Architecture and Design

Table of Contents

This guide describes the internal architecture and design of the framework.

Chapter 1. Maven modules

```
adocs/
 ├──── documentation/
 └──── template/
core/               # see 'core', below
example/            # see 'archetypes', below
scripts/
```

1.1. core

The core modules ....

All of these have the same Maven `groupId`, namely `org.apache.isis.core`.

```
core/
 ├──── .m2/                              # used in gitlab CI
 ├──── applib/                           # isis-core-applib
 ├──── commons/                          # isis-core-commons
 ├──── detached-tests/                   # isis-core-detached-tests
 ├──── integtestsupport/                 # isis-core-integtestsupport
 ├──── legacy/                           # legacy, see below
 ├──── log4j/                            # isis-core-log4j
 ├──── maven-plugin/                     # see 'maven plugins', below
 ├──── mavendeps/                        # see 'mavendeps', below
 ├──── plugins/                          # see 'plugins', below
 ├──── runtime/                          # isis-core-runtime
 ├──── schema/                           # isis-core-schema
 ├──── security/                         # isis-core-security
 ├──── specsupport/                      # isis-core-specsupport
 ├──── unittestsupport/                  # isis-core-unittestsupport
 ├──── viewer-restfulobjects-applib/     # isis-core-viewer-restfulobjects-applib
 ├──── viewer-restfulobjects-rendering/  # isis-core-viewer-restfulobjects-rendering
 ├──── viewer-restfulobjects-server/     # isis-core-viewer-restfulobjects-server
 ├──── viewer-wicket-impl/               # isis-core-viewer-wicket-impl
 ├──── viewer-wicket-model/              # isis-core-viewer-wicket-model
 ├──── viewer-wicket-ui/                 # isis-core-viewer-wicket-ui
 ├──── webdocker/                        # isis-webdocker
 └──── webserver/                        # isis-core-webserver
```

Table 1. core maven modules

Module	Description
`isis-core-applib`	Core application library.

Module	Description
`isis-core-commons`	Framework internal API, not 'public' API and not meant to be used with domain code. The internal API is subject to change without notice.
`isis-core-detached-tests`	To break cyclic dependencies some (JUnit) tests needed to be detached from their originating module and moved here.
`isis-core-integtestsupport`	Integration test support. Application integration tests typically extend from adapter superclasses defined in this module.
`isis-core-log4j`	Configures Log4j as the logging framework
`isis-core-metamodel`	The classes that make up the metamodel which is used to render the UI. See the section below which also includes a simplified UML diagram of these classes.
`isis-core-runtime`	The classes that make up runtime management and persistence of domain objects, as well as framework for security (concepts of authentication or authorisation). Also provides an implementation of the `WrapperFactory` domain service.
`isis-core-schema`	Defines XSDs and generated classes that capture commands and interactions in XML form.
`isis-core-security`	Defines a "bypass" implementation of security, for prototyping only. Using this implementation, any user/password is accepted and
`isis-core-specsupport`	Application BDD specs typically inherit from classes defined in this module.
`isis-core-unittestsupport`	Application unit tests may use some of the utilities defined in this module.
`isis-core-viewer-restfulobjects-applib`	Defines a client-side Java library for interacting with the REST API exposed by the Restful Objects viewer.

Module	Description
`isis-core-viewer-restfulobjects-rendering`	Provides a `RepresentationService` API and a lower-level `ContentNegotiationService` API, along with implementations of each. These implementations provide support for the representations defined by Restful Objects spec v1.0, as well as a number of other Apache Isis-specific representations.
`isis-core-viewer-restfulobjects-server`	Defines the JAX-RS resources supported by the Restful Objects viewer. These parse the input, delegate to the runtime for a response, and hand control to the rendering module to generate a representation.
`isis-core-viewer-wicket-impl`	The top-level integration with Wicket, for example defining the Apache Isis-specific implementations/subclasses of the Wicket APIs for application, web session, localizer and request cycle. Also defines registries of pages and components, as well as a number of domain services and mixins (for use by applications) that are only available within the Wicket viewer.
`isis-core-viewer-wicket-model`	Serializable mementos representing the state of runtime domain objects (or their individual members).
`isis-core-viewer-wicket-ui`	UI components that render the moduls.
`isis-webdocker`	Creates a Docker image for Tomcat that also contains the Apache Isis libraries, thereby enabling "skinny war" support.
`isis-core-webserver`	For development within an IDE, provides a utility class to bootstrap the application (using Jetty).

