TH BINGEN
University of Applied Sciences

Running LibreSocial: A Legacy Java Project Setup Guide

Master's Thesis

of

Faisal Imam
Matriculation No.: 2195258

Technische Hochschule Bingen
Fachbereich 2

Reviewer: Prof. Dr.-Ing. Kalman Graffi
May 5, 2025

Contents

1 Introduction **1**
1.1 Introduction 1

2 Environment Setup **1**
2.1 System Requirements 1
2.1.1 Operating System 1
2.2 Tools and Dependencies 2
2.2.1 LibreSocial Source Code 2
2.2.2 Version Control (Git Support) 2
2.2.3 Java Development Kit (JDK) 3
2.2.4 Integrated Development Environment (IDE) 3
2.2.5 External Dependencies 3
2.3 Network Requirements 4

3 Running LibreSocial **4**
3.1 Overview 4
3.2 Step 1: Cloning the Repository 5
3.3 Step 2: Installing Java JDK 12
3.4 Step 3: Importing the Project into Eclipse Indigo 13
3.4.1 Fixing Build Path or JRE Issues 13
3.5 Step 4: Applying Unlimited JCE Policy Files 14
3.6 Running the Project 14
3.7 Conclusion 14

4 Troubleshooting Tips **14**
4.1 JDK Version Mismatch 14
4.2 Eclipse Project Not Recognized 14
4.3 Unresolved Libraries or Missing JARs 15
4.4 GUI Not Launching or Runtime Crash 15

5 Conclusion **16**

Appendix **17**
A Useful Commands and Configuration Paths 17

Abstract

LibreSocial is a fully decentralized peer-to-peer (P2P) framework designed to enable secure, scalable, and privacy-preserving online social networking. This documentation provides a practical guide for running and maintaining the LibreSocial project, which integrates structured P2P overlays, modular plugins, and a zero-trust architecture. It addresses the challenges of dependency resolution, outdated Java APIs, and the migration of legacy code. Key components such as identity management, access-controlled data storage, secure communication channels, and distributed data structures (e.g., DHTs, social caches) are emphasized. The documentation also outlines the setup of the development environment, compilation process, and execution flow, while incorporating insights into replication strategies, metadata-based search, and capacity management protocols. Targeted at developers and researchers, this document supports the sustainable deployment and extension of LibreSocial in real-world distributed environments.

1 Introduction

1.1 Introduction

LibreSocial is a fully decentralized peer-to-peer (P2P) framework that enables secure, privacy-preserving, and scalable online social networking. Originally developed as an academic project and evolved from its predecessor. LibreSocial presents an alternative to conventional centralized social networks by leveraging structured overlay networks and a zero-trust architecture. It incorporates key distributed system components such as identity management, encrypted communication, and modular plugin integration to build a robust social platform free from central control or monetization of user data.

This documentation aims to guide users through the process of setting up and running the LibreSocial project, which was originally developed using legacy Java technologies. The project relies on Java 1.6 (compiler compliance level) and JavaSE-1.7 runtime environment, making it incompatible with modern IDEs (eg. Eclipse 2023 or JetBrains IntelliJ IDEA). Due to these constraints, the project requires the use of an older IDE such as Eclipse Indigo for proper compilation and execution.

The primary challenge addressed in this documentation is running the project in its current state without upgrading or rewriting its core components. By following the steps and solutions outlined, users will be able to replicate a working environment and successfully run the LibreSocial system despite its outdated dependencies.

2 Environment Setup

To successfully run the LibreSocial project in its current legacy form, the following system requirements and software tools must be properly configured. This chapter is divided into system-level setup and software dependencies.

2.1 System Requirements

To successfully run the LibreSocial project in its current legacy form, the following system requirements and configurations must be met:

2.1.1 Operating System

The project has been successfully tested on:

- Windows 10/11
- Ubuntu 20.04 LTS and above
- macOS (with Java 7 compatibility layer)

2.2 Tools and Dependencies

This section outlines the tools used in more detail, including external libraries that must be manually configured.

