com.fr.io.fun.ResourceRepositoryFactoryProvider

-
-
-
-
-
-
-
-
-

FTPSFTP

assetsreportletsresourcesscheduledashboardstreasuresbackup ResourceRepositoryFactoryProviderOSSHDFS

ResourceRepositoryFactoryProvider.java

```
package com.fr.io.fun;

import com.fr.io.base.provider.RepositoryFactoryProvider;
import com.fr.io.config.RepositoryConfig;
import com.fr.stable.fun.mark.Mutable;
import com.fr.stable.script.CalculatorKey;

/**
 *
 * <p>
 * Created by rinoux on 2018/3/13.
 */
public interface ResourceRepositoryFactoryProvider<T extends RepositoryConfig> extends Mutable {

    CalculatorKey KEY = CalculatorKey.createKey(ResourceRepositoryFactoryProvider.class.getName());

    String MARK_STRING = "ResourceRepositoryFactoryProvider";

    RepositoryFactoryProvider<T> getFactory();
}
```

RepositoryConfig.java

```
package com.fr.io.config;

import com.fr.common.annotations.Open;
import com.fr.config.utils.UniqueKey;
import com.fr.io.base.provider.Updatable;
import com.fr.third.fasterxml.jackson.annotation.JsonIgnoreProperties;

/**
 * Created by rinoux on 2018/5/15.
 */
@JsonIgnoreProperties({"id", "data", "classInfo", "nameSpace"})
@Open
public abstract class RepositoryConfig extends UniqueKey implements Updatable {

    private String identity;

    public RepositoryConfig() {}

    public RepositoryConfig(String identity) {
        this.identity = identity;
    }


    public String getIdentity() {
        return identity;
    }

    public void setIdentity(String identity) {
        this.identity = identity;
    }

    @Override
    public Object clone() throws CloneNotSupportedException {
        RepositoryConfig cloned = (RepositoryConfig) super.clone();
        cloned.setIdentity(this.identity);

        return cloned;
    }
}
```

RepositoryFactoryProvider.java

```
package com.fr.io.base.provider;

import com.fr.common.annotations.Open;
import com.fr.io.config.RepositoryConfigManagerProvider;
import com.fr.io.config.RepositoryConfig;
import com.fr.io.context.info.RepositoryProfile;
import com.fr.io.repository.ResourceRepository;

import java.io.Serializable;

/**
 * Created by rinoux on 2018/3/20.
 */
@Open
public interface RepositoryFactoryProvider<T extends RepositoryConfig> extends Serializable {

    /**
     *
     *
     * @return
     */
    String getIdentity();
```

```
    /**
     *
     *
     * @return
     */
    RepositoryConfigManagerProvider<T> getConfigManager();


    /**
     * profile
     *
     * @return
     */
    Class<? extends RepositoryProfile<T>> getProfileClass();


    /**
     *
     *
     * @return
     */
    Class<T> getConfigClass();


    /**
     *
     *
     * @param config
     * @return
     */
    boolean verifyConfig(T config);


    /**
     *
     *
     * @param repoName
     * @param workRoot
     * @param config
     * @return
     */
    ResourceRepository produce(String repoName, String workRoot, T config);


    /**
     *
     *
     * @param repoName
     * @param workRoot
     * @return
     */
    ResourceRepository produce(String repoName, String workRoot);
}
```

RepositoryInfoProvider.java

```
package com.fr.io.base.provider;

/**
 * Created by rinoux on 2018/3/8.
 */
public interface RepositoryInfoProvider {

    /**
     *
     *
     * @return
     */
    String getRepoName();

    /**
     *
     *
     * @param repoName
     */
    void setRepoName(String repoName);

    /**
     * (FTP,OSS)
     *
     * @return getIdentity
     */
    String getIdentity();


    /**
     *
     *
     * @return workRoot
     */
    String getWorkRoot();


    /**
     *
     *
     *
     * @return
     */
    boolean isAccurateDiskSize();
}
```

