Sunbeam Infotech

Wake up from Hibernate, Spring up!!!

Course Introduction

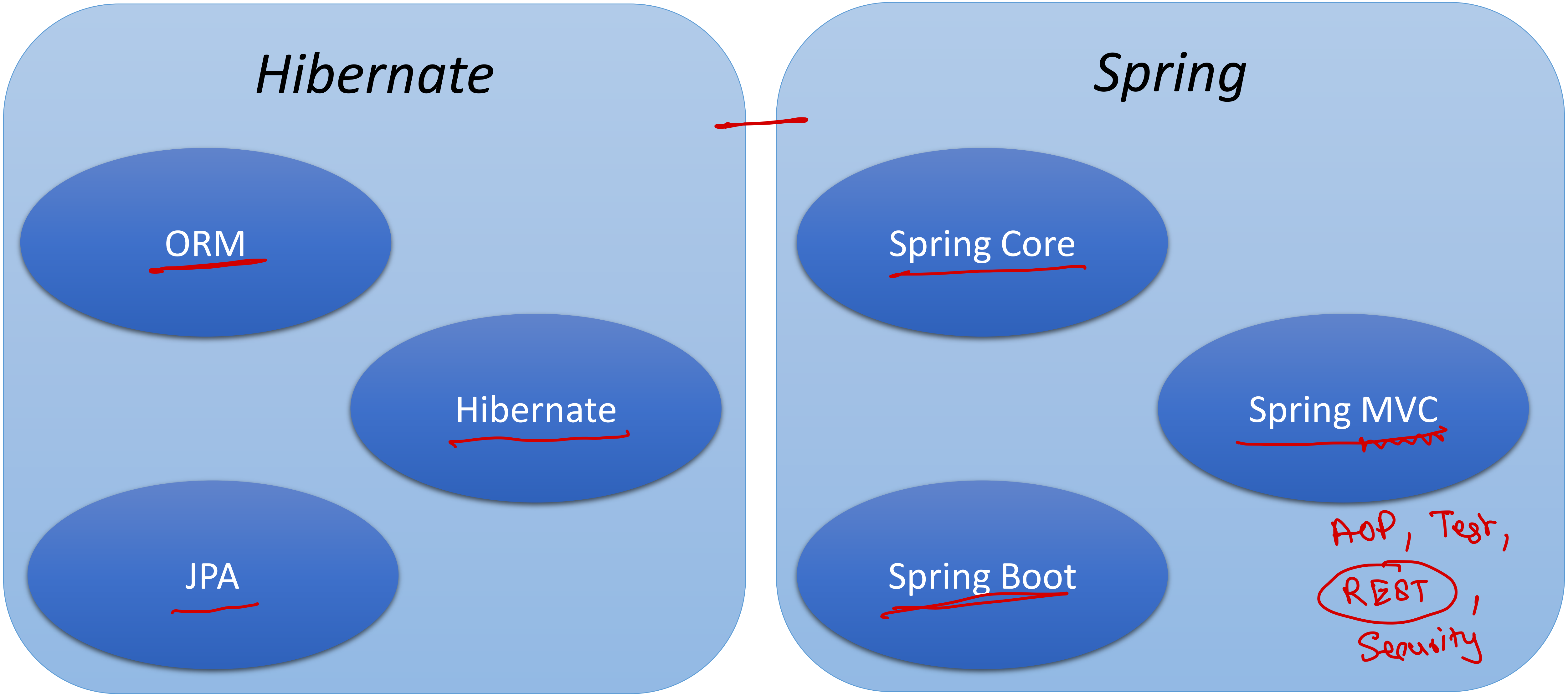
SH-09

weekday : Mon-Fri

8:00 AM to 11:00 AM.

