dx-magnolia-ocm

dx-magnolia-ocm is a library which integrates Jackrabbit OCM with Magnolia CMS.
It supports all Jackrabbit OCM features, adapting them to Magnolia, and adds its own.

dx-magnolia-ocm doesn't depend on Magnolia, allowing to read and write Magnolia Jackrabbit storage from external applications.

Contents

Advantages over Jackrabbit OCM

- More polished API
- Generic type safety
- Magnolia content and node data collections are supported
- Magnolia mgnl:resource type is supported
- 1 to 1 mapping of OCM beans and JCR types is not required
- Enums are fully supported, enum type is detected by field type

Getting started

Download library:
dx-magnolia-ocm-1.0.1.jar
dx-magnolia-ocm-1.0.1-sources.jar

Install it into your Maven repository:

```
mvn install:install-file -DgroupId=com.devexperts.web -DartifactId=dx-magnolia-ocm -Dversion=1.0.1 -
Dpackaging=jar -Dfile=dx-magnolia-ocm-1.0.1.jar -DgeneratePom=true
mvn install:install-file -DgroupId=com.devexperts.web -DartifactId=dx-magnolia-ocm -Dversion=1.0.1 -
Dpackaging=jar -Dclassifier=sources -Dfile=dx-magnolia-ocm-1.0.1-sources.jar -DgeneratePom=false
```

You may also put this library into external repository instead of local one.

Note: the library is going be published to publicitly available Maven repository soon, making these steps reduntant. Stay tuned.

Add the following dependency to your Maven project:

```
<dependency>
    <groupId>com.devexperts.web</groupId>
    <artifactId>dx-magnolia-ocm</artifactId>
    <version>1.0.1</version>
</dependency>
```

Specify packages where OCM bean classes are searched. Add the following JVM parameter:

```
-Dcom.devexperts.ocm.packages=<comma separated list of packages>
```

Now you can use this library in your code.

Creating OCM beans

All bean classes must inherit com.devexperts.ocm.BaseOCM class.
Also they must be placed in packages specified by com.devexperts.web.packages system property.

dx-magnolia-ocm uses the same rules as Jackrabbit OCM does. See

- http://jackrabbit.apache.org/5-with-jackrabbit-ocm.html
- http://jackrabbit.apache.org/how-to-map-associations-between-objects.html

Example bean class:

```
@Node(jcrType = BaseOCM.MGNL_CONTENT_NODE_TYPE)
public class FinancePlan extends BaseOCM {
    public enum PricingModel {
        FREE, PAID
    }

    @Field
    private String name;
    @Field
    private PricingModel pricingModel;
    @Field(jcrDefaultValue = "false")
    private boolean active;
    @Bean
    private Money price;

    public FinancePlan() {}

    public PricingModel getPricingModel() {
        return pricingModel;
    }

    public void setPricingModel(PricingModel pricingModel) {
        this.pricingModel = pricingModel;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    public boolean getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Money getPrice() {
        return price;
    }

    public void setPrice(Money price) {
        this.price = price;
    }

}
```

Note the following:

- The class is inherited from BaseOCM class which contains path and uuid fields.
- The class has default constructor.
- JCR type is explicitly specified. By default, nt:unstructured type is used, so the type should be specified in most cases.
- Each field and bean has its own getter and setter.
- You may specify default values for fields.
- Enum is supported as field type.

Bean operations

Creating OCM object

To perform operations with OCM beans, you need com.devexperts.ocm.OCM object.
To create OCM object you need javax.jcr.Session.

In Magnolia, it can be constructed in the following way:

```
Session session = MgnlContext.getHierarchyManager("data").getWorkspace().getSession();
OCM ocm = new OCM(session);
```

Specify the workspace you need when calling getHierarchyManager method.

When you have an instance of OCM object you may perform operations on bean objects.

Inserting new object

```
FinancePlan plan = new FinancePlan();
plan.setName("My plan");
plan.setMoney(new Money("10.00$"));
plan.setPricingModel(FinancePlan.PricingModel.PAID);
plan.setPath("/plans/myplan"); // Path in Jackrabbit repository
ocm.insert(plan);
ocm.save();
```

Update an existing object

```
FinancePlan plan = ocm.getObject(FinancePlan.class, "/plans/myplan");
plan.setActive(true);
ocm.update(plan);
ocm.save();
```

Locking object for an update

```
String objectPath = "/plans/myplan";
ocm.lock(objectPath, false, 5000);
try {
        FinancePlan plan = ocm.getObject(FinancePlan.class, objectPath);
        plan.setActive(true);
        ocm.update(plan);
        ocm.save();
} finally {
        ocm.discardPendingChangesAndUnlock(objectPath);
}
```

Querying objects

```
List<FinancePlan> activePlans = ocm.filter(FinancePlan.class)
        .setScopeFolder("/plans")
        .addEqualTo("active", true).getObjects();
```

Working with Magnolia data types

dx-magnolia-ocm has a number of bean and collection converters which can read and write data types used in Magnolia.

Node data collection

To read and write node data collection, add the following attribute to the field.

```
@Collection(elementClassName = String.class, collectionConverter = NodeDataCollectionConverter.class)
private List<String> videos;
```

Specify collection class element in elementClassName argument and NodeDataCollectionConverter.class in collectionConverter argument.

Collections of java.util.List and java.util.Map types are supported.

Bean objects collection

To read and write collections of bean objects, add the following attribute to the field:

```
@Collection(collectionConverter = DxCollectionConverterImpl.class, elementClassName = Money.class)
private List<Money> prices;
```

Specify collection class element in elementClassName argument and DxCollectionConverterImpl.class in collectionConverter argument.

Collections of java.util.List and java.util.Map types are supported.

mgnl:resource

Magnolia saves files as a mgnl:resource objects. It is widely used in 'website', 'dms' and 'data' workspaces.
dx-magnolia-ocm comes with FileInfo class represents mgnl:resource Jackrabbit object.

To read and write mgnl:resource objects, specify the following attribute:

```
@Bean(converter = FileInfoConverter.class)
private FileInfo icon;
```

Or, if you need to read and write collection of mgnl:resource objects, specify the following attribute:

```
@Collection(collectionConverter = FileInfoCollectionConverter.class)
private List<FileInfo> images;
```

License

Questions

We really appreciate your feedback, feature requests, bug reports or any kind of comments.
Contact us at avasiliev@devexperts.com