
@Service
UserServiceImpl
authenticateUser(){
dao.findByEmail()
}
@Repository
UserDaoImpl{
findByEmail(){
}
}

Using spring tags

- Spring can work well with JSP and JSTL tags.
- Spring also have its own set of custom tags mainly for HTML input form, localization and validation.
- <%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
- <sf:form modelAttribute="command" action="auth">
  - Form backing bean or command object.
    - <sf:input path="email"/>
    - <sf:password path="password"/>
- The *command* is model (POJO) object that carry data between view and controller.
- Model objects are request scoped.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring and Hibernate

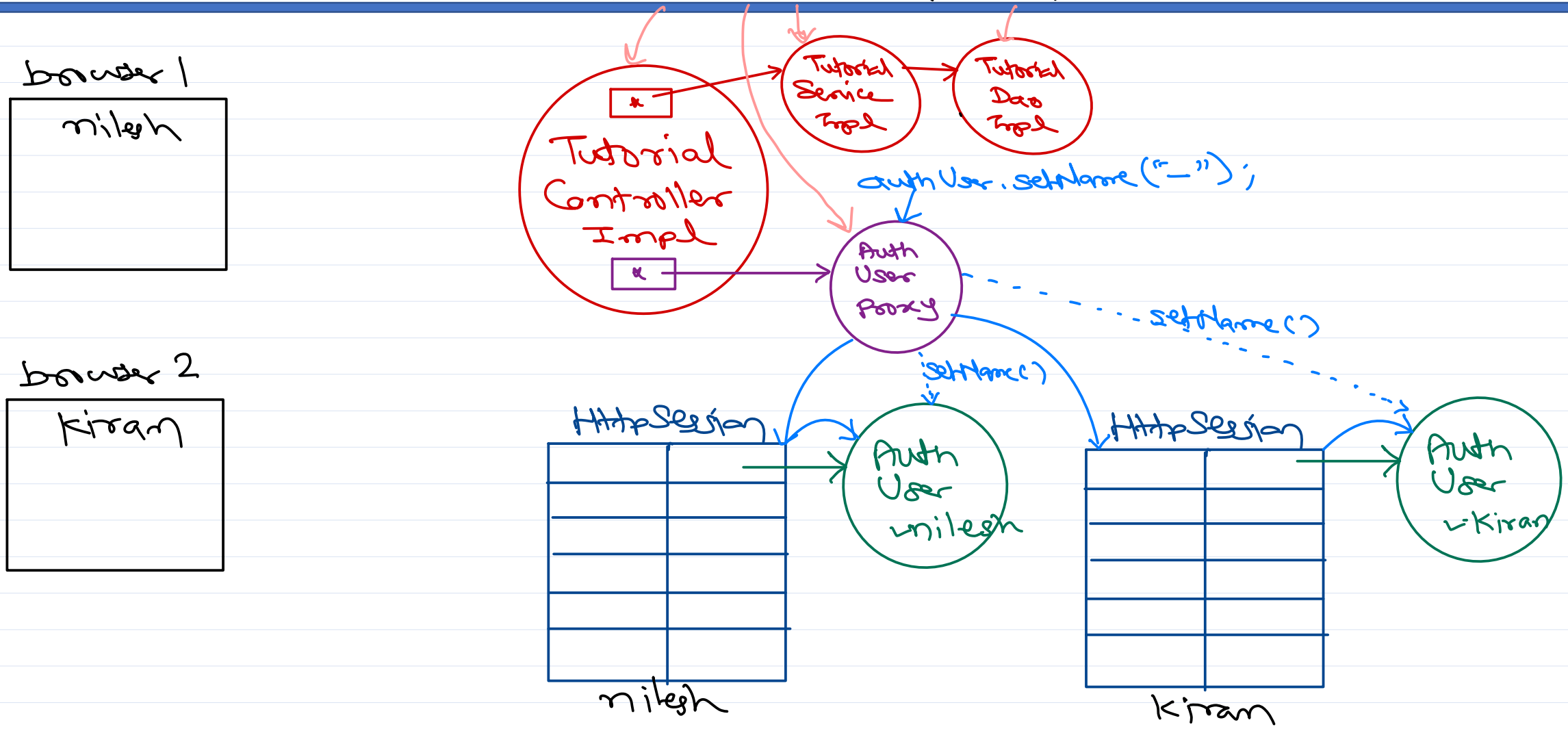
Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring Web MVC

Session and Request scoped beans

- Spring bean scopes
  - Singleton
  - Prototype
  - Session
  - Request
  - Application
- Session and Request scope beans are only possible in web applications.
- The scope management is done via proxies.
- Bean proxy is auto-wired in controller class. Depending on session/request internally new bean is created and made available in that context.