WorkResource.java

```
package com.fr.workspace.resource;

import com.fr.stable.Filter;
import com.fr.workspace.base.WorkspaceAPI;
import com.fr.workspace.base.WorkspaceConstants;
import org.jetbrains.annotations.Nullable;

import java.io.InputStream;

/**
 * Created by juhaoyu on 2018/6/14.
 *
 */
@WorkspaceAPI(description = "Fine-Core_Workspace_Description_Of_Work_Resource")
public interface WorkResource {
```

```
/**
 *
 */
byte[] readFully(String path);


/**
 *
 *
 * @param path
 * @return null
 */
@Nullable
InputStream openStream(String path);

/**
 *
 */
void write(String path, byte[] data);

/**
 *
 *
 */
boolean createFile(String path);

/**
 *
 */
boolean createDirectory(String path);

/**
 *
 */
boolean delete(String path);

/**
 *
 */
boolean rename(String from, String to);

/**
 *
 */
boolean exist(String path);


/**
 * name
 *
 * @param dir
 * @return
 */
String[] list(String dir);


/**
 * name
 *
 * @param dir
 * @param filter
 * @return
 */
String[] list(String dir, Filter<String> filter);


/**
 *
 *
 * @param path
 * @return
```

```
     */
    boolean isDirectory(String path);



    /**
     *
     *
     * @param path
     * @return mills
     */
    long lastModified(String path);


    /**
     *
     *
     * @param path
     * @return
     */
    long length(String path);

    /**
     * {@link this#write(String, byte[])}, {@link this#delete(String)}  {@link this#rename(String, String)}
     *
     * @param tmpPath tmp
     * @param path path
     * @param content
     * @return
     */
    default void save(String tmpPath, String path, byte[] content) {
        //
        write(tmpPath, content);
        //
        delete(path);
        //
        rename(tmpPath, path);
    }
}
```

ResourceRepository.java

```
package com.fr.io.repository;


import com.fr.io.base.provider.RepositoryInfoProvider;
import com.fr.stable.Filter;
import com.fr.workspace.resource.ResourceIOException;
import com.fr.workspace.resource.WorkResource;

import java.io.InputStream;
import java.net.URL;

/**
 * .
 */
public interface ResourceRepository extends RepositoryInfoProvider, WorkResource {

    /**
     *
     *
     * @return
     */
    String getSeparator();


    /**
     * entry
```

```
     *
     * @param path
     * @return FineFileEntry null
     */
    FineFileEntry getEntry(String path);


    /**
     * FineFileEntry
     *
     * @param dir
     * @return FineFileEntry
     */
    FineFileEntry[] listEntry(String dir);

    /**
     * url
     *
     * @param path
     * @return url
     */
    URL getResource(String path);

    /**
     *
     * <p>
     * InputStreamclient
     *
     * @param file
     * @return  null
     * @throws ResourceIOException read
     */
    InputStream read(String file) throws ResourceIOException;

    /**
     * filter
     * <p>
     * filternullDefaultFilter
     * <p>
     * Listdirfiltername,path
     * <p>
     *
     *
     * @param dir
     * @param filter
     * @return
     */
    @Override
    String[] list(String dir, Filter<String> filter);

    /**
     *
     * <p>
     * Listdirfiltername,path
     * <p>
     *
     *
     * @param dir
     * @return
     */
    @Override
    String[] list(String dir);

    /**
     *
     * <p>
     *
     * <p>
     * datawrite
     *
     * @param file path
```

```
     * @param data
     * @throws ResourceIOException
     */
    void write(String file, InputStream data) throws ResourceIOException;

    /**
     *
     * <p>
     *
     *
     * @param file
     * @param data
     * @throws ResourceIOException
     */
    void appendWrite(String file, InputStream data) throws ResourceIOException;

    /**
     *
     * <p>
     *
     *
     * @param file
     * @param data
     * @throws ResourceIOException
     */
    void appendWrite(String file, byte[] data) throws ResourceIOException;

    /**
     *
     * <p>
     * origPathdesPath
     * <p>
     * desPathfalse
     *
     * @param origPath
     * @param desPath
     * @return copy
     * @throws ResourceIOException
     */
    boolean copy(String origPath, String desPath) throws ResourceIOException;

    /**
     *
     * <p>
     * key"/"
     *
     * @param path
     * @return
     */
    @Override
    boolean isDirectory(String path);



    /**
     *
     * <p>
     *
     */
    void shutDown();

    /**
     *
     *
     * @return <code>true</code>
     */
    default boolean authentication(){
        return true;
    }
```

```
}
```

