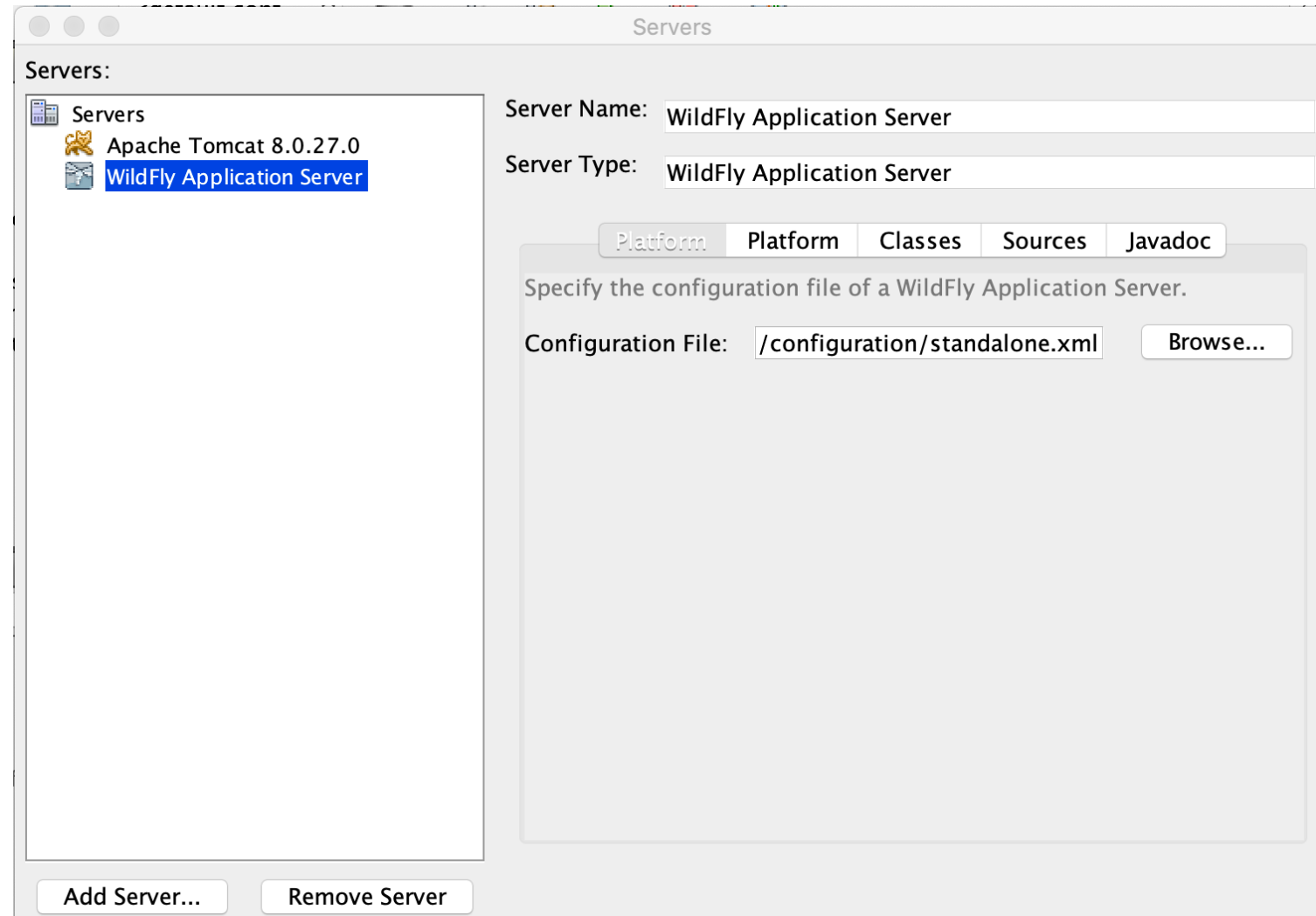


# How to configure Wildfly with Netbeans

## Part 1: Stateless (and Stateful) beans

# Configure the server

In Netbeans, go to Tools->Servers and add Wildfly  
Check which is the active configuration file, or choose one (e.g. standalone.xml)



# The remote interface

included both in the client and in the server!

```
package beans;  
import javax.ejb.Remote;
```

```
@Remote
```


```
public interface SessionBeanRemote {  
    String hello();  
}
```

