Permission checks

Permission checks must always be included into Command execution methods.

AssetPermissions and PermissionService

The permissions must always be checked by the server. However, there some permission checks that can be done by the client. Example: disabling special buttons/actions, while the server re-checks the permissions.

AssetPermissions

Use the `AssetPermissions` service interface to perform permission checks in the context of an asset or to a set of assets:

```
AssetPermissions permService = Platform.getCCServiceEx(AssetPermissions.class);
```

The interface provides methods for Asset specific and Domain logic specific permission checks:

```
public interface AssetPermissions {
    //Generic asset specific permission checks (key = permission key)
    public boolean hasAssetPermission(String key, AssetRef asset);
    public boolean hasAssetPermission(String key, AssetRef[] assets);
    public boolean hasAssetPermission(String key, Asset asset);
    public boolean hasAssetPermission(String key, Asset[] assets);

    //Generic asset specific permission checks (Any/All checks with list of permission keys)
    public boolean hasAnyAssetPermission(String[] keys, AssetRef asset);
    public boolean hasAllAssetPermissions(String[] keys, AssetRef asset);
    public boolean hasAnyAssetPermission(String[] keys, Asset asset);
    public boolean hasAllAssetPermissions(String[] keys, Asset asset);

    //Domain logic specific permission checks
    public boolean getPermissionDeletion(AssetRef[] assets, int state);
    public boolean getPermissionAssetRel(AssetRef parentAsset, RelationType relType);
    public boolean getPermissionAssetRel(RelationType relType);
    public boolean getPermissionCheckOut(AssetRef[] assets);
    public boolean getPermissionExportAll(AssetRef[] assets);
    public boolean getPermissionReplaceStorages(AssetRef[] assets);
    public boolean getPermissionDeletionStorages(AssetRef[] assets);
    public boolean getPermissionShowVersions(AssetRef[] assets);
    public boolean getPermissionRestoreVersion(AssetRef[] assets);
}
```

PermissionService

Use the `PermissionService` interface to perform global (user domain specific) permission checks:

```
PermissionService permissionService = Platform.getCCService(PermissionService.class);
```

The `PermissionService` interface provides methods to check for global permissions:

```
public interface PermissionService {
    ...
    public boolean hasPermission(String permissionKey);
    ...
}
```

Example

```
@CommandHandler(command = "com.censhare.api.dam.assetmanagement.moveToNextWorkflowStep")
public static final class MoveToNextWorkflowStepHandler {
    @Execute
    public Result execute(CommandContext context, Input input) throws Exception {
        AssetRef assetRef = AssetRef.fromString(input.assetRef);
        ...
        // Check permission
        AssetPermissions permService = Platform.getCCServiceEx(AssetPermissions.class);
        if (!permService.getPermissionEditAssetMetaData(new AssetRef[] { assetRef }, true))
            throw new IllegalArgumentException(
                "Access denied: No permission to edit meta data (changing workflow) of asset "
                + asset);
        ...
    }
}
```

User permissions

Checking a user's permission is an important part when implementing a **Command Handler**. Every request sent by the client should be verified by the server, since:

- The command can be executed by different users within different roles. Some of the users may have the right to execute the action, others not.
- Even if permission checks are already done on client side, the server should never trust the client blindly. The client may actually be hacked.

If you want to check a user's global permissions, you can use the service **PermissionService**. It checks if the current user has the given right in any of his roles and domains (global administrator permission in this example):

```
@Execute
public ResultData execute(CommandContext context, InputData input) {
    PermissionService permService = Platform.getCCService(PermissionService.class);
    if (!permService.hasPermission("app_admin_all"))
        throw new SecurityException("No permission");
    ...
}
```

One can also check for a list of permissions within one single service call. The method returns true if the user has at least one of the given permissions (OR condition):

```
@Execute
public ResultData execute(CommandContext context, InputData input) {
    PermissionService permService = Platform.getCCService(PermissionService.class);
    if (!permService.hasAnyPermission("app_admin_all", "app_admin_data"))
        throw new SecurityException("No permission");
    ...
}
```

There's also an alternative way to check the user's permissions. The necessary permission keys can be specified as a parameter of the `@Execute` annotation. If the user does not have at least one of the specified rights, an exception is thrown if the client makes an attempt to execute the method. Checking permissions that way is more elegant and should be used preferably. However, it can only be used for global permission checks, not for asset-specific permission checks. Using permission parameter, the example from above can be rewritten and simplified like this:

```
@Execute(permissions="app_admin_all, app_admin_data")
public ResultData execute(CommandContext context, InputData input) {
    ...
}
```

Typically it's necessary to check the permissions not only in the context of the current user, but also in the context of a particular asset. The domains of the given asset are then used to filter the roles of the user. If the user does not have the appropriate rights in those domains, the action must fail. For that purpose, you have to use the service **AssetPermissions** and provide the asset object (or its key) as the second parameter in the service call. Example: Check if the current user has the right to see the preview of a particular asset:

```
Asset asset = queryAsset();
AssetPermissions permService = Platform.getCCService(AssetPermissions.class);
if (!permService.hasAssetPermission("asset_preview", asset))
    throw new SecurityException("No permission");
...
```

The service interface also provides special functions for the most common permission checks on assets, e.g. permission for check out, editing meta data or deletion. Those methods should be used in favor if available, since additional checks are made internally. Example: Check if the current user has the right to open (check out) a particular asset:

```
Asset asset = queryAsset();
AssetPermissions permService = Platform.getCCService(AssetPermissions.class);
if (!permService.getPermissionCheckOut(asset.getSelf()))
    throw new SecurityException("No permission");
...
```