RepositoryProfile.java

```
package com.fr.io.context.info;

import com.fr.cluster.ClusterBridge;
import com.fr.common.annotations.Open;
import com.fr.config.Identifier;
import com.fr.config.holder.Conf;
import com.fr.config.holder.factory.Holders;
import com.fr.config.holder.impl.MapConf;
import com.fr.config.utils.UniqueKey;
import com.fr.io.base.exception.RepositoryException;
import com.fr.io.base.provider.RepositoryFactoryProvider;
import com.fr.io.base.provider.Updatable;
import com.fr.io.config.RepositoryConfig;
import com.fr.io.context.ResourceModuleContext;
import com.fr.stable.StringUtils;

import java.io.Serializable;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

/**
 * Created by rinoux on 2018/3/26.
 */
@SuppressWarnings("all")
@Open
public abstract class RepositoryProfile<T extends RepositoryConfig> extends UniqueKey implements Serializable,
Updatable {

    /**
     *
     */
    @Identifier("identity")
    private Conf<String> identity = Holders.simple(StringUtils.EMPTY);

    /**
     *
     */
    @Identifier("repoName")
    private Conf<String> repoName = Holders.simple(StringUtils.EMPTY);
    /**
     *
     */
    @Identifier("shared")
    private Conf<Boolean> shared = Holders.simple(Boolean.FALSE);
    /**
     * workroot[id, workroot]
     */
    @Identifier("workroots")
    private MapConf<Map<String, String>> workroots = Holders.map(new HashMap<String, String>(), String.class,
String.class);


    public RepositoryProfile() {
    }

    public RepositoryProfile(String identity, String repoName, String workRoot) {
        init(identity, repoName, workRoot, false);
    }


    public RepositoryProfile(String identity, String repoName, String workRoot, T config) {
        init(identity, repoName, workRoot, false);
```

```
}

public RepositoryProfile(String identity, String repoName, String workRoot, boolean shared) {
    init(identity, repoName, workRoot, shared);
}


public RepositoryProfile(String identity, String repoName, String workRoot, boolean shared, T config) {
    init(identity, repoName, workRoot, shared);
}

private void init(String identity, String repoName, String workRoot, boolean shared) {
    this.identity.set(identity);
    this.repoName.set(repoName);
    this.shared.set(shared);
    this.setWorkRoot(workRoot);
}


public String getIdentity() {
    return identity.get();
}

public void setIdentity(String identity) {
    this.identity.set(identity);
}

public String getRepoName() {
    return repoName.get();
}

public void setRepoName(String repoName) {
    this.repoName.set(repoName);
}

public String getWorkRoot() {
    String currentNodeId = ClusterBridge.getView().getCurrent().getID();
    if (workroots.containsKey(currentNodeId)) {
        return (String) workroots.get(currentNodeId);
    }
    else if (isShared()) {
        Iterator<String> iterator = workroots.get().values().iterator();
        while (iterator.hasNext()) {
            return iterator.next();
        }
    }

    return null;
}

public void setWorkRoot(String workRoot) {
    String currentNodeId = ClusterBridge.getView().getCurrent().getID();
    if (isShared()) {
        //  workroots
        Set<String> ids = workroots.get().keySet();
        for (String id : ids) {
            workroots.put(id, workRoot);
        }
    }
    //
    this.workroots.put(currentNodeId, workRoot);
}

public boolean isShared() {
    return shared.get();
}

public void setShared(boolean shared) {
    this.shared.set(shared);
}
```

```
public abstract T getConfig();

public abstract void setConfig(T config);



@Override
public RepositoryProfile<T> clone() throws CloneNotSupportedException {
    RepositoryProfile<T> cloned = (RepositoryProfile<T>) super.clone();
    cloned.identity = (Conf<String>) this.identity.clone();
    cloned.repoName = (Conf<String>) this.repoName.clone();
    cloned.shared = (Conf<Boolean>) this.shared.clone();
    cloned.workroots = (MapConf<Map<String, String>>) this.workroots.clone();

    return cloned;
}

/**
 *
 *
 * @return
 * @throws RepositoryException
 */
public InstalledComponent<T> install() {
    return ResourceModuleContext.install(this);
}


private boolean validate() {
    if (StringUtils.isEmpty(getIdentity())) {
        return false;
    }

    if (StringUtils.isEmpty(getRepoName())) {
        return false;
    }

    // workroot
    if (!isShared() && StringUtils.isEmpty(getWorkRoot())) {
        return false;
    }

    // workroot
    if (isShared() && workroots.get().isEmpty()) {
        return false;
    }

    return true;
}


/**
 *
 *
 * @return
 */
public boolean suitable() {
    if (ResourceModuleContext.getFactory(getIdentity()) == null) {
        return false;
    }
    return validate();
}

@Override
public final void update(String srcRepo) {
    InstalledComponent c = ResourceModuleContext.getInstalledComponent(srcRepo);
    this.setWorkRoot(c.getWorkRoot());
    this.setShared(c.isShared());
    if (getConfig() != null) {
        this.getConfig().update(srcRepo);
    }
```

```
    }

    /**
     *
     *
     * @return
     */
    public final InstalledComponent<T> generateComponent() {
        RepositoryFactoryProvider<T> fac = ResourceModuleContext.getFactory(getIdentity());
        return new InstalledComponent<T>(fac, getRepoName(), getWorkRoot(), isShared());
    }

}
```