## Our stateless bean (in the server)

```
package beans;
import javax.ejb.Remote;
import javax.ejb.Stateless;
@Stateless
@Remote(SessionBeanRemote.class)
public class SessionBean implements SessionBeanRemote {
    @Override
    public String hello() {
        return "ciao";
    }
}
```

# Hint: How do I find the right JNDI name?

Alternative 1: look at the starting log of the server, **choose the "ejb" one**



```
17:31:14,997 INFO [org.jboss.as.server.deployment] (MSC service thread 1-7) WFLYSRV0027: Starting deployment
17:31:15,010 INFO [org.jboss.weld.deployer] (MSC service thread 1-4) WFLYWELD0003: Processing weld deployment
17:31:15,019 INFO [org.jboss.as.ejb3.deployment] (MSC service thread 1-4) WFLYEJB0473: JNDI bindings for ses:

    java:global/SessionBeansEA-ejb/SessionBean!beans.SessionBeanRemote
    java:app/SessionBeansEA-ejb/SessionBean!beans.SessionBeanRemote
    java:module/SessionBean!beans.SessionBeanRemote
    java:jboss/exported/SessionBeansEA-ejb/SessionBean!beans.SessionBeanRemote
    ejb:/SessionBeansEA-ejb/SessionBean!beans.SessionBeanRemote
    java:global/SessionBeansEA-ejb/SessionBean
    java:app/SessionBeansEA-ejb/SessionBean
    java:module/SessionBean

17:31:15,052 INFO [io.smallrye.metrics] (MSC service thread 1-5) MicroProfile: Metrics activated (SmallRye Me
17:31:15,120 INFO [org.jboss.as.server] (DeploymentScanner-threads - 2) WFLYSRV0016: Replaced deployment "Se
```

# How do I find the right JNDI name?

<https://docs.jboss.org/author/display/WFLY10/EJB%20invocations%20from%20a%20remote%20client%20using%20JNDI.html>

Alternative 2: Compose it following a rule:

```
// The app name is the application name of the deployed EJBs. This is typically the ear name
// without the .ear suffix. However, the application name could be overridden in the application.xml
// of the EJB deployment on the server.
// If haven't deployed the application as a .ear, the app name for us will be an empty string
    final String appName = "";

// This is the module name of the deployed EJBs on the server. This is typically the jar name of the
// EJB deployment, without the .jar suffix, but can be overridden via the ejb-jar.xml
// In this example, we have deployed the EJBs in a jboss-as-ejb-remote-app.jar, so the module name is
// jboss-as-ejb-remote-app
    final String moduleName = "jboss-as-ejb-remote-app";

// AS7 allows each deployment to have an (optional) distinct name. We haven't specified a distinct name for
// our EJB deployment, so this is an empty string
    final String distinctName = "";

// The EJB name which by default is the simple class name of the bean implementation class
    final String beanName = CalculatorBean.class.getSimpleName();

// the remote view fully qualified class name -
// add a ?stateful string as the last part of the jndi name for stateful bean lookup
    final String viewClassName = RemoteCalculator.class.getName();

// let's do the lookup
    return (RemoteCalculator) context.lookup("ejb:" + appName + "/" + moduleName + "/" + distinctName +
        "/" + beanName + "!" + viewClassName);
```