1.2. core/plugins

The `core/plugins` modules …

All of these have the same Maven `groupId`, namely `org.apache.isis.core`.

Plugin Modules

```
core/
 └──── plugins/
     ├──── codegen-bytebuddy/      # isis-core-plugins-codegen-bytebuddy
     ├──── codegen-javassist/      # isis-core-plugins-codegen-javassist
     ├──── discovery-reflections/  # isis-core-plugins-discovery-reflections
     ├──── eventbus-axon/          # isis-core-plugins-eventbus-axon
     ├──── eventbus-guava/         # isis-core-plugins-eventbus-guava
     ├──── jaxrs-resteasy-3/       # isis-core-plugins-jaxrs-resteasy-3
     ├──── jaxrs-resteasy-4/       # isis-core-plugins-jaxrs-resteasy-4
     ├──── jdo-datanucleus-4/      # isis-core-plugins-jdo-datanucleus-4
     ├──── jdo-datanucleus-5/      # isis-core-plugins-jdo-datanucleus-5
     └──── security-shiro/         # isis-core-plugins-security-shiro
```

Table 2. core/mavendeps maven modules

Module	Description
isis-core-plugins-codegen-bytebuddy	Framework support for Java byte code generation at runtime utilizing ByteBuddy.
isis-core-plugins-codegen-javassist	Framework support for Java byte code generation at runtime utilizing Javassist.
isis-core-plugins-discovery-reflections	Framework support for Java class hierarchy discovery utilizing reflections.org.
isis-core-plugins-eventbus-axon	Integrates axon-framework's eventbus.
isis-core-plugins-eventbus-guava	Integrates guava's eventbus.
isis-core-plugins-jaxrs-resteasy-3	Framework support for RESTful viewer utilizing JBoss RestEasy version 3.x. (JEE 7 compliant)
isis-core-plugins-jaxrs-resteasy-4	Framework support for RESTful viewer utilizing JBoss RestEasy version 4.x. (JEE 8 compliant)
isis-core-plugins-jdo-datanucleus-4	Framework support for JDO utilizing DataNucleus 4.x.
isis-core-plugins-jdo-datanucleus-5	Framework support for JDO utilizing DataNucleus 5.x.
isis-core-plugins-security-shiro	Defines an implementation of security authentication which delegates to Apache Shiro.

1.3. core/mavendeps

The `core/mavendeps` modules ...

All of these have the same Maven `groupId`, namely `org.apache.isis.mavendeps`.

mavendeps Modules

```
core
 └──── mavendeps/
     ├──── isis-mavendeps-intellij/    # isis-mavendeps-intellij
     ├──── isis-mavendeps-testing/     # isis-mavendeps-testing
     └──── isis-mavendeps-webapp/      # isis-mavendeps-webapp
```

Table 3. core/mavendeps maven modules

Module	Description
`isis-mavendeps-intellij`	Defunct.

`isis-mavendeps-testing`	Aggregates dependencies on various test-scope plugins useful for unit- and integration testing a module. These include Apache Isis' own `unittestsupport`, `integtestsupport` and `specsupport` modules, as well as a number of common testing/mocking/assertion libraries.
	