Component	Purpose
IntelliJ IDEA (for Git)	For cloning the repository and managing version control.
LibreSocial Source Code	The main codebase, including P2P logic and GUI.
Java JDK 1.7.0_80	Required to compile the project; must be installed first.
Eclipse Indigo (RCP/RAP)	Used to build and run the project in a compatible IDE.
Unlimited JCE Policy Files	Enables full cryptographic support in Java.

Table 1: Summary of Tools and Dependencies (in recommended order)

2.2.1 LibreSocial Source Code

The source code for the LibreSocial project is hosted on GitLab and can be accessed at the following URL:

- **Repository URL:** `https://gitlab.rlp.net/k.graffi/libresocial`

The repository contains all essential modules of the project, including the core peer-to-peer framework, social network features, plugin interfaces, and graphical user interface components. The latest and most stable version of the code is maintained in the "master" branch.

2.2.2 Version Control (Git Support)

Due to the legacy nature of Eclipse Indigo, integrated version control features such as Git are either unavailable or extremely limited. Therefore, it is recommended to manage version control tasks such as cloning, committing, pushing, and branching using a modern IDE or Git client.

Recommended: IntelliJ IDEA

A highly recommended tool for version control is **IntelliJ IDEA**, which offers robust and user-friendly Git integration.

- **Download Link:** `https://www.jetbrains.com/idea/download/?section=windows`
- **Student/Teacher License:** JetBrains provides free licenses for students and educators. You can register using your academic email address and apply for a free activation code at: `https://www.jetbrains.com/community/education/`

You can use IntelliJ IDEA solely for version control without modifying or opening the Eclipse Indigo project files directly.

Other Version Control Options

If you prefer different tools, here are additional alternatives that can be used safely alongside Eclipse Indigo:

- **Git (Command Line):** Lightweight and flexible option for full control.
- **GitHub Desktop:** Beginner-friendly graphical interface for GitHub repositories.
- **Sourcetree:** Advanced Git GUI client with visual commit history and branch management.
- **Visual Studio Code:** A lightweight editor with built-in Git support and extensions.

Note: Regardless of the tool, ensure that version control actions (e.g., staging, commits, merges) are done outside of Eclipse Indigo to avoid compatibility issues.

2.2.3 Java Development Kit (JDK)

LibreSocial is built using legacy Java technologies and requires:

- **JDK Version:** 1.7.0_80
- **Download Link:** `https://www.oracle.com/java/technologies/javase/javase7-archive-downloads.html`

Ensure the following:

- `JAVA_HOME` is set to the JDK installation path.
- The JDK's `bin` directory is added to the system `PATH`.

2.2.4 Integrated Development Environment (IDE)

LibreSocial must be compiled using:

- **IDE:** Eclipse for RCP and RAP Developers
- **Version:** Indigo Release (Build ID: 20110615-0604)
- **Download Link:** `https://www.eclipse.org/downloads/packages/release/indigo/sr2/eclipse-rcp-and-rap-developers`

2.2.5 External Dependencies

The project requires the **Unlimited Strength Jurisdiction Policy Files** to enable full cryptographic functionality.

- **Download Link:** `https://www.oracle.com/java/technologies/javase-jce-all-downloads.html`

- After downloading:
  1. Extract the downloaded archive.
  2. Locate `local_policy.jar` and `US_export_policy.jar`.
  3. Backup the original files in your Java installation directory.
  4. Replace them with the new versions by navigating to the following path, depending on your operating system:
     - **Windows:** `C:\Program Files\Java\jdk1.7.0_80\lib\security`
     - **Ubuntu/Linux:** `/usr/lib/jvm/java-7-oracle/jre/lib/security` (or wherever your JDK is installed under `/usr/lib/jvm/`)
     - **macOS:** `/Library/Java/JavaVirtualMachines/jdk1.7.0_80.jdk/Contents/Home/`

2.3 Network Requirements

Internet access is only required for downloading the necessary tools and dependencies. No internet access is needed during the build or execution phases.

3 Running LibreSocial

This chapter provides a complete step-by-step guide to running the LibreSocial project from start to finish using a legacy Java environment. Each step includes practical instructions, tool usage, and screenshots where applicable.