Singleton beans are created when app ctx is loaded (i.e. app start)



Spring MVC validation

- For web applications client side validations are preferred for better user experience
- However client side validations can be bypassed/skipped by client.
- To ensure only validity of data provide server side validations (as well).
- Spring validation framework helps for server side validations.
- Spring supports JSR-303 validations.
  - @NotBlank, @NotEmpty, @Size, @Email, @Pattern, @DateTimeFormat, @Min, @Max
- Steps:
  - Add dependency hibernate-validator in pom.xml
  - Use appropriate annotations on model classes.
  - Use @Valid on @ModelAttribute in request handler method.
  - Next argument in request handler should be BindingResult method.
  - Use <sf:errors/> in view to show error codes.

Spring Static resources

- Static resources like JS, CSS, images should be mapped to some location.
- <mvc:resources location="/WEB-INF/static/css/“ mapping="/css/**“/>

↳ xml config Spring mvc appln

For Spring Boot → add resources under “static” folder.

Internationalization / Localization

- Adapting computer software to different languages, regional peculiarities and technical requirements of a target locale.
- Internationalization is the process of designing a software application so that it can be adapted to various languages and regions. → messages.property
- Localization is the process of adapting internationalized software for a region or language. Localization is done for each locale by added respective components.
- Current locale can be accessed in request handler as Locale argument.
- Spring application steps → auto created in spring boot.
  - Create messageSource bean and provide base path of properties file.
  - Create properties file for different locale.
  - In views add <s:message code="property-key-name"/>.
- Current locale can be stored using SessionLocaleResolver or CookieLocaleResolver.
- Locale can be changed dynamically using LocaleChangeInterceptor (configured using <mvc:interceptors/>).

messages_hi.properties
messages_mr.properties

language	lang_country	
en	en_US	fr_FR
hi	en_GB	
mr	hi_IN	
fr	mr_IN	

Thank you!

