Implement Recovery Techniques

This page is for Odysseus developers and shows how to add new recovery techniques to Odysseus.

A recovery technique is in Odysseus a composition of smaller recovery components. Processing Image and BaDaSt are for example two different recovery components. The rollback recovery is a composition of both.

To implement a new recovery component, you have to implement the interface IRecoveryComponent:

IRecoveryComponent

```
/**
 * A recovery component handles the backup and recovery of certain information
 * (e.g., installed queries).
 *
 * @author Michael Brand
 *
 */
public interface IRecoveryComponent {

	/**
	 * Initializes the component with a given configuration.
	 *
	 * @param config
	 *            the configuration for the component.
	 */
	public void initialize(Properties config);

	/**
	 * Runs the recovery mechanism for given queries.
	 *
	 * @param qbConfig
	 *            The used query build configuration.
	 * @param session
	 *            The session of the user, who wants to back up the data.
	 * @param queries
	 *            The queries.
	 * @param caller
	 *            The executor that called this method.
	 * @return {@code queries} either modified for recovery or not. Depends on
	 *         the used recovery strategy.
	 */
	public List<ILogicalQuery> activateRecovery(QueryBuildConfiguration qbConfig, ISession session,
			List<ILogicalQuery> queries, IExecutor caller);

	/**
	 * Activates the backup mechanism for given queries.
	 *
	 * @param qbConfig
	 *            The used query build configuration.
	 * @param session
	 *            The session of the user, who wants to back up the data.
	 * @param queries
	 *            The queries.
	 * @param caller
	 *            The executor that called this method.
	 * @return {@code queries} either modified for recovery or not. Depends on
	 *         the used recovery strategy.
	 */
	public List<ILogicalQuery> activateBackup(QueryBuildConfiguration qbConfig, ISession session,
			List<ILogicalQuery> queries, IExecutor caller);

}
```

To compose a new recovery technique, you have to implement the interface IRecoveryExecutor:

IRecoveryExecutor

```
/**
 * A recovery executor represents a complete non-distributed recovery (NDR)
 * strategy by calling certain {@link IRecoveryComponent}s in a certain order.
 *
 * @author Michael Brand
 *
 */
public interface IRecoveryExecutor {

	/**
	 * Gets the name of the executor.
	 *
	 * @return A string unique for recovery executors.
	 */
	public String getName();

	/**
	 * Creates a new recovery executor with a given configuration.
	 *
	 * @param config
	 *            the configuration for the executor. Typically, it is the
	 *            totality of configurations for the
	 *            {@link IRecoveryComponent}s.
	 * @return A new recovery executor instance.
	 */
	public IRecoveryExecutor newInstance(Properties config);

	/**
	 * Runs the NDR mechanism for given queries.
	 *
	 * @param qbConfig
	 *            The used query build configuration.
	 * @param session
	 *            The session of the user, who wants to recover the data.
	 * @param queries
	 *            The queries to recover.
	 * @param caller
	 *            The executor that called this method.
	 * @return {@code queries} either modified for recovery or not. Depends on
	 *         the used recovery strategy.
	 */
	public List<ILogicalQuery> activateRecovery(QueryBuildConfiguration qbConfig, ISession session,
			List<ILogicalQuery> queries, IExecutor caller);

	/**
	 * Activates the backup mechanism for given queries.
	 *
	 * @param qbConfig
	 *            The used query build configuration.
	 * @param session
	 *            The session of the user, who wants to back up the data.
	 * @param queries
	 *            The queries to backup.
	 * @param caller
	 *            The executor that called this method.
	 * @return {@code queries} either modified for backup or not. Depends on the
	 *         used recovery strategy.
	 */
	public List<ILogicalQuery> activateBackup(QueryBuildConfiguration qbConfig, ISession session,
			List<ILogicalQuery> queries, IExecutor caller);

	/**
	 * Checks, if a recovery is needed.
	 *
	 * @return True, if
	 *         {@link #activateRecovery(QueryBuildConfiguration, ISession, List)}
	 *         should be called.
	 */
	public boolean isRecoveryNeeded();

}
```

But typically the abstract class AbstractRecoveryExecutor can be used as super class.

After implementing a new recovery executor, it must be declared as an OSGi declarative service that provides an implementation of IRecoveryExecutor.

If you want to use the checkpointing component to get a notification when a checkpoint is reached, you have to register like in the following example:

Usage of Checkpointing

```
RollbackRecoveryExecutor instance = new RollbackRecoveryExecutor();
List<IRecoveryComponent> components = new ArrayList<>();
components.add(new CheckpointingRecoveryComponent());
components.add(new ProcessingImageRecoveryComponent());
components.add(new BaDaStRecoveryComponent());
components.add(new DuplicatesDetectorRecoveryComponent());
instance.init(config, components);
((CheckpointingRecoveryComponent) instance.components.get(0))
                .addCheckpointManagerListener((ProcessingImageRecoveryComponent) instance.components.get(1));
return instance;
```