- What is prerequisite of the course?
    - Java – OOP, Collections, Reflection, Annotation & Proxy.
    - Development – JUnit & Maven.
    - RDBMS/SQL – JDBC
    - HTML, JS, CSS – Servlets, JSP
- What should I expect from this course?
    - In depth knowledge of Hibernate/JPA (5.4.17), Spring (5.2.7)
    - Introduction to Spring Boot.
    - Hands-on experience in Hibernate and Spring.
- What is NOT covered in this course?
    - Core Java, JVM internals, SQL queries, HTML/JS coding
    - Spring Boot Micro-services and Spring Cloud
- Trainer: Mr. Nilesh Ghule
    - 14+ years experience of Java training

Course Contents



Hibernate

ORM

Hibernate

JPA

Spring

Spring Core

Spring MVC

Spring Boot

AOP, Test, REST, Security

Agenda

- Quick revision of Reflection ✓
- Understanding Java annotations ✓
- MySQL setup ✓
- Eclipse setup for Spring & Hibernate ✓
- Creating Maven project ✓
- Quick revision of JDBC ✓
- Java EE Overview ✓
- Project Idea ✓
- Object Relational Mapping ✓
- Hibernate Introduction ✓

- gitlab.com repository: https://gitlab.com/nilesh-g/sh-09 ✓

Reflection

by javac → .class

byte code
+ metadata

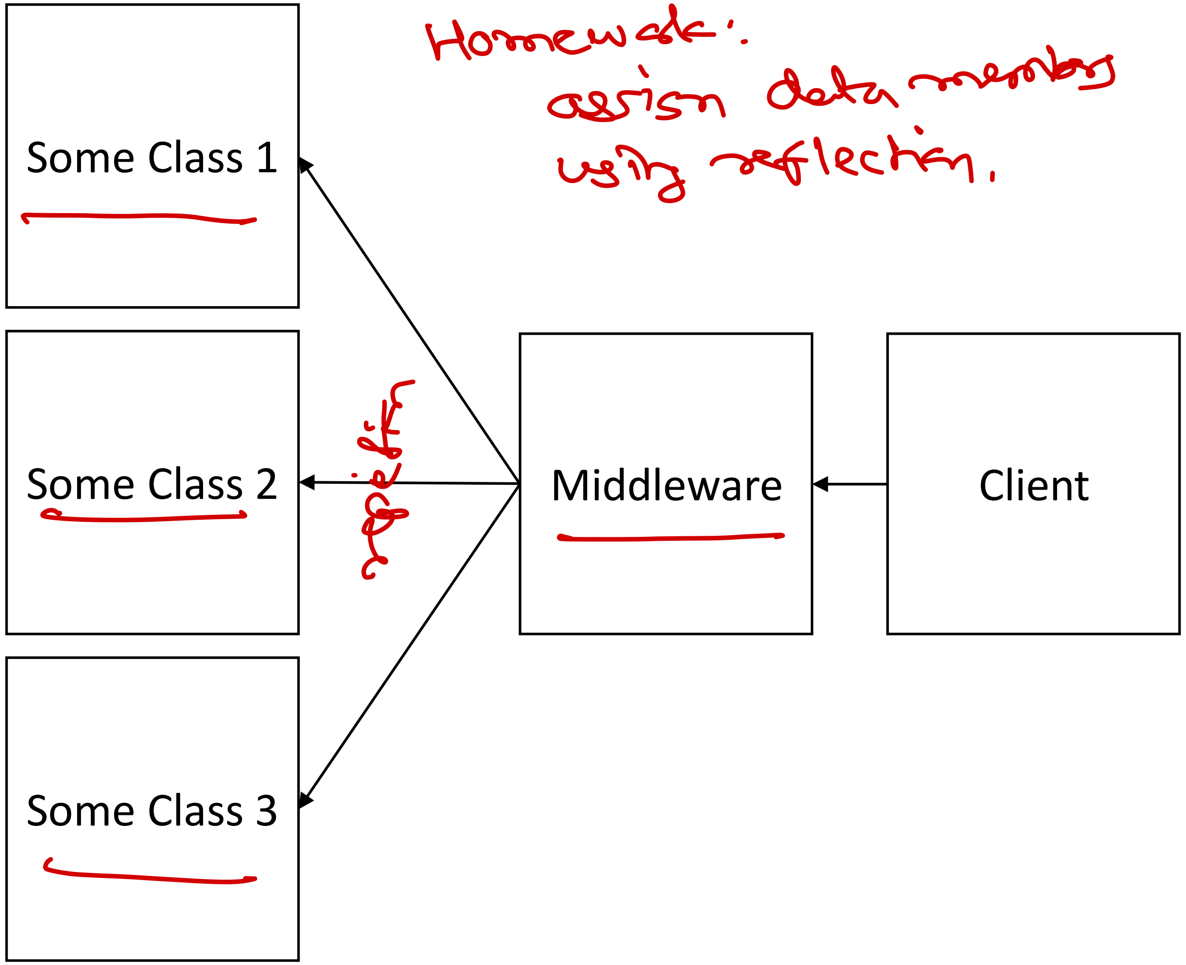
- Each class has some information (metadata) associated with it.
  - class name, super class & super interfaces
  - fields, methods, parameters and flags
- JDK *javap* tool can be used to inspect metadata associated with class.
- This information is encapsulated in java.lang.Class object.
- Three ways to take java.lang.Class object:
  - c = Class.forName("pkg.ClassName");
  - c = ClassName.class;
  - c = obj.getClass();
- getClass() is final method of Object class.