	These can then be included using a single dependency declaration:
	
	`<dependencies>`
	`  <dependency>`
	`  <groupId>org.apache.isis.mavendeps</groupId>`
	`  <artifactId>isis-mavendeps-testing</artifactId>`
	`  <type>pom</type>`
	`  <scope>test</scope>`
	`  </dependency>`
	`</dependencies>`

Module	Description

isis-mavendeps-webapp	Aggregates dependencies on Apache Isis runtime itself when used within a webapp.
	
	These can then be included using a single dependency declaration:
	
	`<dependencies>`
	`  <dependency>`
	`  <groupId>org.apache.isis.mavendeps</groupId>`
	`  <artifactId>isis-mavendeps-webapp</artifactId>`
	`  <type>pom</type>`
	`  </dependency>`
	`</dependencies>`

1.4. core/legacy

The core/legacy modules ...

All of these have the same Maven groupId, namely org.apache.isis.core.

Legacy Modules

```
core/
 └──── legacy/
     └──── transition-1-2/             # isis-core-transition-1-2
```

Module	Description
isis-core-transition-1-2	Provides a compatibility layer for applications that migrate from Apache Isis 1.x to 2.x. API (Some of the Java interfaces and built-in domain-services that got removed with version 2 can be found here.)

1.5. core/mavenplugins

There is a single Maven plugin module. Its Maven groupId is org.apache.isis.tools.

Plugin Modules

```
core/
 └──── maven-plugin/               # isis-maven-plugin
```

Module	Description
isis-maven-plugin	Code to build a maven plugin for the build. This plugin can validate the metamodel and generate Swagger specs for a domain model as part of the application's build pipeline.

1.6. archetypes

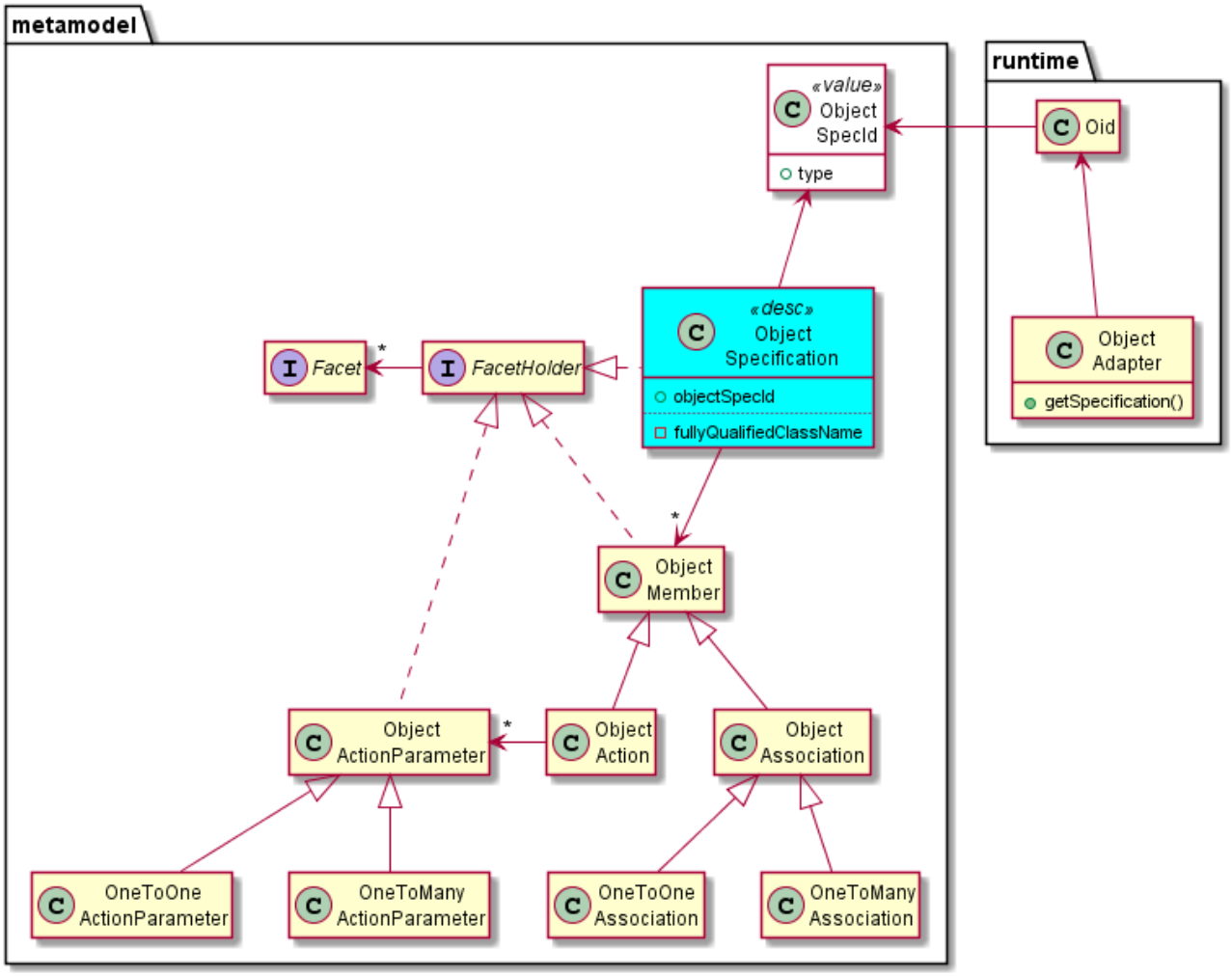
```
example/
 ├──── application/
 │    ├──── helloworld/     # org.apache.isis.example.application:helloworld
 │    └──── simpleapp/      # org.apache.isis.example.application:simpleapp
 │         ├──── application/   # org.apache.isis.example.application:simpleapp-
application
 │         ├──── module-simple/ # org.apache.isis.example.application:simpleapp-module-
simple
 │         └──── webapp/        # org.apache.isis.example.application:simpleapp-webapp
 └──── archetype/
     ├──── helloworld/     # org.apache.isis.archetype:helloworld-archetype
     └──── simpleapp/      # org.apache.isis.archetype:simpleapp-archetype
```