3.1 Overview

Before running the project, ensure that all required tools are installed as described in Chapter 2. This chapter walks through the following execution flow:

1. Install IntelliJ IDEA for version control.
2. Clone the LibreSocial project from GitLab.
3. Install and configure Java Development Kit (JDK 1.7).
4. Install Eclipse Indigo and import the project.
5. Apply the Unlimited Strength JCE Policy files.

3.2 Step 1: Cloning the Repository

The LibreSocial codebase is hosted on a private GitLab instance. To interact with it programmatically and securely via IntelliJ IDEA, a Personal Access Token (PAT) is required. This section outlines the necessary configuration in IntelliJ and the secure cloning of the repository.

3.2.1 Forking the LibreSocial Repository

To maintain a clean workflow and enable independent development, it is recommended to fork the original repository before cloning:

1. Open the main LibreSocial repository: `https://gitlab.rlp.net/k.graffi/libresocial`
2. In the top-right corner, click `Fork`.
3. Select the target namespace (your GitLab profile).
4. Ensure that the branch `2025_LibreSocial_Docker` is present in your forked repository. This is the latest and active development branch.

Note: Forking ensures that your changes remain isolated and do not affect the upstream repository. It is also a prerequisite for submitting merge requests later.

3.2.2 GitLab Integration in IntelliJ IDEA

IntelliJ supports GitLab through built-in version control integration. Before cloning, it is recommended to authenticate via a GitLab PAT for secure and persistent access.

