**Installation Liferay Developer in system**

* Install java jdk 8 java 1.8
  + Check multiple java installed chekc by this:
    - sudo update-alternatives --config java
  + sudo apt update
  + sudo apt-get install openjdk-8-jdk
  + java -version
* Install liferay developer tool
  + Go to projects>Trainee>liferay-training> create three folder code server workspace
  + Go to this <https://github.com/liferay/liferay-ide/releases/>
  + Install developer workspace for your system
  + Copy from download and create a system folder on root level and paste it here
  + Use this file go to properties>permissions>execute checkbox: true and close it
  + Click on this file
  + Choose a jdk version
  + Select installation directory as on same directory
  + Open new folder click on developer studio
  + Select location for launching workspace as projects>Trainee>liferay-training> workspace
  + Install liferay plugin on dev-studio
    - Go to https://liferay.dev/project/-/asset\_publisher/TyF2HQPLV1b5/content/ide-installation-instructions
    - On above site install box just drag and drop to dev-studio
    - And check it by window>preference>seee the liferay on left side
  + On developer studio File>new>Liferay workspace Project
    - Project name: lr-workspace
    - Uncheck default location and set location as projects>Trainee>liferay-training>code
    - Choose gradle
    - Product-version: portal-7.4-ga50
    - Finish
  + Create liferay module workspace as
    - Project name: employee as employee-learning
    - Project template name: service builder
    - Build type: gradle
    - Click on next
    - Package name: com.stpl.learning.employee (com.companyname.purpose.whom)
    - Save
  + Build the project
  + On dev-studio topbar window>show view>other>gradle task open
  + Open terminal go to project>module >right click select show in terminal
  + lr-workspace>build>buildService **OR** Terminal gw clean build/buildService/deploy

**Connect Liferay to MySQL using local server**

* Check on the gradle.properties file
  + Liferay.workspace.product = <version>
* Select a particular version <https://github.com/liferay/liferay-portal/releases/tag/7.4.3.43-ga43>
* Select for ubuntu [liferay-ce-portal-tomcat-7.4.3.43-ga43-20220921132136623.tar.gz](https://github.com/liferay/liferay-portal/releases/download/7.4.3.43-ga43/liferay-ce-portal-tomcat-7.4.3.43-ga43-20220921132136623.tar.gz)
* Extract on folder on project server
* Add portal-ext.properties on server add content
  + include-and-override=portal-developer.properties
  + jdbc.default.driverClassName=com.mysql.cj.jdbc.Driver
  + <jdbc.default.url=jdbc:mysql://<hostname>:<port>/<schema-name>>
  + jdbc.default.url=jdbc:mysql://localhost:3306/training-db
  + jdbc.default.username=root
  + jdbc.default.password=root
  + Save it
* Add server/<tomcate-path> into gradle-local.properties
  + liferay.workspace.home.dir=/home/st22122101/Projects/Trainee/liferay-service-builder/server/liferay-ce-portal-7.4.3.43-ga43/
* For check the log of liferay server
  + Till bin path and ./startup.sh
  + cd Projects/Trainee/liferay-service-builder/server/liferay-ce-portal-7.4.3.43-ga43/tomcat-9.0.56/logs$ tail -f catalina.out
* Start tomcat
  + Till bin path and ./startup.sh
  + cd Projects/Trainee/liferay-service-builder/server/liferay-ce-portal-7.4.3.43-ga43/tomcat-9.0.56/bin$ ./startup.sh

**Liferay Doc**

* <https://www.opensource-techblog.com/complete-liferay-guide>
* Official doc: https://help.liferay.com/hc/en-us/articles/360017886672-Understanding-the-Code-Generated-by-Service-Builder-

**Understanding of service builder**

* Service Builder to generate your persistence framework and implement your business logic
* It’s a model-driven code generator for custom object models are entities
* It generates a service layer through object-relational mapping(ORM) that separates between object modal and database
* It takes XML file for creating a service layer
* It’s used for creating crud operations and finding data on database
* It gives model, persistence(provide internal database for liferay), service layer, remote or local service, hibernate and spring configuration, generating a find method for entities and account permissions, save developers time
  + Model layer – define an object that provides entities
  + Persistence layer – saving entities and retrieving data
  + Service Layer – responsible for CRUD operation and creating entities as API
  + Each entity contains model, local or remote service implementation class by service.XML (Logic added in three classes)
  + The local service – responsible for the persistence layer
  + The Remote service – Additional code for permissions and accessible over the internet (Service automatically generates the code to allowing remote services)
  + Save time by hibernating and spring configuration
  + Service builder uses spring dependency for making service implementation class at runtime
  + Spring AOP – database transaction management
  + Hibernate persistence framework – ORM
  + Generating a find method for entities and account permissions,
  + Liferay catches objects at three layers – entity, finder, Hibernate
  + Gradle service builder has Liferay service builder dependency
* Generating persistence layer by auto-creation of interface and class
* The persistence layer persists data by configured entities

