A Jboss Runnable ear, how to make it run on Tomcat 9.x server

**ChatGPT said:**

Tomcat 9 is a **Servlet container**, not a full **Java EE (Jakarta EE)** application server like JBoss (now WildFly). So, it **doesn't support EAR files**, which are used to package enterprise applications with EJBs, WARs, and other components.

However, you can **migrate or adapt your EAR-based application** to run on Tomcat by doing the following:

**🔧 1. Understand What's in Your EAR**

An EAR file typically contains:

* One or more WAR files (web apps)
* One or more JAR files (business logic, usually EJBs)
* application.xml (deployment descriptor)

You can unzip your EAR and inspect the contents.

**🔁 2. Convert EAR to WAR (If Possible)**

Tomcat only supports WAR deployments, so:

* If your EAR just wraps a single WAR file:
  + ✅ You can extract and deploy the WAR to Tomcat’s webapps/ directory.
* If your EAR contains:
  + EJB modules
  + Messaging (JMS)
  + JPA (with container-managed transactions)

Then you need to **refactor** or **replace** those components to run in a Tomcat-compatible way.

**⚙️ 3. Replace JBoss-Specific Features**

JBoss supports:

* EJBs
* JPA with container-managed transactions
* JMS
* JNDI

Tomcat **doesn't support these natively**, so you’ll need to:

**✔ Replace or Embed:**

| **JBoss Feature** | **Tomcat Alternative** |
| --- | --- |
| **EJBs** | Replace with POJOs or use **Spring Framework** (Spring beans) |
| **JMS** | Use **ActiveMQ / RabbitMQ** client libraries |
| **JPA** | Use **Hibernate** with manually managed transactions |
| **JNDI** | Configure via context.xml or use Spring for dependency injection |

**🛠 4. Refactor & Package as a WAR**

Steps:

1. Extract the WAR from the EAR (or refactor the modules into one WAR).
2. Ensure that all dependencies (JARs) are included in WEB-INF/lib/.
3. Configure:
   * web.xml
   * context.xml (for DB resources, JNDI, etc.)
4. Remove EJB usage, or refactor them into Spring-managed beans.

**🚀 5. Deploy to Tomcat**

* Place the .war in TOMCAT\_HOME/webapps/
* Start Tomcat: bin/startup.sh or bin/startup.bat

**🧪 Optional: Use a Lightweight EE Container in Tomcat**

If refactoring is too complex, you can:

* Embed **Spring Boot** (ideal for microservices)
* Use **TomEE** (Tomcat + Java EE support)
* Use **Payara Micro** or **Open Liberty** for lighter EE features

**✅ Summary**

| **Task** | **Required?** |
| --- | --- |
| Extract WAR from EAR | ✅ |
| Remove/replace EJBs | ✅ |
| Configure JNDI, DB, JMS manually | ✅ |
| Package everything into a WAR | ✅ |
| Use Spring / TomEE if needed | Optional |

If you share what's inside your EAR (structure, technologies used), I can give you more tailored steps to convert it.

**You said:**

The same application able to run on Eclipse integrated Tomcat with simulated response , with out packaging . what can be done to make ear deployed similar way to validate server startup..

**ChatGPT said:**

Ah, got it. So your application is running **inside Eclipse** using **Tomcat with simulated responses**, probably through a **dynamic web project**, maybe using **mocked services** or **Spring beans**, and you want to replicate that same lightweight startup behavior for a full EAR-based app — **without needing full packaging/deployment** — just enough to validate startup and basic flow.

Here’s a breakdown of what’s happening and how to make it run similarly outside Eclipse, even for EAR apps:

**✅ Why It Works in Eclipse**

* Eclipse is smart about unpacking EAR/WARs.
* It lets you run modules directly from source.
* You might have:
  + Web modules as Dynamic Web Projects
  + Backend logic as Java Projects
* Simulated responses or mocks mean you're not using full EJBs or messaging.

**✅ Goal: Simulate EAR Deployment Outside of Eclipse Without JBoss**

Since EAR deployment **requires a full Java EE container** (like JBoss/WildFly), but your app **runs fine in Eclipse + Tomcat** using just web modules and mocks, you can:

**🔁 Option 1: Flatten EAR into a Dynamic Web Project**

Instead of deploying an EAR, **merge** the components into a single WAR-compatible project.

**Steps:**

1. **Unzip the EAR**, extract WAR and JARs.
2. Create a **Dynamic Web Project** or Maven WAR project.
3. Place:
   * WEB-INF/classes → compiled classes.
   * WEB-INF/lib → all EAR’s JAR dependencies.
4. Add your web.xml and any mock logic.

**🧪 Option 2: Simulate EAR Deployment Using Maven Multi-Module Project**

If you want to **preserve the modularity** of the EAR (EJBs, WARs), but run on Tomcat:

**Structure:**

cpp

CopyEdit

parent-ear/

├── web-module/ (WAR)

├── service-module/ (JAR logic, no EJB)

├── ear/ (EAR packaging module, not needed for Tomcat)

Run the web-module directly on Tomcat.

* Use **Spring Boot** or plain **Servlet-based** architecture.
* Inject services from the JAR (service-module).

**🛠 Option 3: Use cargo-maven2-plugin or Tomcat Maven Plugin to Simulate Deployment**

You can run Tomcat locally like Eclipse does:

xml

CopyEdit

<plugin>

<groupId>org.codehaus.cargo</groupId>

<artifactId>cargo-maven2-plugin</artifactId>

<version>1.10.6</version>

<configuration>

<container>

<containerId>tomcat9x</containerId>

<type>installed</type>

<home>/path/to/tomcat9</home>

</container>

<configuration>

<type>standalone</type>

<home>${project.build.directory}/tomcat-home</home>

</configuration>

<deployables>

<deployable>

<groupId>com.example</groupId>

<artifactId>web-module</artifactId>

<type>war</type>

</deployable>

</deployables>

</configuration>

</plugin>

Or even simpler using:

bash

CopyEdit

mvn tomcat7:run

(Many apps still use tomcat7 plugin for 8.x/9.x too.)

**🔄 Option 4: Use Embedded Tomcat (Programmatic Startup)**

Create a simple Java class that bootstraps Tomcat, loads your web app from the file system or a WAR:

java

CopyEdit

Tomcat tomcat = new Tomcat();

tomcat.setPort(8080);

tomcat.getHost().setAppBase(".");

tomcat.addWebapp("", new File("src/main/webapp").getAbsolutePath());

tomcat.start();

tomcat.getServer().await();

Put this in your main() method. It starts your app like Eclipse does, from code.

**✅ Best Practice Summary**

| **Goal** | **Recommendation** |
| --- | --- |
| Simulate EAR without JBoss | Flatten to WAR, run in Tomcat |
| Preserve module structure | Use Maven multi-module and run WAR module |
| Start like Eclipse | Use embedded Tomcat in main() method |
| Validate EAR-style startup | Use mocks/spring + Tomcat deployment |

If you can share the EAR structure (what's inside — WARs, JARs, any application.xml, etc.), I can give you a working example to convert or simulate the EAR deployment behavior. Would that help?

4o