1. Launch IntelliJ and open the settings panel via `File > Settings`.

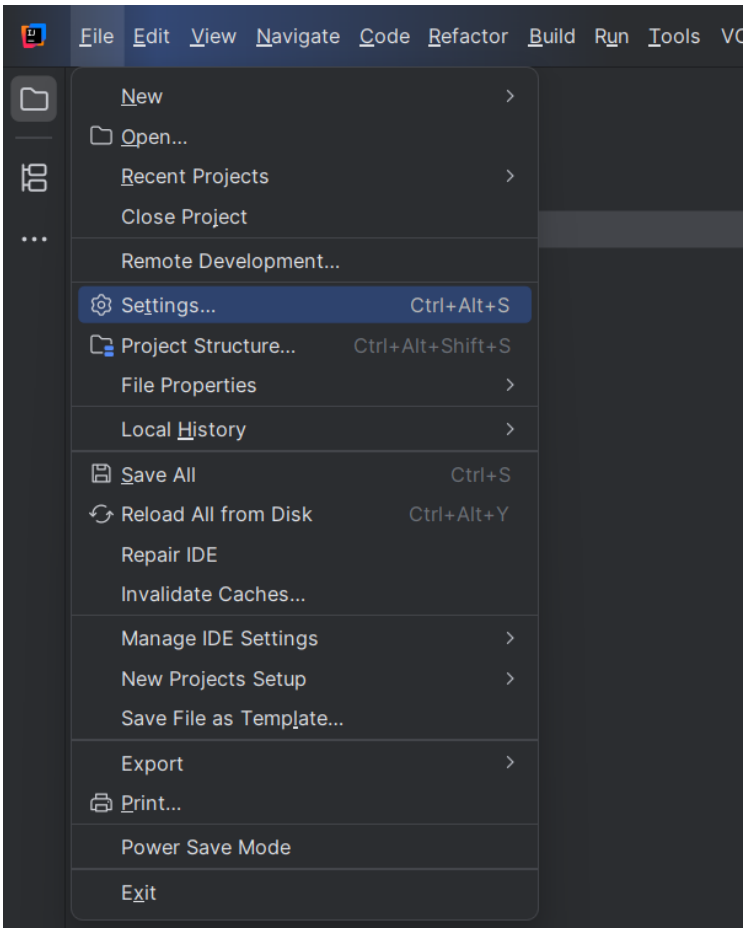


Figure 1: Accessing Settings in IntelliJ IDEA

2. Navigate to `Version Control > GitLab` to manage GitLab accounts.

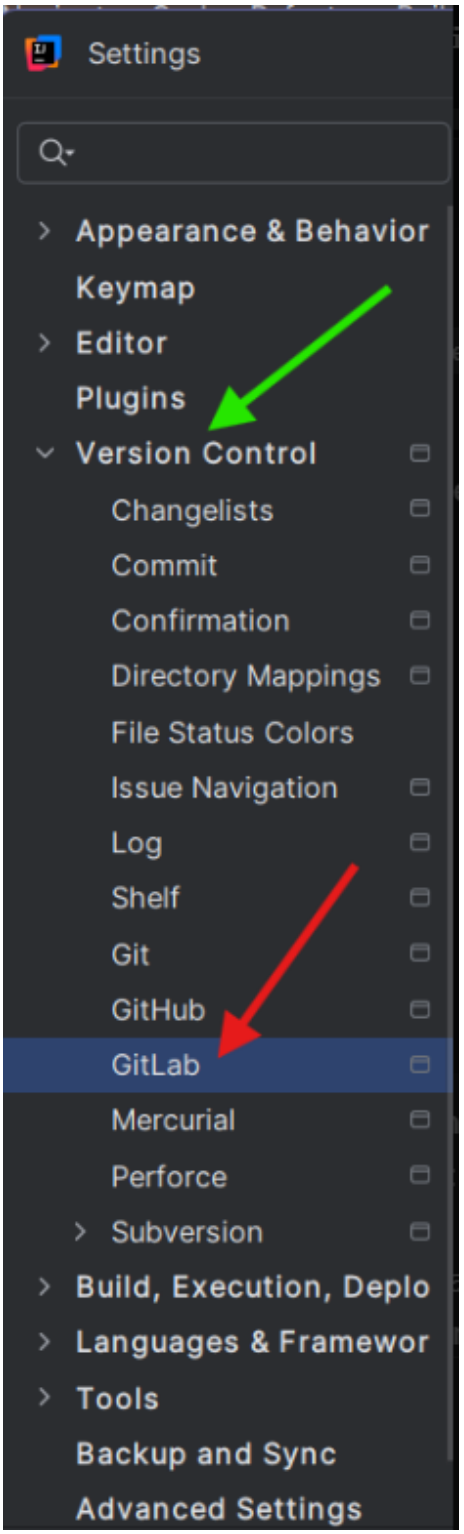


Figure 2: Locating GitLab under Version Control settings

3. Click the + icon to register a new GitLab account and set the GitLab server URL to `https://gitlab.rlp.net`.

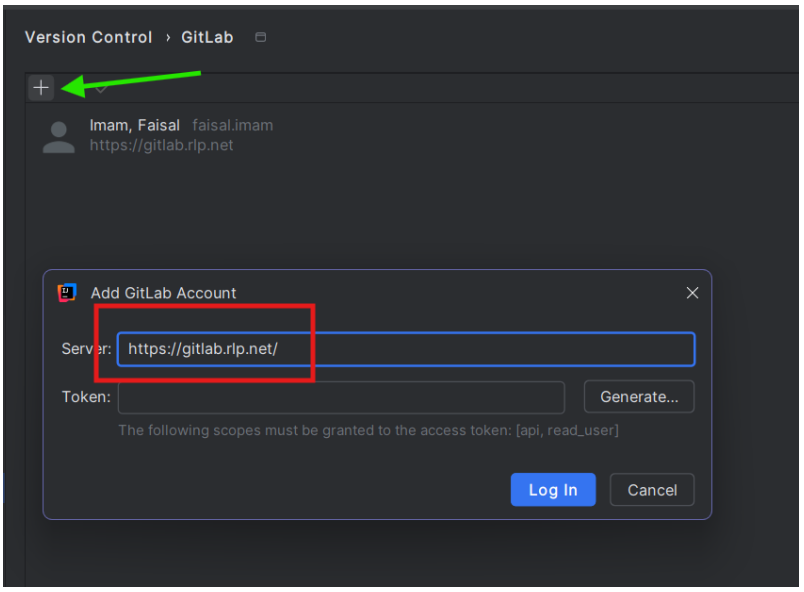


Figure 3: Adding a GitLab server to IntelliJ

3.2.3 Creating a Personal Access Token (PAT)

For authentication, a personal token must be generated on GitLab:

1. In GitLab, click your profile icon and choose `Preferences`.

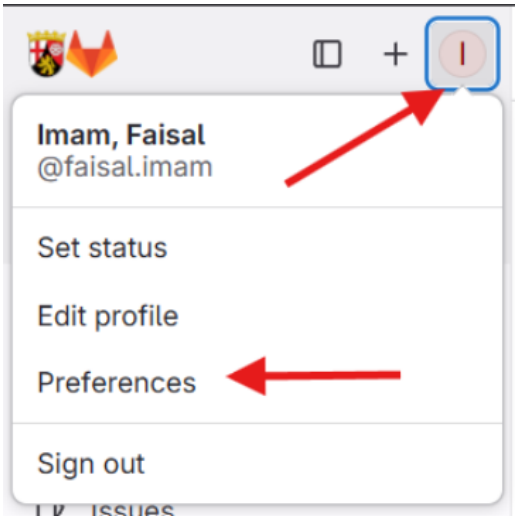


Figure 4: Accessing GitLab Preferences

2. Navigate to `Access Tokens`. Provide a meaningful name, set an expiration date, and ensure the following scopes are enabled: `api`, `read_api`, and `read_user`.

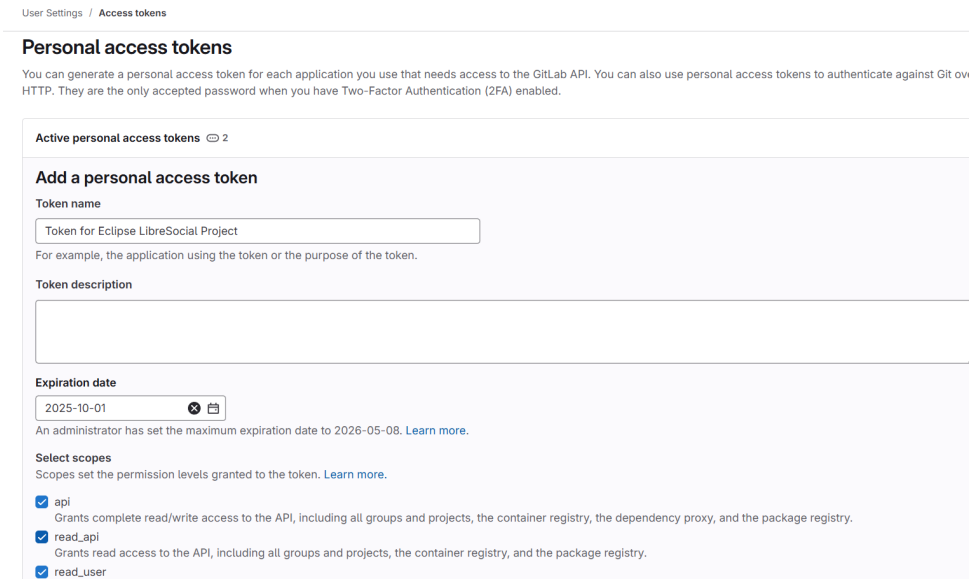


Figure 5: Creating a Personal Access Token in GitLab

3. After generating the token, **save it securely**. It will not be displayed again.

Paste this token into IntelliJ's GitLab login window to complete the authentication process.

3.2.4 Cloning Your Forked Repository

Once authenticated, the next step is to clone your fork of the LibreSocial repository using IntelliJ IDEA.

1. Go to `File > New > Project from Version Control`. This option is used when starting a new project directly from a remote Git repository.