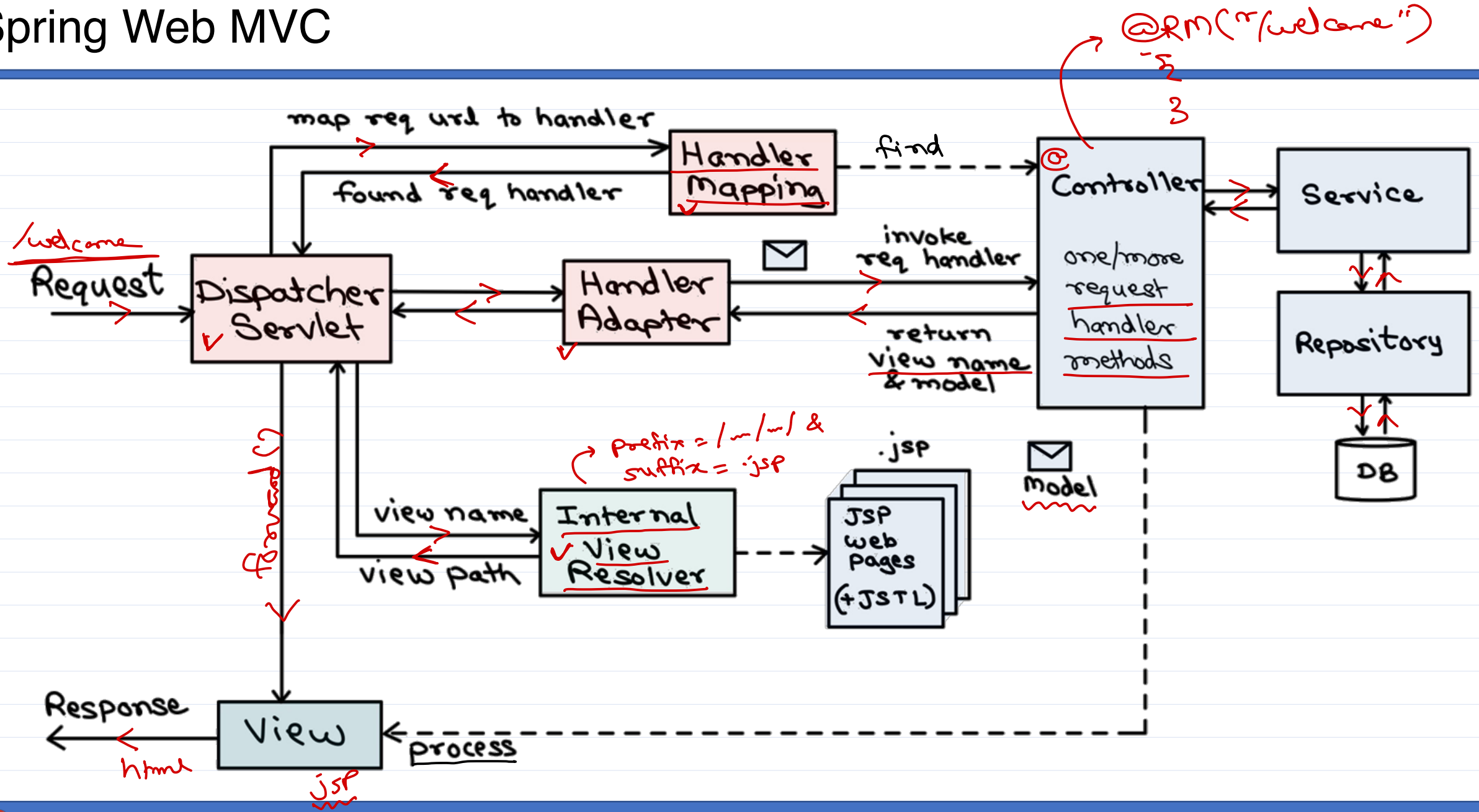
Nilesh Ghule <nilesh@sunbeaminfo.com>

Spring and Hibernate

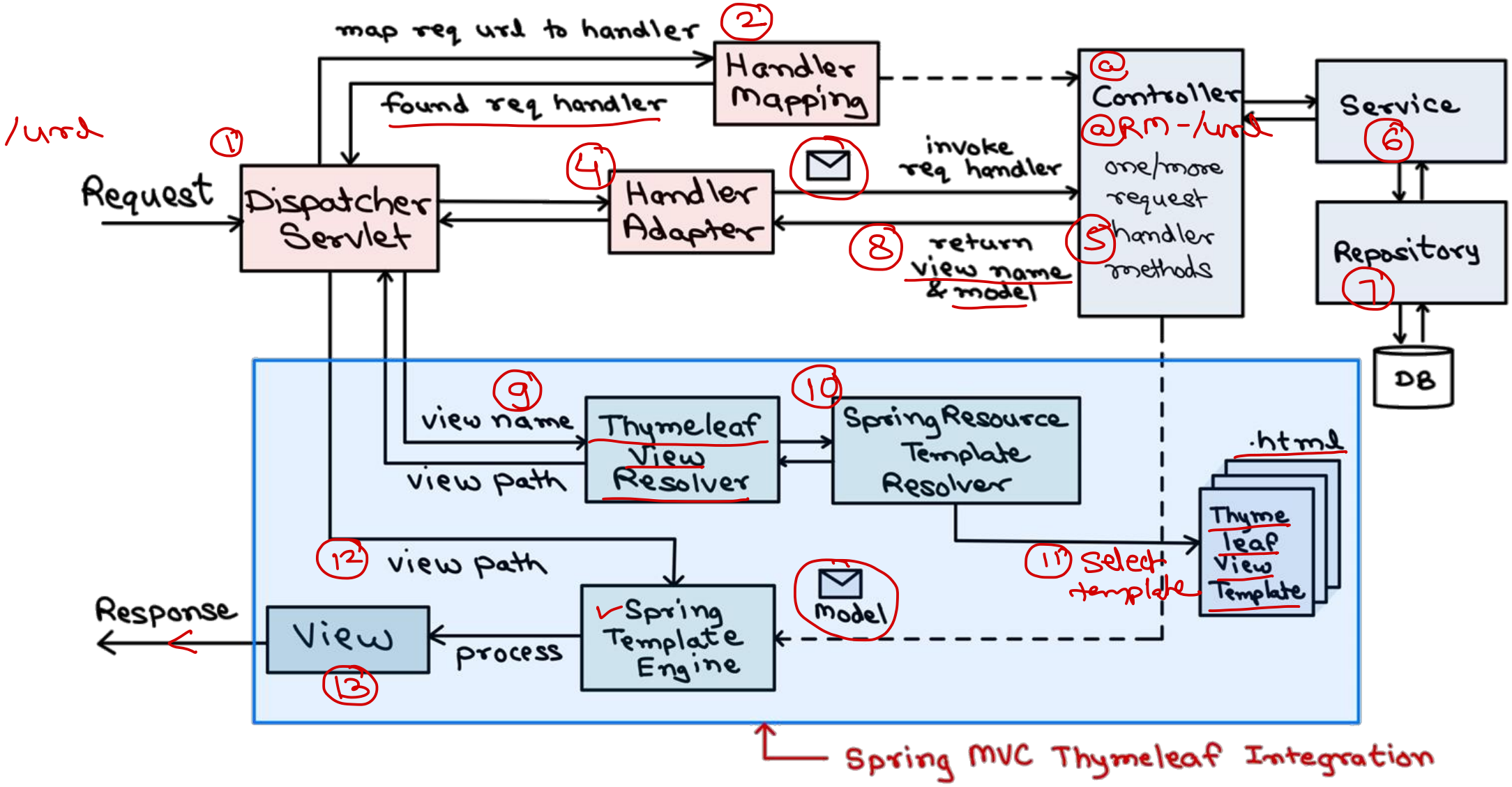
Nilesh Ghule <nilesh@sunbeaminfo.com>

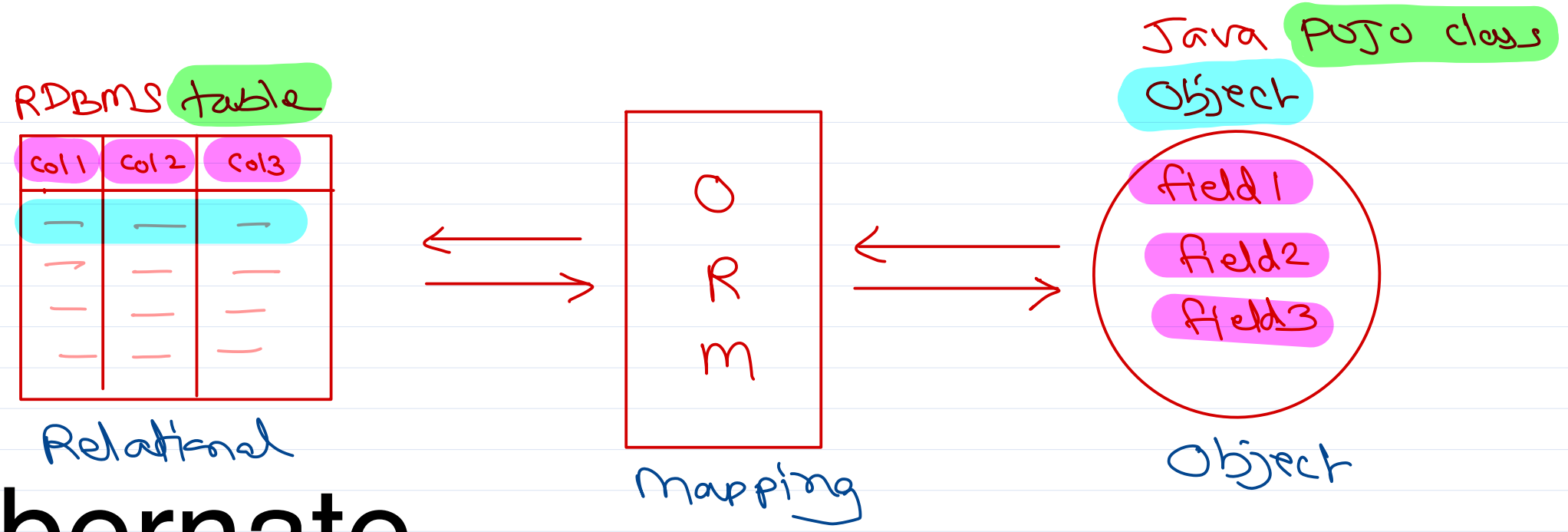
Spring Web MVC

Spring Web MVC



Spring Web MVC – Thymeleaf Views