Module	Description
helloworld	An example application as a single Maven module, including domain classes themselves plus code to bootstrap Apache Isis. This is reverse engineered into the "helloworld" archetype.
simpleapp	The top-level aggregator module for the "simpleapp" example application. This is an extended version of helloworld, providing more structure (separating out domain model into modules) as well as unit tests, integration tests, BDD specs and fixtures. The simpleapp modules in aggregate are reverse engineered into the "simpleapp" archetype.
simpleapp-application	Defines the contents of the "simpleapp" application using Apache Isis-defined classes, as well as globally scoped domain services and the home page.
simpleapp-module-simple	Contains the domain model for a single module. The intention is to allow this module structure to be copied so that the developer can easily create further modules as their app increases in size.

Module	Description
`simpleapp-webapp`	Bootstraps Apache Isis as a webapp.
`helloworld-archetype`	Helloworld archetype, reverse engineered from the "helloworld" application (above).
`simpleapp-archetype`	Simpleapp archetype, reverse engineered from the "simpleapp" application (above).

Chapter 2. Metamodel

The diagram below shows a simplified version of Apache Isis' internal metamodel.



where in the `metamodel` package:

`ObjectSpecification`
: is equivalent to `java.lang.Class`

`ObjectSpecId`
: is a value object equivalent to the `@DomainObject#objectType` or `@DomainService#objectType` attribute

`OneToOneAssociation`
: represents a scalar property

`OneToManyAssociation`
: represents a collection

`ObjectAction`
: represents an action (with multiple parameters, either scalar or list)

and in the `runtime` package:

`Oid`
: is equivalent to the applib `Bookmark`

  and appears in URLs in the Wicket and Restful Objects viewers

`ObjectAdapter`
: is equivalent to `java.lang.Object`