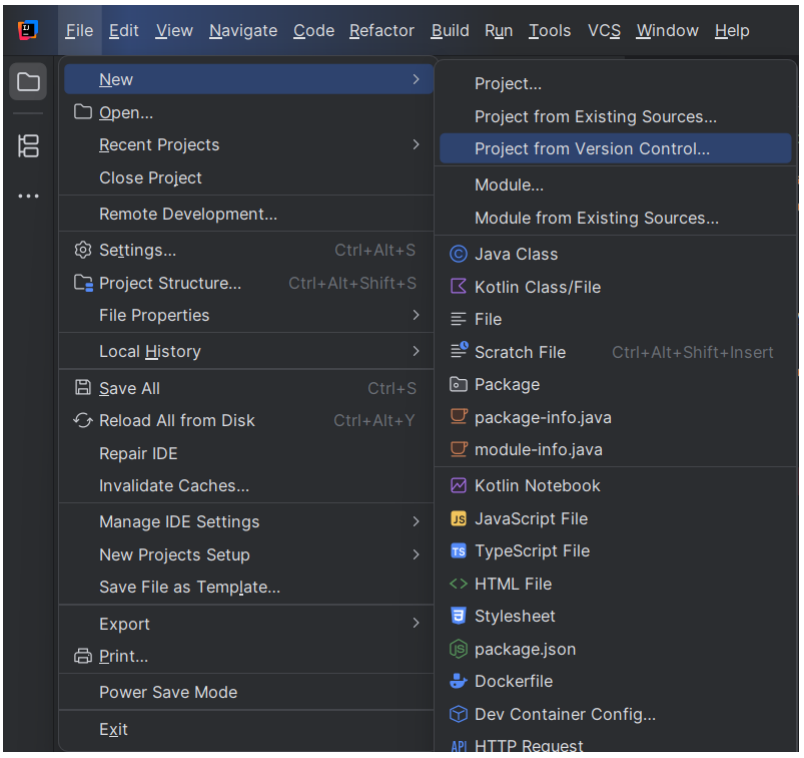


Figure 6: Initiating a new project from Version Control

2. In the dialog that appears, select the `GitLab` tab from the left sidebar. IntelliJ will display the list of repositories associated with your GitLab account.
3. Locate and select your forked repository, typically named `<yourusername>/libresocial`.
4. Choose the local path where you want the project to be cloned (e.g., `C:\Users\yourname\Projects\`
5. Click the `Clone` button to download the full repository. IntelliJ will fetch all branches, commit history, and metadata.

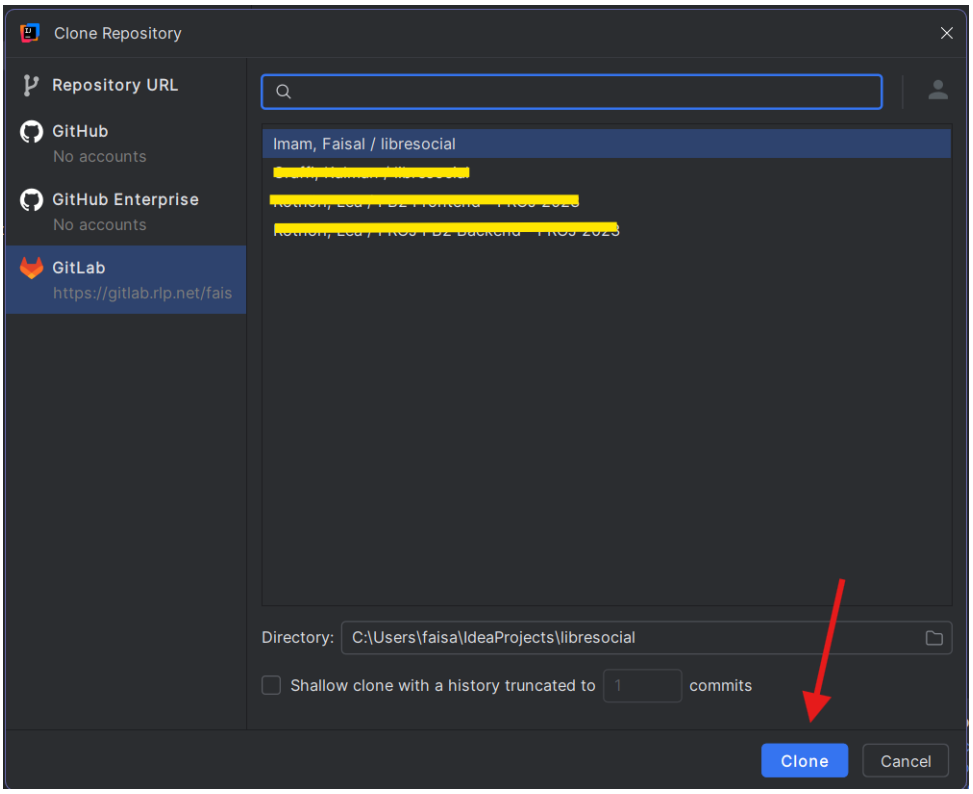


Figure 7: Selecting and cloning the LibreSocial repository

3.2.5 Verifying the Active Branch

Once cloned, ensure that the correct working branch is checked out. By default, older branches such as `2011_Compilation_PG11_PG2011-Final_Stage` may be selected.