Hibernate

ORM frameworks

1. Hibernate
2. Eclipse Link
3. iBatis
4. Torque
5. ...

JPA specification

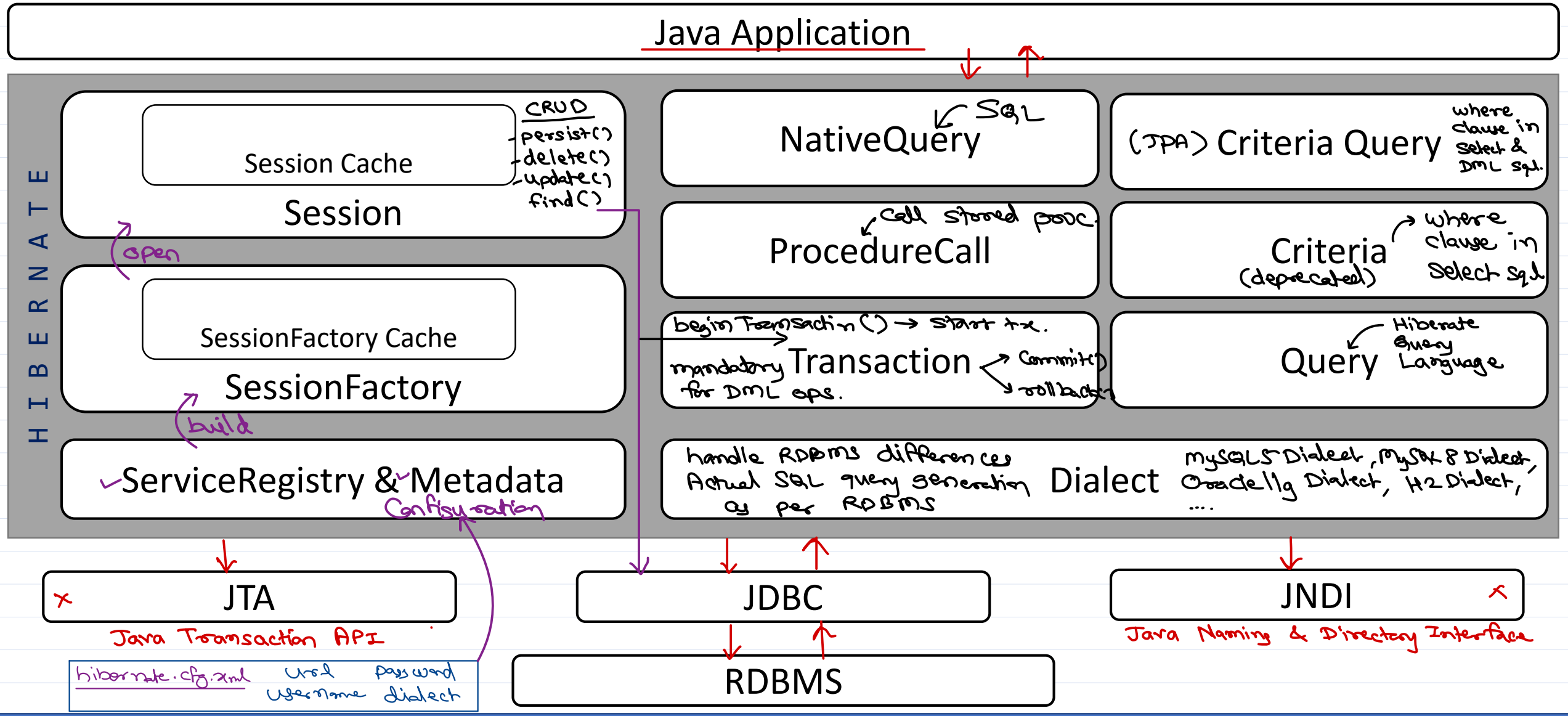
Object Relational Mapping

- Converting Java objects into RDBMS rows and vice-versa is done manually in JDBC code.
- This can be automated using Object Relational Mapping.
- Class → Table and Field → Column
- It also map table relations into entities associations/inheritance and auto-generates SQL queries.
- Hibernate is most popular ORM tool.
- Other popular ORM are EclipseLink, iBatis, Torque, ...
- **JPA is specification for ORM.**