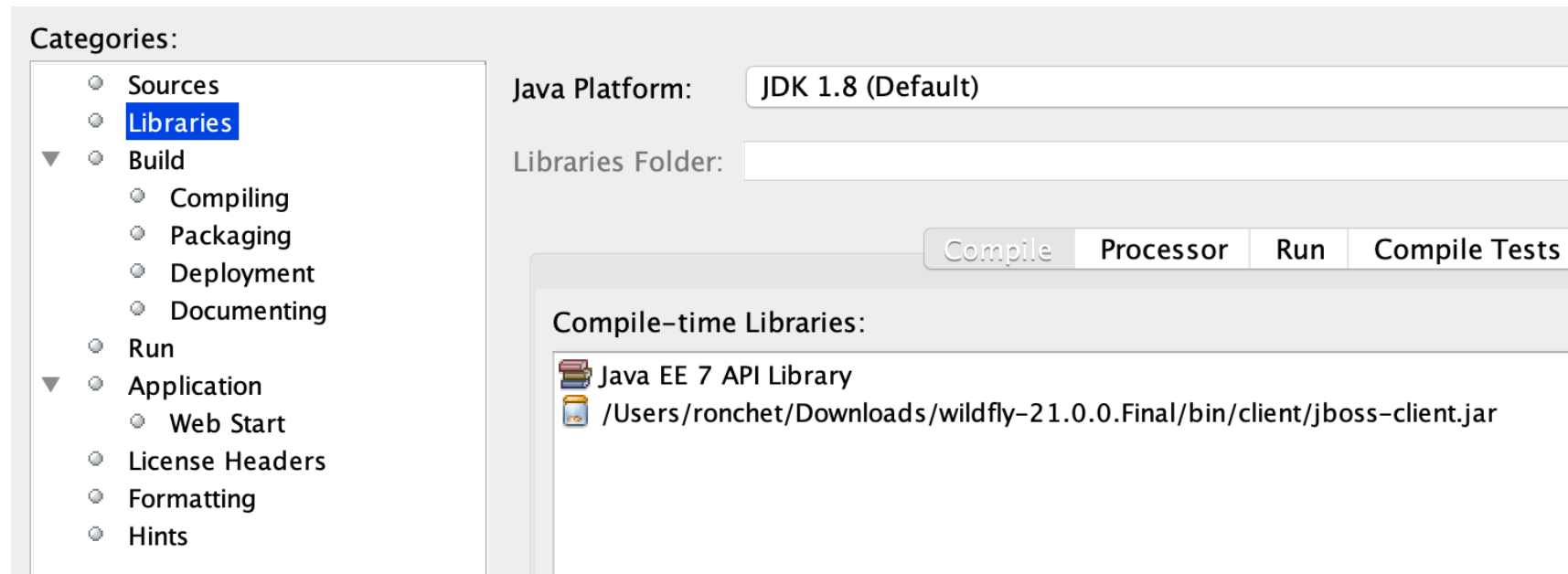
```
package clientea;

import java.util.logging.Level; import java.util.logging.Logger; import javax.naming.Context;
import javax.naming.InitialContext; import javax.naming.NamingException; import
beans.SessionBeanRemote; import java.util.Hashtable; import javax.ejb.Stateless;

public class ClientHello {
    public ClientHello() {
        final Hashtable jndiProperties = new Hashtable();
        jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY,
            "org.wildfly.naming.client.WildFlyInitialContextFactory");
        jndiProperties.put(Context.PROVIDER_URL,
            "http-remoting://localhost:8080");
        Context ctx=null; SessionBeanRemote hello=null;
        try {
            ctx = new InitialContext(jndiProperties);
            System.out.println("before");
            hello = (SessionBeanRemote) ctx.lookup(
                "ejb:/SessionBeansEA-ejb/SessionBean!beans.SessionBeanRemote");
        } catch (NamingException ex) {
            Logger.getLogger(ClientHello.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.println(hello.hello());
        System.out.println("after");
    }
}
```

Our client

# Hint 1: Check the client libraries!



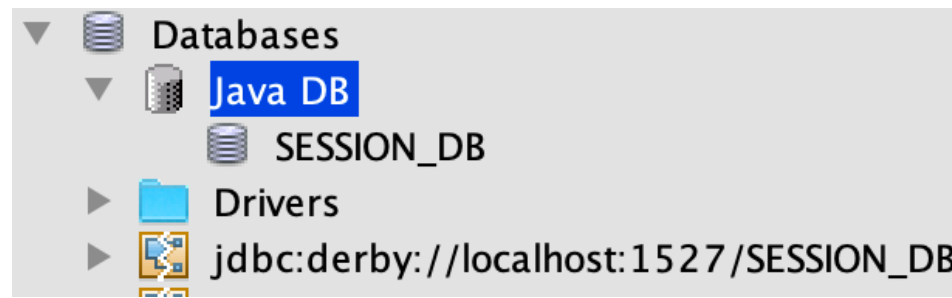
Make sure you use the Java EE 7 API, not the version 6!  
Make sure you include the jboss-client.jar