- If necessary, switch to the active development branch: `2025_LibreSocial_Docker`.

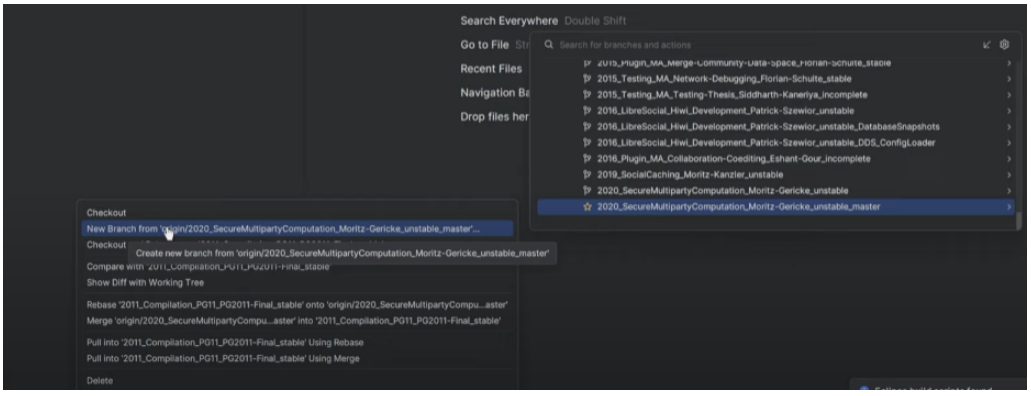


Figure 8: Switching to the `2025_LibreSocial_Docker` branch

3.3 Step 2: Installing Java JDK

LibreSocial is built on legacy Java technology and strictly depends on Java 7 (JDK version 1.7.0_80). To ensure compatibility with the Eclipse Indigo build environment and legacy classpaths, this exact version must be used. Modern versions of Java are incompatible and may result in class version or runtime errors.

2.1 Download and Install JDK 1.7.0_80

1. Navigate to the official Oracle Java Archive:
   `https://www.oracle.com/java/technologies/javase/javase7-archive-downloads.html`
2. Download the installer for your operating system:
   - Windows x64 or x86: `jdk-7u80-windows-x64.exe`
   - macOS: `jdk-7u80-macosx-x64.dmg`
   - Linux: `jdk-7u80-linux-x64.tar.gz`
3. Complete the installation using the system installer. The default path for installation is typically:
   - Windows: `C:\Program Files\Java\jdk1.7.0_80`
   - Linux/macOS: `/usr/lib/jvm/jdk1.7.0_80` (may vary by distribution)

2.2 Configure Environment Variables

To enable the JDK globally on your system, you must define two environment variables: `JAVA_HOME` and `PATH`.

Windows

- Open the Start Menu and search for `Environment Variables`.
- Click `Edit the system environment variables`, then `Environment Variables...`
- Add a new variable:
  - **Variable name:** `JAVA_HOME`
  - **Variable value:** `C:\Program Files\Java\jdk1.7.0_80`
- Append the following to the `Path` variable: `%JAVA_HOME%`
  `bin`

Linux/macOS

Add the following lines to your shell profile file (e.g., `.bashrc`, `.zshrc`):

```
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_80
export PATH=$JAVA_HOME/bin:$PATH
```

After saving, apply the changes with:

```
source ~/.bashrc
```

2.3 Verify JDK Installation

Open a new terminal or command prompt and run:

```
java -version
```

The expected output is:

```
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
```

This confirms the JDK is properly installed and accessible globally via the command line.

3.4 Step 3: Importing the Project into Eclipse Indigo

- Download and launch Eclipse Indigo (RCP/RAP version).
- Go to `File > Import > Existing Projects into Workspace`.
- Browse to the cloned LibreSocial folder and import the project.

3.4.1 Fixing Build Path or JRE Issues

- Go to `Project Properties > Java Build Path`.
- Ensure the JRE system library points to Java 1.7.