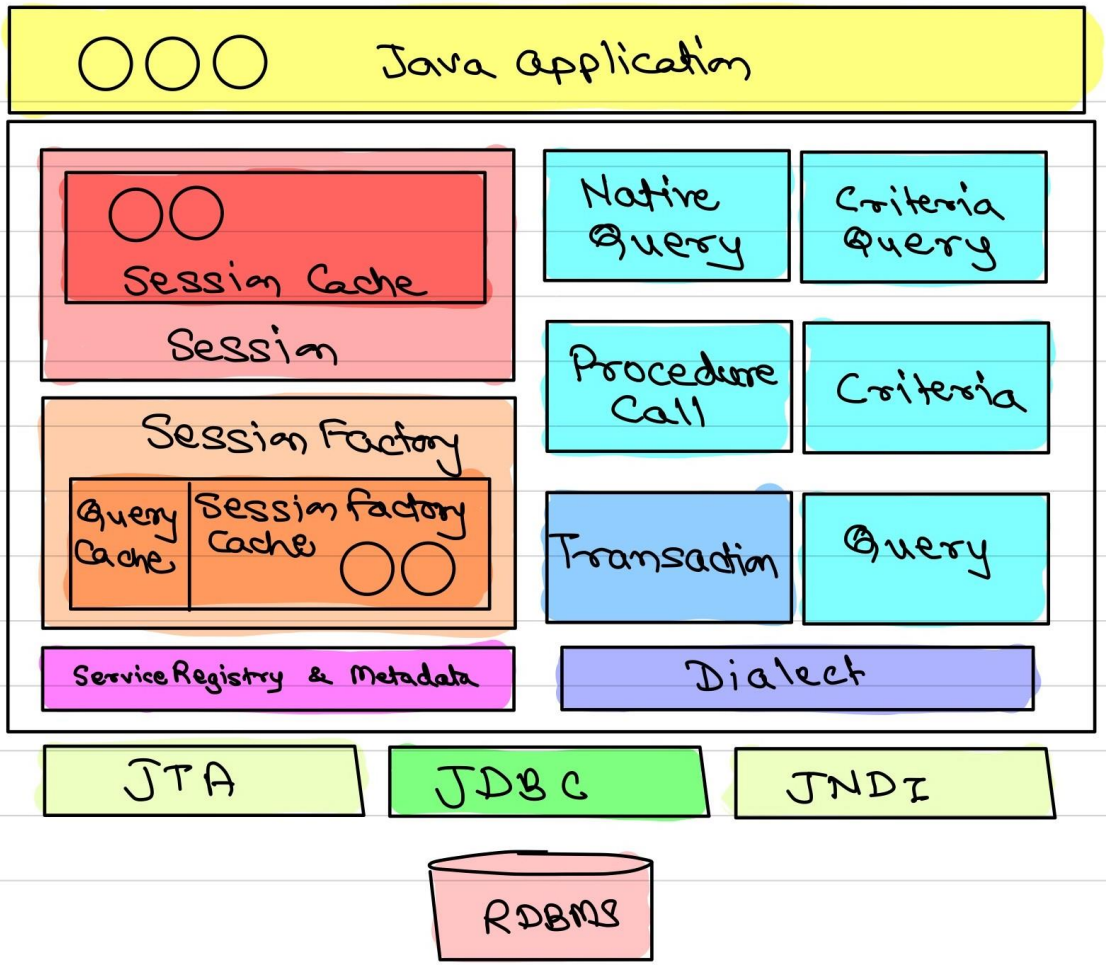
Hibernate Architecture

CRUD ops
Hibernate Associations
HQL & Autogenerated PK.

Session Factory → like → DriverManager - get connection
CRUD ops / Create query ↔ Session → wrapper → Connection - send query & get result rdbms Communication.
Transaction → wrapper → con.commit(), con.rollback()
Query → wrapper → Statement - represent query & execute it