JSCSSJS

```
!(function () {
    BI.config("dec.constant.intelligence.cluster.file.server", function (items) {
        items.push({
            value: "DEMO", //RepositoryConfig#identity
            id: "decision-intelligence-cluster-file-demo",// ID
            text: "DEMO REPOSITORY",//
            cardType: "dec.intelligence.cluster.file.demo"//
        });
        return items;
    });
        //getValue/validation
    var LABEL_WIDTH = 116, EDITOR_WIDTH = 300;
    var DEMO = BI.inherit(BI.Widget, {
        ...

        render: function () {
            var self = this, o = this.options;
            return {
                type: "bi.vertical",
                tgap: 15,
                items: [
                    {
                        type: "dec.label.editor.item",
                        textWidth: LABEL_WIDTH,
                        editorWidth: EDITOR_WIDTH,
                        watermark: BI.i18nText("Dec-Please_Input"),
                        text: "ROOT",
                        value: this.model.root,//
                        ref: function (_ref) {
                            self.rootRow = _ref;
                        },
                        listeners: [{
                            eventName: BI.Editor.EVENT_CHANGE,
                            action: function () {
                                self.store.setRoot(this.getValue());
                            }
                        }]
                    }]
            };
        },
        getValue: function () {//getValueconfigconfigGetConifgSetConfigvalue
            return {
                root: this.model.root//
            };
        },
        validation: function () {//
            var valid = true, root = this.model.root;
            if (BI.isEmpty(root)) {
                this.rootRow.showError(BI.i18nText("Dec-Error_Null"));
                valid = false;
            }
            return valid;
        }
    });
    ...
})();
```

FR	10.0		

plugin.xml

```
<extra-core>
        <ResourceRepositoryFactoryProvider class="your class name"/>
</extra-core>
```

ResourceRepository

ResourceRepositoryFactoryProvider

RepositoryConfig

RepositoryProfileRepositoryConfig

demodemo-resource-repository-factory-provider

JSCSS