# How to configure Wildfly with Netbeans

## Part 2: Datasources and Entities

# Let's take a look at our DB



- name: SESSION\_DB
- connection: jdbc:derby://localhost:1527/SESSION\_DB

```
<datasources>
```

```
  <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enabled="true" use-java-context="true" statistics-  
enabled="{wildfly.datasources.statistics-enabled:{wildfly.statistics-enabled:false}}">
```

```
    <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE</connection-url>
```

```
    <driver>h2</driver>
```

```
    <security>
```

```
      <user-name>sa</user-name>
```

```
      <password>sa</password>
```

```
    </security>
```

```
  </datasource>
```

```
  <datasource jndi-name="java:jboss/datasources/DerbyDS" pool-name="DerbyDS" enabled="true" use-ccm="false">
```

```
    <connection-url>jdbc:derby://localhost:1527/SESSION_DB;create=true</connection-url>
```

```
    <driver>org.apache.derby</driver>
```

```
    <security>
```

```
      <user-name>user1</user-name>
```

```
      <password>pw</password>
```

```
    </security>
```

```
    <validation>
```

```
      <validate-on-match>>false</validate-on-match>
```

```
      <background-validation>>false</background-validation>
```

```
    </validation>
```

# standalone.xml

you must change the content  
of the datasources section like this

JNDI name

Symbolic  
name

Name or your  
DB

```
<statement>
```

```
  <share-prepared-statements>>false</share-prepared-statements>
```

```
</statement>
```

```
</datasource>
```

```
<drivers>
```

```
  <driver name="h2" module="com.h2database.h2">
```

```
    <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
```

```
  </driver>
```

```
  <driver name="org.apache.derby" module="org.apache.derby">
```

```
    <xa-datasource-class>org.apache.derby.jdbc.ClientXADataSource</xa-datasource-class>
```

```
  </driver>
```

```
</drivers>
```

```
</datasources>
```

- you must change the name of your DB
- you may change the symbolic name and the last token of the JNDI name

# Configure the datasource

- Make sure the driver libraries are included. In your Wildfly home directory, you should have the following directory structure (if not, create it):

modules->org->apache->derby-> main

- In main you should have:
  - derbyclient.jar
  - module.xml

## **Module.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.3" name="org.apache.derby">
  <resources>
    <resource-root path="derbyclient.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

# Check the datasource





- Make sure the DB Service is started (e.f. from within Netbeans)
- Open a shell, and cd to the bin directory of wildfly.
- Start wildfly with the following command:  
**`./standalone.sh &`**
- Check it in the console

# Connect to the console, check the driver

← → ↻ ⓘ localhost:9990/console/index.html#configuration;path=configuration~subsystems!css~datasou

**HAL Management Console**

Homepage Deployments **Configuration** Runtime Patching Access Control

Configuration	Subsystem (32)	Datasources & Drivers	JDBC Driver  
Subsystems >	<i>Filter by: name or subtitle</i>	Datasources >	<i>Filter by: driver name or provide</i>
Interfaces >	Batch	JDBC Drivers >	 h2
Socket Bindings >	JBeret		
Paths	Core Management		 org.apache.derby
System Properties	Datasources & Drivers >		
	Deployment Scanners		
	Discovery		
	Distributable Web		

# Connect to the console, check the datasource

← → ↻ ⓘ localhost:9990/console/index.html#configuration;path=configuration~subsystems!css~datasources!data-source-... ☆ S [copy] [gear] M

**HAL Management Console**

🔔 admin1

HomepageDeployments**Configuration**RuntimePatchingAccess Control

Configuration	Subsystem (32)	Datasources & Drivers	Datasource <div>⊕⌵ ↺</div>
Subsystems >	<i>Filter by: name or subtitle</i>	Datasources >	<i>Filter by: name, xa, .../disabled, c</i>
Interfaces >	Batch		✔ Derby... <div>View ⌵</div>
Socket Bindings >	JBeret	JDBC Drivers >	✔ ExampleDS
Paths	Core Management		
System Properties	Datasources & Drivers >		
	Deployment Scanners		
	Discovery		

## DerbyDS

Datasource

✔ The datasource **DerbyDS** is enabled. [Disable](#)

### Main Attributes

JNDI Name:	java:jboss/datasources/DerbyDS
Driver Name:	org.apache.derby
Connection URL:	jdbc:derby://localhost:1527/SESSIO...