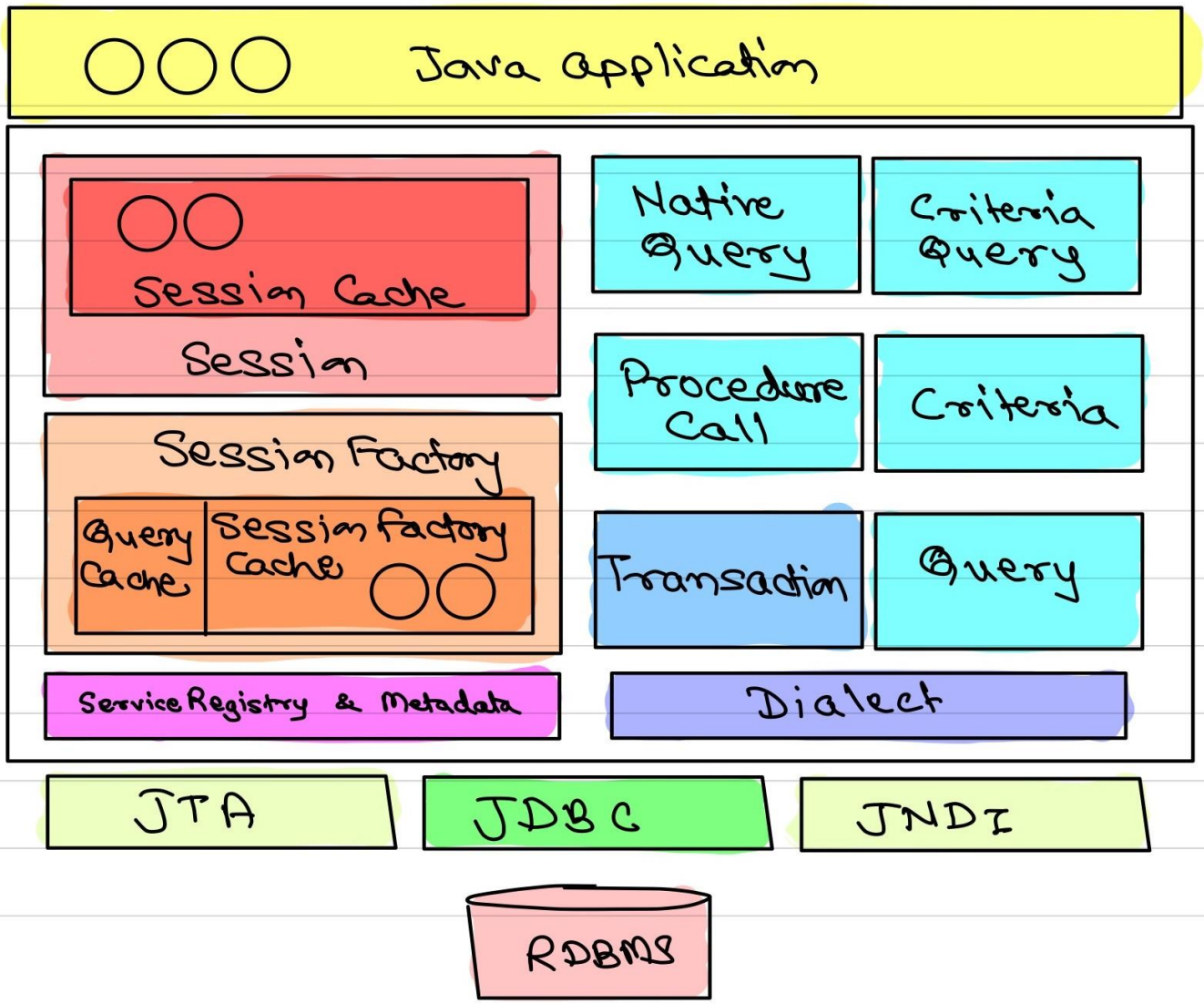


Hibernate



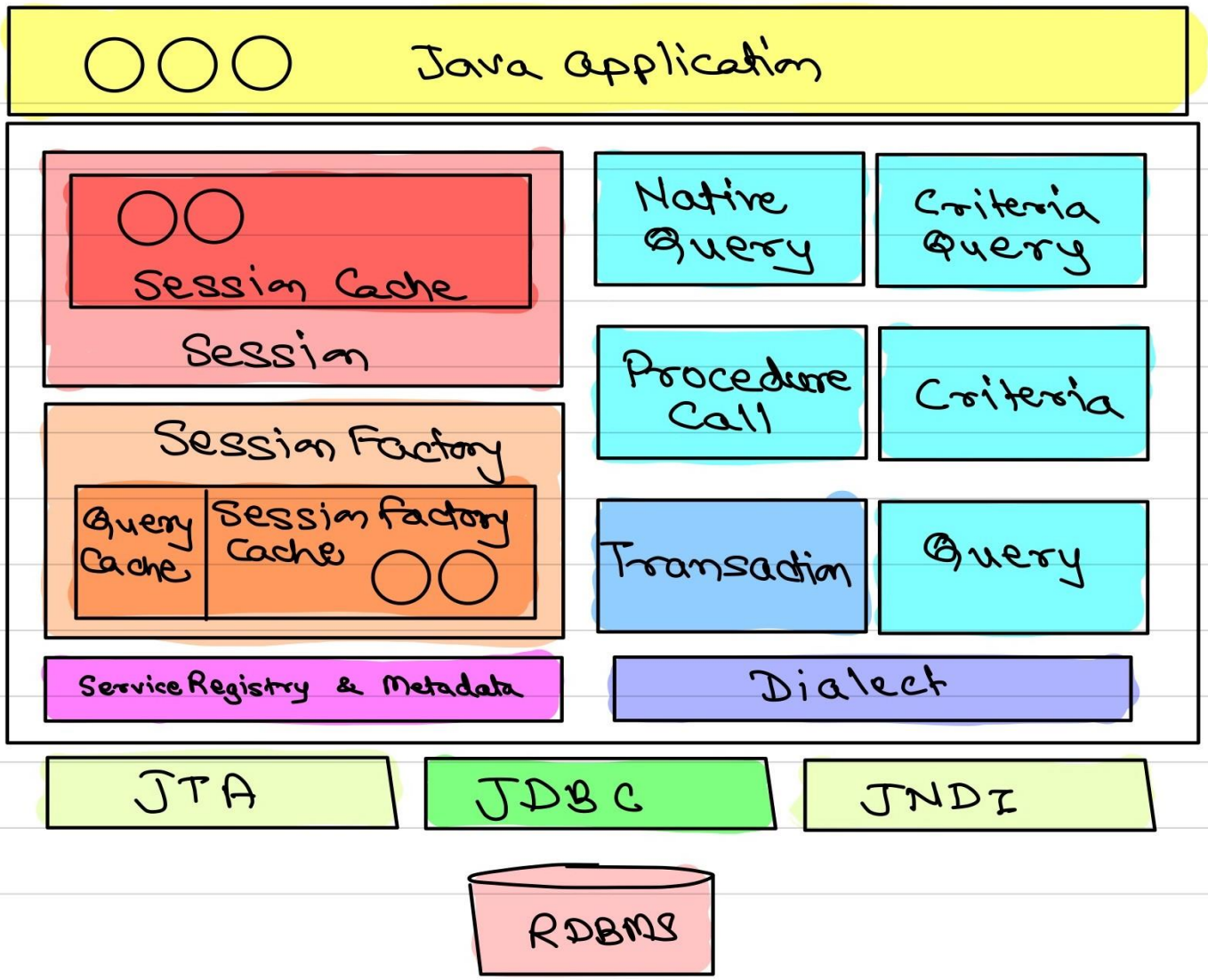
- SessionFactory
  - One SessionFactory per application (per db).
  - Heavy-weight object. Not recommended to create multiple instances. → singleton class
  - Thread-safe. Can be accessed from multiple threads (synchronization is built-in).
  - Typical practice is to create singleton utility class for that.
- Session
  - Created by SessionFactory & it encapsulate JDBC connection.
  - All hibernate operations are done on hibernate sessions.
  - Is not thread-safe. Should not access same session from multiple threads.
  - Light-weight. Can be created and destroyed as per need.

