Understanding Stack Trace in JBoss

What is a Stack Trace?

A stack trace is a list of method calls that shows where exactly an error happened in your code or application.

Think of it like a stack of boxes = method calls.

One box (method) causes an error.

The stack trace tells you which box failed and how the boxes are stacked on top of each other.

When do you see a stack trace?

You see a stack trace in the JBoss logs when an exception or error happens, such as:

- Deployment failure
- NullPointerException
- DB connection error
- File not found

Stack Trace Example:

Suppose your app throws a NullPointerException. You'll see this in server.log:

```
java.lang.NullPointerException: Cannot invoke "String.length()" because "name" is null
at com.myapp.CustomerBean.processName(CustomerBean.java:45)
at com.myapp.CustomerService.getCustomer(CustomerService.java:23)
at com.myapp.MainServlet.doGet(MainServlet.java:18)
```

How to read a stack trace?

Start from top to bottom:

1. Error type: NullPointerException - the exception.

2. Message: "name" is null - what caused it.

3. Location in code:

- Line 45 of CustomerBean.java
- Called by CustomerService.java at line 23
- Called by MainServlet.java at line 18

Common Stack Trace Types

- NullPointerException: A variable was null
- ClassNotFoundException: Missing .jar or class
- OutOfMemoryError: JVM ran out of memory
- SQLException: Problem connecting to DB
- BindException: Port already in use

How it helps in troubleshooting?

A stack trace tells you exactly:

- What failed
- Where in the code it failed
- Which line number
- Which method was involved

Summary:

Stack Trace = A list of method calls showing where the error occurred.

Found in = JBoss server.log

Helps to = Debug errors quickly

Always look for = The first 'Caused by:' line - it's the root cause.