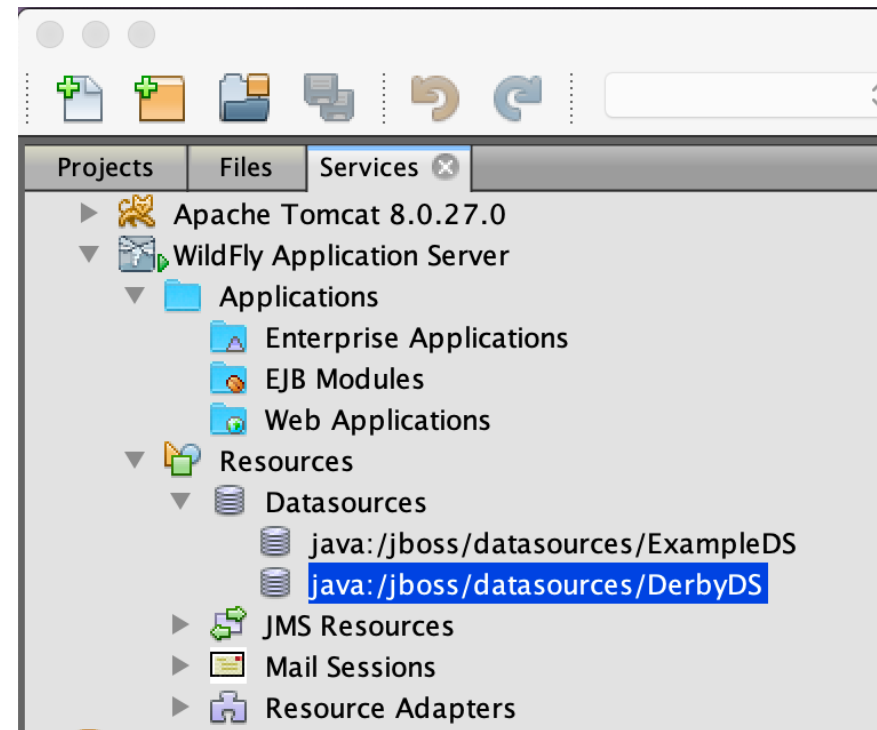
# Shut down Wildfly – repeat from within Netbeans

In the shell, issue the command:

```
./jboss-cli.sh --connect command=:shutdown
```

Repeat the check after starting wildfly from within Netbeans.

You should also see the datasource from within Netbeans.

