Nimbus some pts:

Client:

**public** **class** ClientAccessEntity **extends** AbstractEntity.IdString {

**private** **static** **final** **long** serialVersionUID = 1L;

**private** String code; // case\_management, member\_management ....,

}

Client user role

**public** **class** ClientUserRole **extends** Role {

**private** **static** **final** **long** serialVersionUID = 1L;

**private** String clientId;

**private** **boolean** allowInheritance;

**private** String status;

**private** RoleType roleType;

**private** String roleCategory;

**private** String displayName;

**public** **enum** Status {

ACTIVE,

INACTIVE

}

**public** **enum** RoleType {

STANDARD,

CUSTOMIZED

}

User:

@Domain(value="cliententity", includeListeners={ListenerType.persistence})

@Repo(Database.rep\_mongodb)

@Getter @Setter @ToString(callSuper=**true**)

**public** **class** ClientEntity **extends** AbstractEntity.IdString {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** **enum** Type {

CLIENT,

ORG;

}

**public** **enum** Status {

INACTIVE,

ACTIVE;

}

@NotNull

//@Model.Param.Values(url="Anthem/icr/p/staticCodeValue/\_search?fn=lookup&where=staticCodeValue.paramCode.eq('/orgType')")

**private** Type type;

**private** String code;

@NotNull

**private** String name;

@NotNull

//@Model.Param.Values(url="Anthem/icr/p/staticCodeValue/\_search?fn=lookup&where=staticCodeValue.paramCode.eq('/orgStatus')")

**private** Status status;

**private** String description;

**private** LocalDate effectiveDate;

**private** LocalDate terminationDate;

//@Ignore private ClientEntity parentEntity;

**private** String parentorganizationId;

//@Ignore private Set<ClientAccessEntity> selectedAccesses;

//private Address.IdString address;

//@Ignore private Set<ClientUserRole> associatedRoles;

/\*\*

\*

\* **@param** cae

\*/

// public void addSelectedAccess(ClientAccessEntity cae) {

// if(getSelectedAccesses() == null) {

// setSelectedAccesses(new HashSet<>());

// }

// getSelectedAccesses().add(cae);

// }

//

// /\*\*

// \*

// \*/

// public void addClientUserRole(ClientUserRole cr){

// if(getAssociatedRoles() == null){

// setAssociatedRoles(new HashSet<>());

// }

// getAssociatedRoles().add(cr);

// }

}

----

@Domain(value="defaultAccessEntity", includeListeners={ListenerType.persistence})

@Repo(value=Database.rep\_mongodb, cache=Cache.rep\_device)

@Getter @Setter @ToString(callSuper=**true**)

**public** **class** DefaultAccessEntity **extends** AbstractEntity.IdString **implements** AccessEntity {

**private** **static** **final** **long** serialVersionUID = 1L;

**private** String type;

**private** String code;

**private** String name;

**private** String domainUri;

**private** Set<DefaultAccessEntity> availableAccesses;

**private** Set<Permission> availablePermissions;

**public** DefaultAccessEntity() { }

**public** DefaultAccessEntity(String type) {

setType(type);

}

@Override

**public** Set<Permission> getPermissions() {

**return** getAvailablePermissions();

}

/\*\*

\*

\* **@param** nested

\*/

**public** **void** addNestedAccess(DefaultAccessEntity nested) {

**if**(getAvailableAccesses() == **null**) {

setAvailableAccesses(**new** HashSet<>());

}

getAvailableAccesses().add(nested);

}

/\*\*

\*

\* **@param** p

\*/

**public** **void** addAvailablePermission(Permission p) {

**if**(getAvailablePermissions() == **null**) {

setAvailablePermissions(**new** HashSet<>());

}

getAvailablePermissions().add(p);

}

**public** **static** **class** Platform **extends** DefaultAccessEntity {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** Platform() {

setType("PLATFORM");

}

}

**public** **static** **class** Application **extends** DefaultAccessEntity {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** Application() {

setType("APPLICATION");

}

}

**public** **static** **class** Module **extends** DefaultAccessEntity {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** Module() {

setType("MODULE");

}

}

**public** **static** **class** Feature **extends** DefaultAccessEntity {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** Feature() {

setType("FEATURE");

}

}

}

--

**public** **interface** AccessEntity {

/\*\*

\*

\* **@return**

\*/

**public** Set<Permission> getPermissions();

}

--

@Getter @Setter @ToString

**public** **class** Permission **extends** IdString {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** **static** **final** **class** Access **extends** Permission {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** **static** **final** String CODE = "ACCESS";

**public** Access() {

**super**("ACCESS");

}

}

**public** **static** **final** **class** Create **extends** Permission {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** **static** **final** String CODE = "CREATE";

**public** Create() {

**super**("CREATE");

}

}

**public** **static** **final** **class** Read **extends** Permission {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** **static** **final** String CODE = "READ";

**public** Read() {

**super**("READ");

}

}

**public** **static** **final** **class** Update **extends** Permission {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** **static** **final** String CODE = "UPDATE";

**public** Update() {

**super**("UPDATE");

}

}

**public** **static** **final** **class** Delete **extends** Permission {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** **static** **final** String CODE = "DELETE";

**public** Delete() {

**super**("DELETE");

}

}

**public** Permission() {}

**public** Permission(String code) {

**this**.code = code;

}

**private** String code;

/\*\*

\*

\*/

@Override

**public** **boolean** equals(Object obj) {

**if**(obj == **null** || !(obj **instanceof** Permission)) **return** **false**;

Permission other = (Permission) obj;

**if**(**this** == other) **return** **true**;

**if**(StringUtils.equals(getCode(), other.getCode())) **return** **true**;

**return** **false**;

}

}