- java.lang.Class
  - static Class<?> forName(String cls);
  - String getName();
  - Class<?> getSuperclass();
  - Method[] getMethods();
  - Method[] getDeclaredMethods();
  - Method getMethod(…);
  - Field[] getFields();
  - Object newInstance();
  - InputStream getResourceAsStream(String name);

Reflection

- Using reflection object of the class can be created at runtime. Also can invoke methods of the class dynamically.
- Private members (fields & methods) of class can also be accessed using reflection.

- Middleware implementation



Homework:
assign data members
using reflection.

Annotation

- Annotations added into Java language from Java 5.0. → .Net Attributes
- Annotations is better way of associating metadata instead of Marker interface.
- Annotation can be applied at class level, field level, method level and even at parameter level.
- *value* is implicit member of annotation and *default* keyword can be used to assign default value to annotation member.

@Override

@SuppressWarning("_____")

@SuppressWarning(value="checked")

Annotation - Retention Policy

- SOURCE
  - Discarded by the compiler.
  - To give information to the compiler.
  - e.g. @Override, @Deprecated, @SuppressWarning, @FunctionalInterface
- CLASS
  - Recorded in class file, but not retained by VM at runtime.
  - Used by code processing tools.
  - e.g. @KeepName (ProGuard obfuscator), @GwtCompatible (Guava),
- RUNTIME
  - Recorded in class file & retained by VM at runtime to access using reflection.
  - e.g. @Table, @Column (hibernate), @Bean, @Value, @Autowired (spring), @WebServlet, @WebFilter (Java EE), …

Annotation

- Custom Annotation ①

```
@Retention(value=RetentionPolicy.RUNTIME)
@Target(value=ElementType.TYPE)
@interface Readme {
   String value();
   String info() default "No info";
   String author() default "Unknown";
}
```

②

class level
field
method
ctor
annotation → meta annotation

- Access annotations using reflection:
  - Annotation[] getDeclaredAnnotations();
  - Annotation getDeclaredAnnotation(Class c);

} java.lang.Class

- Accessing annotation members:
  - ann = cls.getAnnotation(Readme.class);
  - sysout(ann.value() + ", " + ann.info());