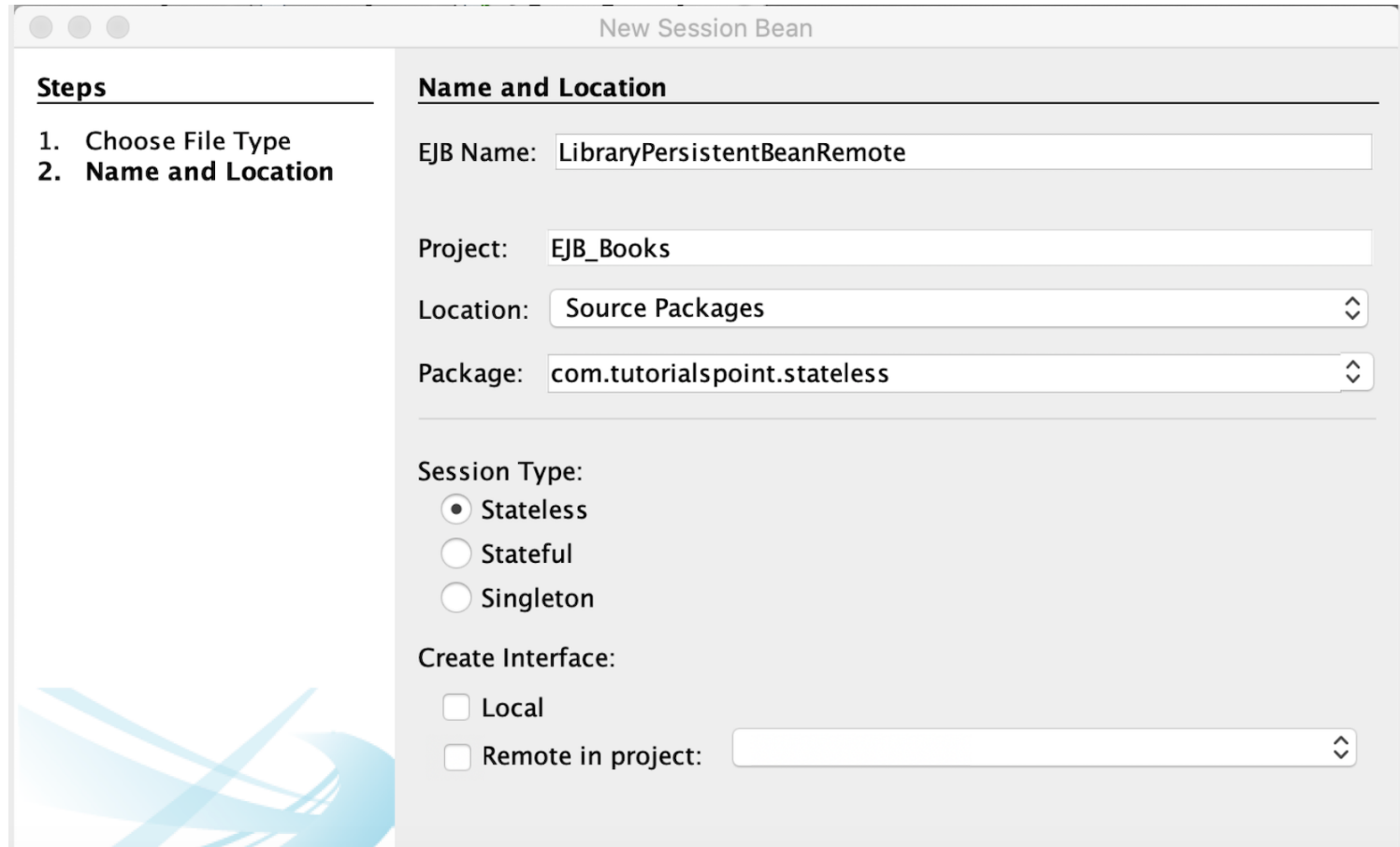




# let's create and configure a project

Let's create an EJB project as EJB Module.

In it let's create a stateless bean



**New Session Bean**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

EJB Name:

Project:

Location:

Package:

**Session Type:**

☒ Stateless

☐ Stateful

☐ Singleton

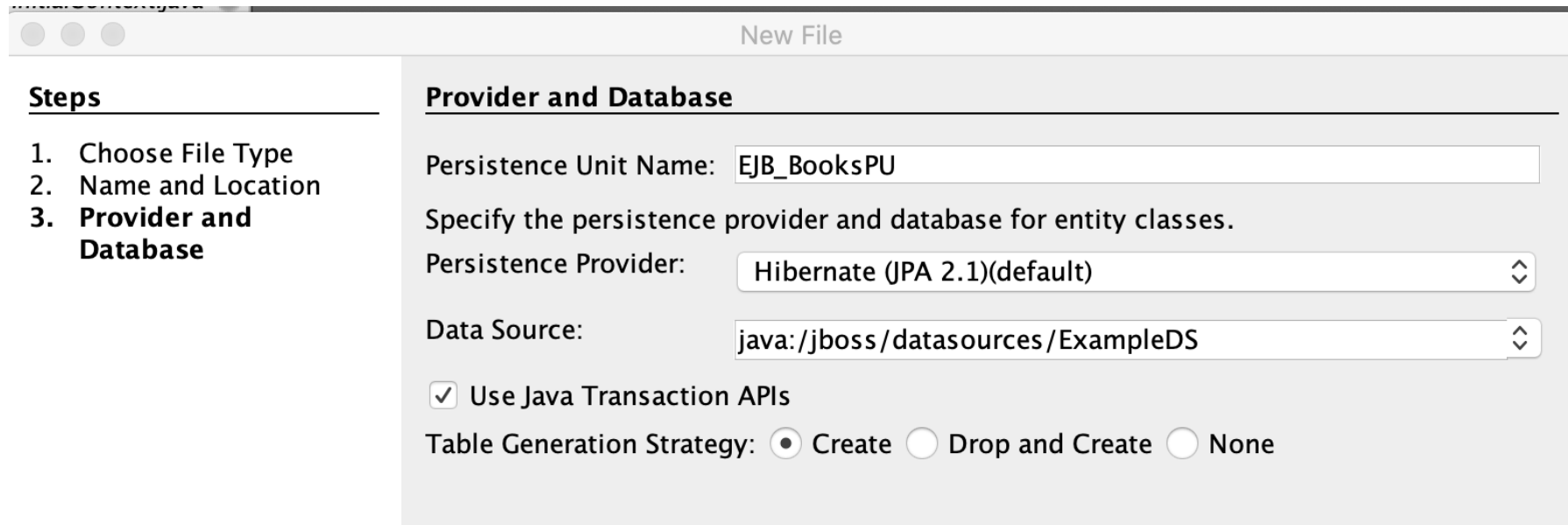
**Create Interface:**

☐ Local

☐ Remote in project:

# let's continue configuring our project

In it let's create an entity



The screenshot shows a 'New File' dialog box with a sidebar on the left and a main configuration area on the right.