**Local Service Understanding File Content**

* \*-API: all content is in interfaces like Persistence
* \*-service: interfaces are implemented here like PersistenceImpl
* LocalService: interface
* LocalServiceBaseImpl: base implementation of local service and its abstract class add persistence class here as @Reference class with object
* LocalServiceImpl: implements interfaces and adds logic here and if you create

the new method it auto add on LocalService **OR** Liferay gives basic crud method if you add a method to put on here where build create same structure on util file for this method

* LocalServiceUtil: this utility class wrap LocalServiceImpl and Use the \*LocalService class by referencing it with the [@Reference annotation](https://help.liferay.com/hc/en-us/articles/360018168491-OSGi-Services-and-Dependency-Injection-with-Declarative-Services-).

Specially for access early version in case any other computer have different version and earlier of version easily to delaware content **OR** its instance of PersistenceImpl

* LocalServiceWrapper: this class wrap LocalService andand it lets you [customize the entity’s local services](https://help.liferay.com/hc/en-us/articles/360018165631-Overriding-Liferay-Services-Service-Wrappers-).(inshort override the local service implementation of baseserviceImpl class)
* ModelResourceImpl: created content from different sources and create response of an object {like u want to get username,email,status etc.}
* Employee: interface
* EmployeeModel: interface **OR** create get set method for column of service.xml
* EmployeeImpl: implements of employee
* EmployeeModelImpl: implements of employeeModel

**Liferay Finder DOC:** https://www.opensource-techblog.com/create-finder-method-for-service.html

* Finder
  + return-type: Gives collection of that
  + name: The only thing you need to make sure it to give first letter as capital.
    - Finder-column name: as same as column you want to find
* Finder after build we show code in persistenceimpl file we can’t direct access in util so for
  + Open LocalServiceImpl
  + created a method and build
    - **public** List<entity-model-name> nameOfMethod(**type name**){

**return** **this**.[entity-name]PersistenceImpl.method-name(name);

}

public List<Employee> getEmployeeByGender(String gender) {

return this.employeePersistence.findByGender(gender);

}

* + Check method in LocalServiceUtil
* We only access LocalServiceUtil class outside the service builder

**Foreign key :**

* DOC: https://stackoverflow.com/questions/24273631/liferay-service-builder-6-2-many-to-one-relationships

**Note::**

* Open gradle task : window-> show view->gradle task(search)
* If an entity exists and u updated service.xml please remove all src/\* packages ->build & deploy
* Refresh content in eclipse F5
* if u change anything in currentMethod of LocalServiceimpl it will not affect on util file so util create a new method
* Searching for entities using search engine faster than searching on database

**Folder Structure**

\*-api

* src/main/java
  + Exception
    - NoSuch<name>Exception
  + Model
    - Foo
    - FooModel
    - FooTable
    - FooWrapper
  + Service
    - FooLocalService
    - FooLocalServiceUtil
    - FooLocalServiceWrapper
  + Service/Persistence
    - FooPersistence
    - FooUtil

\*-Service

* src/main
  + Java
    - Model
      * Impl
        + FooBaseimpl
        + FooCatchModel
        + FooImpl
        + FooModelImpl
  + resource
    - Service
      * Base
        + FooLocalserviceBaseImpl
        + FooServiceBaseImpl
      * impl
        + FooLocalServiceImpl
        + FooServiceImpl
      * Persistence
        + Impl

FooModelArgumentsResolver

FooPersistenceImpl

* + - * + Impl/Constants

persistenceConstants

**Model structure**

****

**Local service structure**



**Persistence Picture Understanding**



**Remote Service Understanding File Content**

* Difference between local and remote: is local assume permission and chek before it’s called

|  |  |
| --- | --- |
| Local Service | Remote Service |
| * It assumes permission before it’s built * Contain business logic bug * Have LocalServiceImpl | * We write code for permission * Containing permission logic bug * Have ServiceImpl |