MySQL setup on Ubuntu

→ 20

- Install MySQL:
  - sudo apt-get install mysql-server mysql-client
- Start or Stop MySQL service:
  - sudo systemctl start | stop | status mysql
- Create new user & database → Ubuntu + Win + mac
  - terminal> sudo mysql –u root –p
  - mysql> CREATE USER nilesh@localhost IDENTIFIED BY ‘nilesh’;
  - password
  - mysql> CREATE DATABASE sh09;
  - mysql> GRANT ALL PRIVILEGES ON sh09.* TO nilesh@localhost;
  - mysql> FLUSH PRIVILEGES;
  - mysql> EXIT;
  - no space
  - terminal> ~~sudo~~ mysql –u nilesh –pnilesh sh09
  - mysql> SOURCE /path/to/books.sql

on windows

download mysql server 8.0.x & install.

on windows

services.msc ← Run

Eclipse Setup

STS 4.x → Spring Boot

- Ensure that JDK 1.8 is installed on system.

eclipse + Tomcat → Spring mvc

- First method: Download Eclipse STS 3.9.x:
  - https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3
  - Ubuntu: spring-tool-suite-3.9.13.RELEASE-e4.16.0-linux-gtk-x86_64.tar.gz
  - Extract and launch STS.
- Second method: Download Java/Java EE Eclipse:
  - Download ZIP from Eclipse web-site, extract and launch eclipse.
  - Go to Help → Eclipse Marketplace → Install plugins one by one.
    - Eclipse Enterprise Java Developers Tools 3.17
    - Spring Tools 3 (Standalone Edition) 3.9.13.RELEASE
    - JBoss Tools 4.15.0.Final → Hibernate Tools

Maven

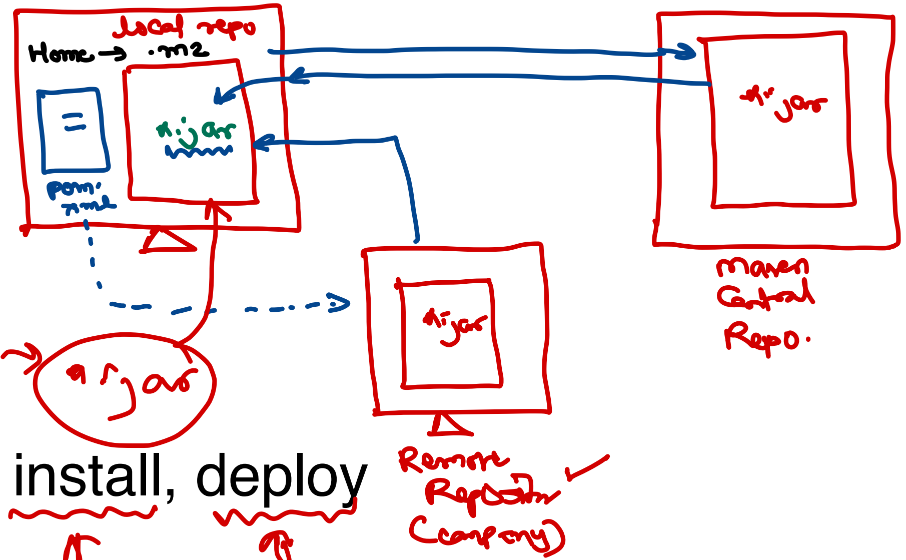
mvn package

mvn install

- 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

  project object model
  - Configurations, Dependencies, Build plugins.
- Maven build life cycles: default or clean

  target → *.jar
- Maven build phases: validate, compile, test, package, install, deploy



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

JDBC Quick Revision

interfaces.

com.mysql.cj.jdbc.Driver

- JDBC is specification given by Sun/Oracle. → mysql-connector-j.jar
- Specification interfaces are implemented by driver.
  - Driver, Connection, Statement, ResultSet → classes

jar

- 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. → maven pom.xml
  - ✓ Load and register JDBC driver. → Class.forName()
  - ✓ Create JDBC connection.
  - ✓ Prepare JDBC statement.
  - ✓ Execute query and process result.
    - executeQuery() → ResultSet
    - executeUpdate() → int
  - ✓ Close all.
- Further topics: Transactions
  - con.setAutoCommit(false)
  - con.commit()
  - con.rollback().

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>