**Steps**

1. Choose File Type
2. Name and Location
3. **Provider and Database**

**Provider and Database**

Persistence Unit Name:

Specify the persistence provider and database for entity classes.

Persistence Provider:

Data Source:

☒ Use Java Transaction APIs

Table Generation Strategy: ☒ Create ☐ Drop and Create ☐ None

# let's continue writing our project

Let's copy the code from

[https://www.tutorialspoint.com/ejb/ejb\\_persistence.htm](https://www.tutorialspoint.com/ejb/ejb_persistence.htm)

but for now NOT the client part.

```
@Remote
public interface LibraryPersistentBeanRemote {
    void addBook(Book bookName);
    List<Book> getBooks();
}
```

```
@Stateless
public class LibraryPersistentBean implements
LibraryPersistentBeanRemote {
    public LibraryPersistentBean() { }
    @PersistenceContext(unitName="EjbComponentPU")
    private EntityManager entityManager;
    public void addBook(Book book) { entityManager.persist(book); }
    public List<Book> getBooks() {
        return entityManager.createQuery("From Book").getResultList(); }
}
```

```
@Entity
@Table(name="books")
public class Book implements Serializable{
    private int id;
    private String name;
    public Book() { }
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    @Column(name="id")
    public int getId() { return id; }
    public void setId(int id) { this.id = id;}
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

# configuration files: persistence.xml

The name of the persistence-unit (in red) can be changed, but then also the name of the **@PersistenceContext** in the stateless must be coherent.

The first time we deploy, we uncomment the green lines. We keep them commented the following times.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns=http://xmlns.jcp.org/xml/ns/persistence
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="EjbComponentPU" transaction-type="JTA">
    <jta-data-source>java:/jboss/datasources/DerbyDS</jta-data-source>
    <exclude-unlisted-classes>>false</exclude-unlisted-classes>
    <!--properties>
      <property name="javax.persistence.schema-generation.database.action" value="create"/>
    </properties-->
  </persistence-unit>
</persistence>
```

# On the console, let's check the deployment

←

→

↻

localhost:9990/console/index.html#deployment;deployment=EJB\_Books.jar

HAL Management Console

« Back / Deployment ⇒ EJB\_Books.jar ▾

Management Model

↻

▾

EJB\_Books.jar

subdeployment

subsystem

ejb3

jpa

logging

microprofile-opentracing-smallrye

batch-jberet

datasources

jaxrs

resource-adapters

undertow

webservices

A deployment represents anything that can be deployed (e.g. an application standard archive such as RAR or JBoss-specific deployment) into a server.

Data

Attributes

Operations

Edit

Reset

Help

Disabled Time	MILLISECONDS
Disabled Timestamp	
Enabled	true
Enabled Time	1607246059520 MILLISECONDS
Enabled Timestamp	2020-12-06 10:14:19,520 CET
Managed	false

# Check the log, and note the JNDI bindings

The screenshot shows the NetBeans IDE 8.2 interface. The top toolbar includes icons for file operations and running the application. The left sidebar displays the project structure for 'EJB\_Books\_Client', including 'Generated Sources', 'Libraries', 'Test Libraries', 'Enterprise Beans', 'Configuration Files', and 'Server Resources'. The 'persistence.xml' file is selected in the 'Server Resources' folder. The main editor window shows the 'persistence.xml' file with the following content:

```
1
2 <persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
3 <A">
```