3.5 Step 4: Applying Unlimited JCE Policy Files

- Download the policy files from Oracle.
- Replace the existing JARs in:
  - Windows: `C:\Program Files\Java\jdk1.7.0_80\lib\security`
  - Ubuntu: `/usr/lib/jvm/java-7-oracle/jre/lib/security`
  - macOS: `/Library/Java/JavaVirtualMachines/jdk1.7.0_80.jdk/Contents/Home/jre/`
- Restart Eclipse after replacing the files.

3.6 Running the Project

- Right-click the main class or launcher in Eclipse.
- Select `Run As > Java Application`.
- Monitor the console or GUI output.

3.7 Conclusion

By following these steps, you should have a working setup of LibreSocial using its original legacy toolchain. Screenshots and command paths have been provided for clarity.

4 Troubleshooting Tips

Despite following all the instructions, certain legacy or environment-specific issues may arise. Below are common problems encountered and their recommended solutions:

4.1 JDK Version Mismatch

- **Error:** ‘Unsupported major.minor version‘ or ‘class file version‘ mismatch.
- **Cause:** Using Java 8+ instead of Java 7.
- **Solution:** Reinstall JDK 1.7.0.80, ensure 'JAVA$_HOME'and'PATH'pointtothecorrectlocation$.

4.2 Eclipse Project Not Recognized

- **Error:** Project does not load as a Java Project in Eclipse Indigo.
- **Cause:** ‘.project‘ or ‘.classpath‘ files missing or corrupted.
- **Solution:** Go to `File > New > Project > Java Project`, then manually import sources and reconfigure the build path.

4.3 Unresolved Libraries or Missing JARs

- **Error:** Classes like ‘org.apache.commons.*‘ or crypto packages not found.
- **Cause:** Missing libraries or unconfigured JCE policy.
- **Solution:**
  - Ensure all required JARs are in the `lib` folder.
  - Check ‘Java Build Path > Libraries‘ in Eclipse and add missing JARs manually.
  - Replace JCE policy files with unlimited strength ones.

4.4 GUI Not Launching or Runtime Crash

- **Cause:** SWT/JFace version incompatibility or missing plugins.
- **Solution:** Recheck plugin dependencies in the ‘manifest.mf‘, and confirm SWT libraries are added to the classpath.

5 Conclusion

This documentation has provided a comprehensive, step-by-step guide to setting up and running LibreSocial—a decentralized social networking framework built with legacy Java technologies. The core objective was to navigate the challenges of working with deprecated toolchains, outdated APIs, and complex dependency configurations.

By leveraging modern tools such as IntelliJ IDEA for version control, and carefully configuring Eclipse Indigo for development and execution, the project can be reliably restored in a contemporary environment. The guide also emphasizes practical troubleshooting techniques, environmental configurations, and secure cryptographic support through the JCE policy upgrade.

With this setup, developers and researchers can now build upon LibreSocial's existing infrastructure, extend its functionality, or run experimental deployments in distributed settings.

...

Appendix

A Useful Commands and Configuration Paths

Common Command Line Instructions

- `java -version` — Verify installed JDK version.
- `javac -version` — Confirm Java compiler version.
- `echo $JAVA_HOME` — Check if the JAVA_HOME environment variable is set (Linux/macOS).
- `git clone <repo-url>` — Clone the LibreSocial repository from GitLab.
- `source  /.bashrc` — Reload environment variables (Linux/macOS).

Important System Paths

- **Windows JDK Path:** `C:\Program Files\Java\jdk1.7.0_80`
- **Linux JDK Path:** `/usr/lib/jvm/jdk1.7.0_80`
- **JCE Security Files:** `$JAVA_HOME/lib/security` or `%JAVA_HOME%`
  `lib`
  `security`

Key URLs

- **JDK 1.7 Archive:** `https://www.oracle.com/java/technologies/javase/javase7-archive-downloads.html`
- **Eclipse Indigo IDE:** `https://archive.eclipse.org/eclipse/downloads/drops/R-3.7-201106131736/`
- **LibreSocial GitLab Repository:** `https://gitlab.rlp.net/k.graffi/libresocial`

Branch Reference

- `2025_LibreSocial_Docker` — Current and maintained branch.
- `2011_Compilation_PG11_PG2011-Final_Stage` — Deprecated legacy branch.