Alright V 👍

Here is your **FINAL REVISITABLE DOCUMENT** — clean, simple, and **INCLUDING Nexus Artifact Uploader** + all fixes we discovered.

I wrote this exactly like we built it together so next time you **won't troubleshoot again**.

☑ PARAMETERIZED JOB — FULL WORKING STEPS (FINAL GUIDE)

Project:

https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git

Runs inside:

```
Jenkins Docker Container
Workspace → /var/jenkins_home/workspace/Parameters
```

🟢 STEP 1 — Create Job

```
New Item → Freestyle Project
Name: Parameters
```

🟢 STEP 2 — Add Parameters

Tick:

```
This project is parameterized
```

Add:

◇ Branch Name

```
Name: BRANCH
Type: String Parameter
Default: master
```

◇ Environment

```
Name: ENVIRONMENT
Type: Choice Parameter
Values:
dev
qa
prod
```

◇ App Context

```
Name: APP_CONTEXT
Type: String Parameter
Default: HelloWorld
```

◇ Maven Version (VERY IMPORTANT FIX)

```
Name: VERSION
Type: String Parameter
Default: 3.0
```

👉 Fixes `${version}` recursion error from pom.xml.

🟢 STEP 3 — Add SonarQube Build Step (BEFORE Maven)

Click:

```
Add build step → Execute shell
```

Paste:

```
echo "Running SonarQube Analysis..."

mvn sonar:sonar \
-Dsonar.projectKey=ncodeit-hello-world \
-Dsonar.projectName=ncodeit-hello-world \
-Dsonar.host.url=http://SONAR_EC2_IP:9000 \
-Dsonar.login=SONAR_TOKEN (it is the token from sonarqube, paste it here rather than credentials)
```

🟢 STEP 4 — Git Clone

Source Code Management → Git

```
Repository:
https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git

Branch:
${BRANCH}
```

🟢 STEP 5 — Maven Build (JAVA 8 FIX)

Because project uses Java 6 syntax → we FORCE Java 8.

Add Build Step:

```
Execute Shell
```

Paste:

```
echo "Running Maven build with Java 8 compatibility..."

mvn clean package \
-Dversion=${VERSION} \
-Dmaven.compiler.source=8 \
-Dmaven.compiler.target=8
```

✅ Fixes:

```
Source option 6 is no longer supported
```

🟢 STEP 6 — Verify WAR File (PROOF STEP)

Add another:

```
Execute Shell
echo "Workspace path:"
pwd

echo "Listing project files:"
ls -l

echo "Checking generated WAR file..."
ls -l target/
```

You should see:

```
ncodeit-hello-world-3.0.war
```

STEP 7 — Nexus Artifact Uploader (NEW ADDITION)

Add Build Step:

```
Nexus Artifact Uploader
```

Fill EXACTLY:

```
Nexus Version : Nexus3
Protocol      : http
Nexus URL     : NEXUS_EC2_IP:8081
Repository    : Project-02
Credentials   : Nexus
```

Artifact Details:

```
GroupId    : com.ncodeit
Version    : ${VERSION}
ArtifactId : ncodeit-hello-world
Type       : war
Classifier :  ← LEAVE EMPTY
File       : target/ncodeit-hello-world-${VERSION}.war
```

What this does

Uploads WAR into Nexus after build.

STEP 8 — Deploy To Tomcat (Docker)

Add Build Step:

```
Execute Shell
```

Paste:

```
echo "Deploying WAR to Tomcat using APP_CONTEXT..."

WAR_NAME=$(ls target/*.war | head -n1)

echo "WAR detected: $WAR_NAME"

docker cp $WAR_NAME tomcat-container:/usr/local/tomcat/webapps/${APP_CONTEXT}.war

echo "Deployment triggered. Waiting for Tomcat..."
sleep 10

echo "App should be available at:"
echo http://34.207.207.112:8082/${APP_CONTEXT}/
```

STEP 9 — What URL To Open

Because Spring3 MVC uses DispatcherServlet:

http://IP:8082/${APP_CONTEXT}/

NOT:

```
/index.jsp
```

STEP 10 — Why You Were Seeing 404 (ROOT CAUSE)

Your WAR deployed successfully.

But:

```
Spring DispatcherServlet mapped to "/"
```

So Tomcat has:

```
META-INF
WEB-INF
resources
```

No public JSP at root.

👉 Spring controller must handle request.

This is **NORMAL**, not deployment failure.

🟢 FINAL FLOW (WITH NEXUS)

```
Git Clone
   ↓
Maven Build (Java8 override)
   ↓
WAR Created
   ↓
Nexus Artifact Upload
   ↓
Docker cp to Tomcat
   ↓
Spring MVC handles routing
```

🧠 SHORT EXPLANATION OF IMPORTANT FIXES

✓ Java Version Error

Project uses Java6 → modern JVM rejects it → we forced:

```
-Dmaven.compiler.source=8
-Dmaven.compiler.target=8
```

✓ VERSION recursion error

pom.xml uses `${version}` → Jenkins must pass VERSION param.

✓ 404 after deploy

Not missing WAR.

Spring MVC uses controller routing → not static JSP.

🟢 FINAL WORKING BUILD ORDER (Copy This Mental Model)

```
Parameters
Git Clone
Execute Shell → mvn clean package
Execute Shell → verify WAR
Nexus Artifact Uploader
Execute Shell → docker cp deploy
```

🎯 Interview One-liner

“Parameterized Jenkins jobs externalize values like branch, version, and context path; we used Maven overrides for Java compatibility, uploaded artifacts to Nexus for storage, and deployed WARs to Tomcat via docker cp while Spring MVC handled routing.”