Problem 1.

**public** **class** Demo {

**public** **static** **void** main(String args[])

{

String[] list = **new** String[args.length];

**int** index = 0;

**while** (index < args.length) {

list[index] = args[index];

index++;

}

// Check all the parameters

**for** (**int** i = 0; i < list.length; i++)

{

**if** (list[i].equals("-help"))

{

System.***out***.println("Help");

}

**else** **if** (list[i].equals("-cp"))

{

System.***out***.println("Setting cp");

}

}

}

}

We must take total argument size while creating array.

While iteration index must be post increment after intilization.

Problem 2.

Error: Cannot make a static reference to the non-static field my\_member\_variable

Solution:

**public** **class** Demo {

**public** **static** String *my\_member\_variable* = "somedata";

**public** **static** **void** main (String args[])

{

System.***out***.println ("This generates a compiler error" +*my\_member\_variable*);

}

}

Made member variable static

Problem 3.

Solution:

**package** model;

**public** **class** Threaddemo {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

Object obj1 = **new** Object();

Object obj2 = **new** Object();

Object obj3 = **new** Object();

Thread t1 = **new** Thread(**new** SyncThread(obj1, obj2), "t1");

Thread t2 = **new** Thread(**new** SyncThread(obj2, obj3), "t2");

Thread t3 = **new** Thread(**new** SyncThread(obj3, obj1), "t3");

t1.start();

Thread.*sleep*(500);

t1.join();

t2.start();

Thread.*sleep*(500);

t2.join();

t3.start();

t3.join();

}

}

**class** SyncThread **implements** Runnable {

**private** Object obj1;

**private** Object obj2;

**public** SyncThread(Object o1, Object o2) {

**this**.obj1 = o1;

**this**.obj2 = o2;

}

@Override

**public** **void** run()

{

String name = Thread.*currentThread*().getName();

System.*out*.println(name + " acquiring lock on " + "obj1");

**synchronized** (obj1) {

System.*out*.println(name + " acquired lock on " + "obj1");

work();

//Thread.currentThread().join();

System.*out*.println(name + " acquiring lock on " + "obj2");

**synchronized** (obj2)

{

System.*out*.println(name + " acquired lock on " + "obj2");

work();

}

System.*out*.println(name + " released lock on " + "obj2");

}

System.*out*.println(name + " released lock on " + "obj1");

System.*out*.println(name + " finished execution.");

}

**private** **void** work()

{

**try** {

Thread.*sleep*(30000);

//

} **catch** (InterruptedException e) {

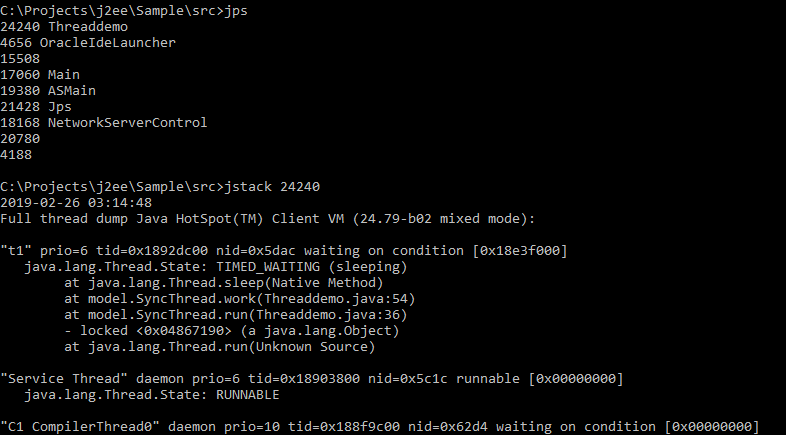
e.printStackTrace();

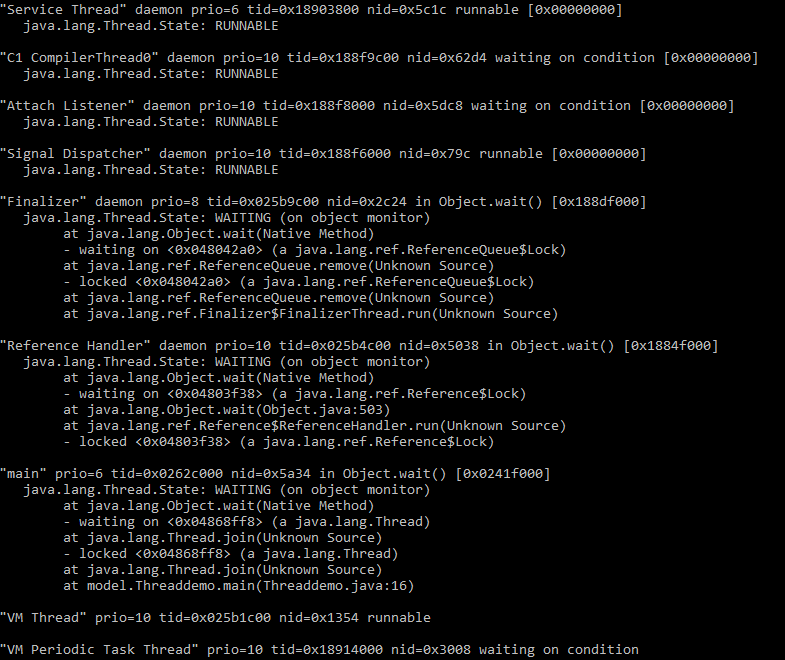
}

}

}

Jstack





Problem 4.