Hibernate



- Transaction
  - In hibernate, autocommit is false by default.
  - DML operations should be performed using transaction.
  - session.beginTransaction()
    - to start new transaction.
  - tx.commit()
    - to commit transaction.
  - tx.rollback()
    - to rollback transaction.
- Hibernate CRUD methods
  - get() / find()
  - save() / persist()
  - update()
  - delete()

Hibernate



- Dialect
  - RDBMS have specific features like data types, stored procedures, primary key generation, etc.
  - Hibernate support all RDBMS.
  - Most of code base of Hibernate is common.
  - Database level changes are to be handled specifically and appropriate queries should be generated. This is handled by Dialect.
  - Hibernate have dialects for all RDBMS. Programmer should configure appropriate dialect to utilize full features of RDBMS.

Hibernate 3 Bootstrapping

```
public class HbUtil {
  private static SessionFactory factory;

  static {
    try {
      Configuration cfg = new Configuration();
      cfg.configure();
      factory = cfg.buildSessionFactory();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  public static SessionFactory getSessionFactory() {
    return factory;
  }

  public static void shutdown() {
    factory.close();
  }
}
```

- Hibernate Configuration
  - hibernate.connection.driver_class
  - hibernate.connection.url
  - hibernate.connection.username
  - hibernate.connection.password
  - hibernate.dialect
  - hibernate.show_sql
- Hibernate3 Bootstrapping
  - Create Configuration object.
  - Read hibernate.cfg.xml file using its configure() method.
  - Create SessionFactory using its buildSessionFactory() method.

Hibernate 5 Bootstrapping

```
public class HbUtil {
  private static final SessionFactory factory
      = createSessionFactory();
  private static ServiceRegistry serviceRegistry;

  private static SessionFactory createSessionFactory() {
    serviceRegistry = new StandardServiceRegistryBuilder()
        .configure() // read from hibernate.cfg.xml
        .build();

    Metadata metadata = new MetadataSources(serviceRegistry)
        .getMetadataBuilder()
        .build();

    return metadata.getSessionFactoryBuilder().build();
  }
  public static void shutdown() {
    factory.close();
  }
  public static SessionFactory getSessionFactory() {
    return factory;
  }
}
```

Hibernate 5 Bootstrapping

- Create ServiceRegistry.
- Create Metadata.
- Create SessionFactory.

ServiceRegistry

- ServiceRegistry is interface.
- Some implementations are StandardServiceRegistry, BootstrapServiceRegistry, EventListenerRegistry, ...
- Add, manage hibernate services.

Metadata

- Represents application's domain model & its database mapping.

https://docs.jboss.org/hibernate/orm/4.3/topical/html/registries/ServiceRegistries.html

Hibernate

- Hibernate3 added annotations for ORM.

- ORM using annotations
  - @Entity
  - @Table
  - @Column
  - @Id
  - @Temporal
  - @Transient

- @Column can be used on field level or on getter methods.

Hibernate

- Earlier versions does ORM using .hbm.xml files.

```java
@Entity
@Table(name = "DEPT")
public class Dept implements Serializable {
        @Id //primary key
        @Column(name = "deptno")
        private int id; //deptno
        @Column(name = "dname")
        private String name; //dname
        @Column(name = "loc")
        private String loc; //loc
```

```xml
<hibernate-mapping>
 <class name="com.sunbeam.sh.Dept" table="DEPT">
   <id name="id" type="int">
     <column name="DEPTNO" />
     <generator class="assigned" />
   </id>
   <property name="name" type="java.lang.String">
     <column name="DNAME" />
   </property>
   <property name="loc" type="java.lang.String">
     <column name="LOC" />
   </property>
 </class>
</hibernate-mapping>
```

Hibernate Transaction

- In hibernate, autocommit is false by default.
- DML operations should be performed using transaction.
  - session.beginTransaction(): to start new tx.
  - tx.commit() & tx.rollback(): to commit/rollback tx.
- session.flush()
  - Forcibly synchronize in-memory state of hibernate session with database.
  - Each tx.commit() automatically flush the state of session.
  - Manually calling flush() will result in executing appropriate SQL queries into database.
  - Note that flush() will not commit the data into the RDBMS tables.
  - The flush mode can be set using session.setHibernateFlushMode(mode).
    - ALWAYS, AUTO, COMMIT, MANUAL
- If hibernate.connection.autocommit is set to true, we can use flush to force executing DML queries.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>