workspace

WorkspaceWorkObjectPool

WorkObjectPool

WorkObjectPool

```
/**
 *
 */
public interface WorkObjectPool {

    /**
     *
     */
    <T> T get(Class<T> type);

    void close();
}
```

WorkObjectPool

- WorkObjectPool
- WorkspaceConnectorfine-coreWorkspaceClient

WorkspaceClient

```
public interface WorkspaceClient {

    /**
     *
     */
    WorkObjectPool getPool();

    /**
     *
     */
    WorkspaceConnection getConnection();

    /**
     *
     */
    void close();
}
```

WorkspaceConnector

```
/**
 *
 */
public interface WorkspaceConnector {
    /**
     *
     */
    boolean testConnection(WorkspaceConnectionInfo connection) throws Exception;
    /**
     * client
     */
    WorkspaceClient connect(WorkspaceConnectionInfo connection) throws Exception;

}
```

- WorkspaceClient“WorkObjectPool”RPC

WorkObjectPool

activator

WorkObjectPool

```
public class DemoActivator extends Activator implements Prepare {

    @Override
    public void start() {
                //
    }

    @Override
    public void stop() {
            //
    }

    @Override
    public void prepare() {
        //WorkObjectPool
        addMutable(WorkspaceKey.RPC,
            WorkPRCRegister.wrap(WorkRPCType.Local, WorkResource.class, FineWorkResource.getInstance()),
            WorkPRCRegister.wrap(WorkspaceHeartbeat.class, new FineWorkspaceHeartbeat()),
            WorkPRCRegister.wrap(FileAssistUtilsOperator.class, new FileAssistUtils())

        );
    }
}
```

WorkPRCRegisterWorkRPCType

```
/**
 *
 */
public enum WorkRPCType {

    /**
     *
     */
    Simple,
    /**
     * Server
     */
    Local,
    /**
     *
     */
    Server,
}

/**
* WorkObject
*/
public class WorkPRCRegister<T> {

    private final Class<T> clazz;

    private final WorkRPCType type;

    private final T object;

    public static <T> WorkPRCRegister<T> wrap(Class<T> clazz, T object) {

        return new WorkPRCRegister<T>(clazz, WorkRPCType.Simple, object);
    }

    public static <T> WorkPRCRegister<T> wrap(WorkRPCType type, Class<T> clazz, T object) {

        return new WorkPRCRegister<T>(clazz, type, object);
    }

    private WorkPRCRegister(Class<T> clazz, WorkRPCType type, T object) {

        this.clazz = clazz;
        this.type = type;
        this.object = object;
    }
}
```

WorkRPCType

- Simple
- Local
- Server

Simple/LocalServer

WorkObjectPool

WorkObjectPool

```
WorkContext.getCurrent().get(CommonOperator.class);
```

WorkContextWorkspaceWorkspaceRPC

WorkspaceClient

```
WorkspaceConnectionInfo connectionInfo = new WorkspaceConnectionInfo(url,userName,password,certPath,
certSecretKey);
WorkspaceClient client = WorkContext.getConnector().connect(connectionInfo);
```

Connection

```
//
WorkspaceConnection connection = WorkspaceServerContext.currentConnection()
```

WorkObjectPoolSerializerSummarySerializerSummary

```
public class DemoActivator extends Activator implements Prepare {

    @Override
    public void start() {
                //
    }

    @Override
    public void stop() {
            //
    }

    @Override
    public void prepare() {
       //
                addMutable(CommonSerializerKey.KEY,(new AbstractCommonSerializer<Demo2>() {

            @Override
            public void serialize(Demo2 obj, OutputStream out) throws Exception {

                DataOutputStream dataOut = new DataOutputStream(out);
                dataOut.writeInt(obj.i);
            }

            @Override
            public Demo2 deserialize(InputStream in) throws Exception {

                DataInputStream dataIn = new DataInputStream(in);
                int i = dataIn.readInt();
                return new Demo2(i);
            }

            @Override
            public boolean accept(Object o) {

                return o instanceof Demo2;
            }
        });
    }
}
```

WorkObjectPool

WorkRPCProvider

WorkspaceChannelWorkspaceClient

todo