As this cron scheduled to run 10 mins the first one will work but successive wont as we are not closing em manager. We need to close the em manager in finally block

@PersistenceUnit

**private** EntityManagerFactory entityManagerFactory;

@Scheduled(cron = "0 0/10 \* \* \* \*")

**private** **void** scheduledOperation() {

**int** rows = 0;

**try** {

EntityManager em = entityManagerFactory.createEntityManager();

em.getTransaction().begin();

rows = em.createNativeQuery("UPDATE table SET ...").executeUpdate();

em.getTransaction().commit();

} **catch** (Exception ex) {

logger.error("Exception while scheduledOperation. Details: " +

ex.getMessage());

}

**finally** {

**if**(em!=**null**)

em.close();

}

DateTime now = **new** DateTime(DateTimeZone.UTC);

logger.info("Scheduled operation completed. Rows affected: {}. UTC time: {}",

rows, now);

}

Problem 5.

We need to close the connection and to support multithreading we must create new objects on each call.

Problem 6.

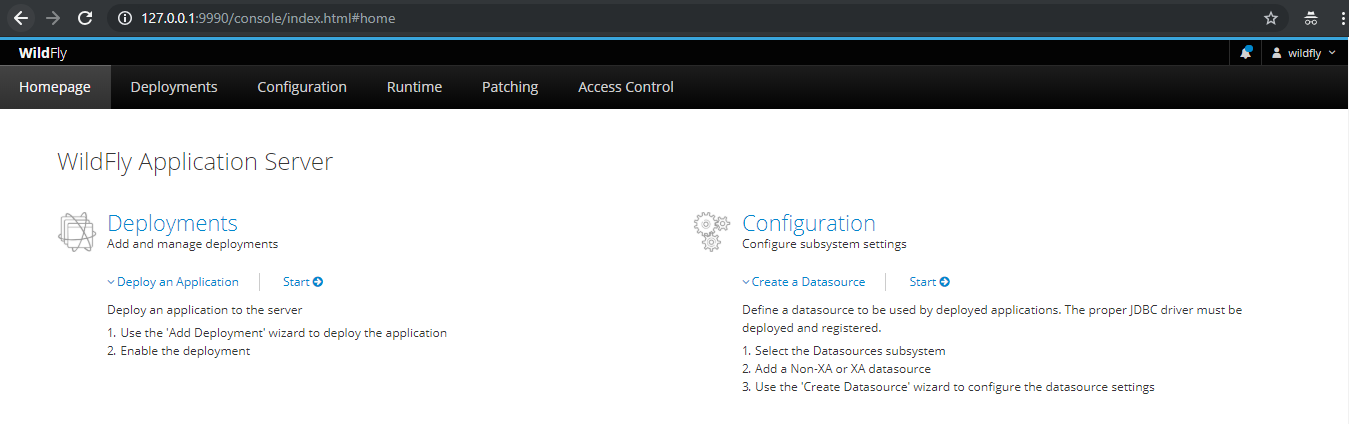
Please find the attached war that needs to be deployed on wildfly. Below is the step by step process to deploy war file.

Notes

1. The war is maven build.
2. Junit is testcase is under src/test/java BookServiceImplTest.java
3. Bookstore.xml is located in : C:\wildfly-14.0.1.Final\standalone\configuration of the local server.
4. The property path is: System.*getProperty*("jboss.server.config.dir")+"\\"+path; where path is bookstore xml file name.
5. Database used is postgres. Details can be found in hibernate.cfg.xml

Deployments

Login to wildfly administration console

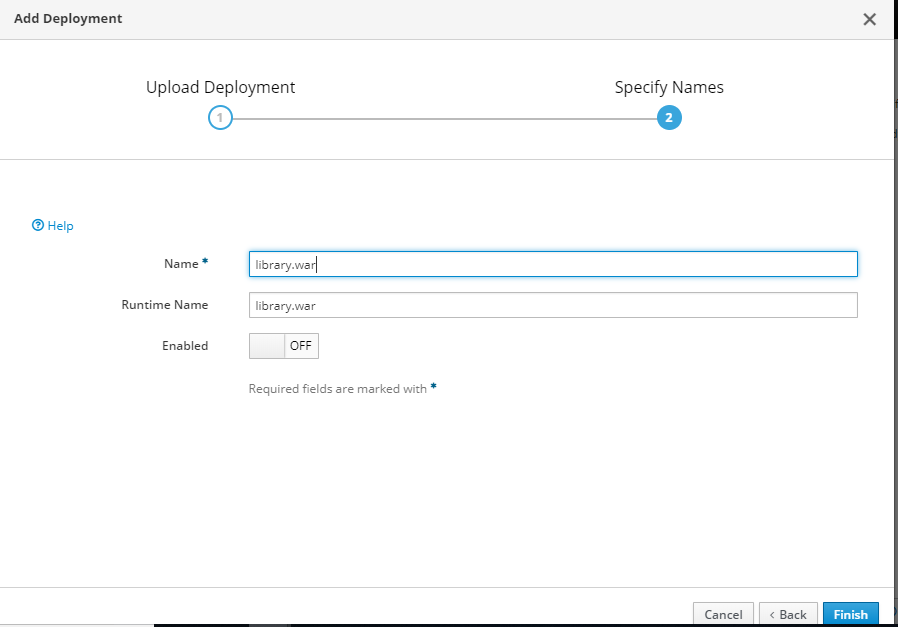


Click to deployments and then upload deployments.

Upload the war file. Create the war file by using maven build (On eclipse right click maven build)

Or else from command line : maven clean install

Post the war is created than upload the war.



Click finish and deployment is done.

You can enable the war deployment and can access the application.