The bottom pane shows the 'Output' window with the 'WildFly Application Server' log. The log contains several informational messages, including:

```
9,704 INFO [org.jboss.as.jpa] (MSC service thread 1-1) WFLYJPA0002: Read persistence.xml for EjbC
9,822 INFO [org.jboss.weld.deployer] (MSC service thread 1-7) WFLYWELD0003: Processing weld deploy
9,895 INFO [org.hibernate.validator.internal.util.Version] (MSC service thread 1-7) HV000001: Hibe
9,990 INFO [org.jboss.as.ejb3.deployment] (MSC service thread 1-7) WFLYEJB0473: JNDI bindings for

java:global/EJB_Books/LibraryPersistentBean!com.tutorialspoint.stateless.LibraryPersistentBeanRemo
java:app/EJB_Books/LibraryPersistentBean!com.tutorialspoint.stateless.LibraryPersistentBeanRemote
java:module/LibraryPersistentBean!com.tutorialspoint.stateless.LibraryPersistentBeanRemote
java:jboss/exported/EJB_Books/LibraryPersistentBean!com.tutorialspoint.stateless.LibraryPersistent
ejb:/EJB_Books/LibraryPersistentBean!com.tutorialspoint.stateless.LibraryPersistentBeanRemote
java:global/EJB_Books/LibraryPersistentBean
java:app/EJB_Books/LibraryPersistentBean
java:module/LibraryPersistentBean

0,068 INFO [org.infinispan.CONTAINER] (ServerService Thread Pool -- 2) ISPN000128: Infinispan ver:
0,095 INFO [org.jboss.as.jpa] (ServerService Thread Pool -- 2) WFLYJPA0010: Starting Persistence I
0,100 INFO [org.infinispan.CONFIG] (MSC service thread 1-1) ISPN000152: Passivation configured wit
0,102 INFO [org.infinispan.CONFIG] (MSC service thread 1-1) ISPN000152: Passivation configured wit
0,111 INFO [org.hibernate.jpa.internal.util.LogHelper] (ServerService Thread Pool -- 2) HHH000204:
name: EjbComponentPU
```

The one we need is: **ejb:/EJB\_Books/LibraryPersistentBean!com.tutorialspoint.stateless.LibraryPersistentBeanRemote**

# Let's create the client project

We need to:

- make sure libraries are ok (same as those we used for the stateless project)

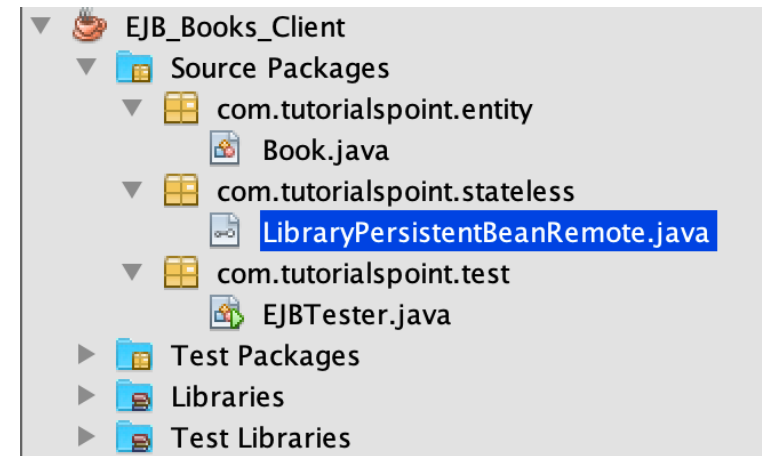
- copy in it the Remote interface and the Entity.

NOTE: the Entity MAY keep the annotations, but they are unused because here it will be detached. So if we want we can as well remove them.

- copy the client code (class EJBTester) from

[https://www.tutorialspoint.com/ejb/ejb\\_persistence.htm](https://www.tutorialspoint.com/ejb/ejb_persistence.htm)

and modify it according to the following slide:



# Let's put the right JNDI bindings:

```
final Hashtable jndiProperties = new Hashtable();
jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY,
    "org.wildfly.naming.client.WildFlyInitialContextFactory");
jndiProperties.put(
    Context.PROVIDER_URL, "http-remoting://localhost:8080");
try {
    ctx = new InitialContext(jndiProperties);
} catch (NamingException ex) {
    ex.printStackTrace();
}
```

```
LibraryPersistentBeanRemote libraryBean= (LibraryPersistentBeanRemote) ctx.lookup(
    "ejb:/EJB_Books/LibraryPersistentBean!com.tutorialspoint.stateless.
    LibraryPersistentBeanRemote");
```



# Now we can run the client project

Of course, the server part must be deployed first.

