Annotations (org.apache.isis.applib.annotation)

@DomainService *type*	Indicates a class is a domain service singleton that contributes *Actions* to the application's menu and/or to the REST backend.
@DomainServiceLayout *type*	Domain service layout customization. When present, overridden by menubars.layout.xml
@HomePage *type*	Indicates which view-model should be used to render as homepage.
@DomainObject *type*	Indicates a class that has an *identity* and is *bookmarkable*, it is either a **view-model** or an **entity**. View-models may alternatively implement interface ViewModel and optionally java.io.Serializable.
@DomainObjectLayout *type*	Domain object layout customization. When present, overriden by Xxx.layout[.layoutName].xml > Xxx.layout.xml > Xxx.layout.fallback.xml.

@Action — *method, mixin*

Indicates that a method contributes an *Action*.

Action support methods

```
@Action .. placeOrder(X x, Y y, Z z)
boolean hidePlaceOrder()
String[1] disablePlaceOrder()
boolean hide{0|1|2}PlaceOrder(X|Y|Z);
String[1] disable{0|1|2}PlaceOrder(X|Y|Z)
String[1] validate{0|1|2}PlaceOrder(X|Y|Z)
String[1] validatePlaceOrder(X x, Y y, Z z)
Collection<X|Y|Z> autoComplete{0|1|2}PlaceOrder(X|Y|Z,
                  @MinLength(3) String search) 2 args
Collection<X|Y|Z> choices{0|1|2}PlaceOrder(X|Y|Z)
X|Y|Z default{0|1|2}PlaceOrder()
String[1] namedPlaceOrder()
String[1] describedPlaceOrder()
```

@ActionLayout *method, mixin*	*Action* layout customization. When present, overridden by Xxx.layout[.layoutName].xml > Xxx.layout.xml > Xxx.layout.fallback.xml.
@Title *field, method*	Indicates which property or properties make up the object title.

@Property — *field, getter, mixin*

Typically used with @lombok.Getter @lombok.Setter

Indicates that a field or method contributes a *Property*.

Property support methods

```
@Property .. Email email, getEmail(), setEmail();
boolean hideEmail();
String[1] disableEmail();
String[1] validateEmail(Email);
Collection<..> autoCompleteEmail(),
Collection<..> choicesEmail();
String[1] namedEmail()
String[1] describedEmail()
```

@PropertyLayout *field, getter, mixin*	*Property* layout customization. When present, overriden by Xxx.layout[.layoutName].xml > Xxx.layout.xml > Xxx.layout.fallback.xml.

@Collection — *field, getter, mixin*

Typically used with @lombok.Getter @lombok.Setter

Indicates that a field or method contributes a *Collection*, meaning a non-scalar property.

Collection support methods

```
@Collection .. List<Order> orders, getOrders()
boolean hideOrders()
String[1] disableOrders()
String[1] namedOrders()
String[1] describedOrders()
```

@CollectionLayout *field, getter, mixin*	*Collection* layout customization. Overriden by Xxx.layout[.layoutName].xml > Xxx.layout.xml > Xxx.layout.fallback.xml.
@Parameter @MinLength *parameter*	Action parameter constraint and behavior customization.
@ParameterLayout *parameter*	Action parameter layout customization.
@MemberSupport *method*	Indicates that a method supports either an *Action* or a *Property* or a *Collection*.

... annotations continued

@Domain.Include *field, method*	Indicates that a field or method must contribute to the metamodel.
@Domain.Exclude @Programmatic *field, method, type*	Indicates that a field, method or type must **not** contribute to the metamodel.

Services (most common)

RepositoryService	access to persistence layer
MessageService	UI notifications
FactoryService	object construction
ClockService	provides virtual clock
WrapperFactory	enforce domain rules on domain objects
EventBusService	emit custom events
BookmarkService	bookmark = object identity
InteractionService	*Action* execution and *Property* modification
TransactionService	request transactions
MetaModelService	export domain model

Object Methods

boolean hidden()	hide all members
String[1] disabled()	disable all members

Layout Support

String[1] title()	imperative title literal
String iconName()	icon file suffix (UI)
String cssClass()	additional CSS class (UI)
String layout()	layout file suffix (grid layout)

Lifecycle Callback

void created()	emitted by FactoryService
void loaded() void saved() void saving() void persisted() void persisting() void deleting() void removing() void updated() void updating()	emitted by persistence layer integration (JDO/JPA)

Translation

[1] All member-support methods (and some object methods) that return String can optionally return TranslatableString.

Value Types

Number Types	byte, Byte, short, Short, int, Integer, long, Long, float, Float, double, Double, BigInteger, BigDecimal
Boolean Types	boolean, Boolean
Text Types	char, Character, String, Password
Markup Types	Markup, AsciiDoc, Markdown
Temportal Types	java.util.Date, java.sql.{Date|Timestamp} java.time.{LocalDate|LocalDateTime} Joda Time (deprecated) *time-of-day types yet not supported by UI, hence not listed*
Enum Types	any
Collection Types	Collection<T>, List<T>, Set<T>, Can<T>, T[]
Additional Types	BufferedImage, Blob, Clob, UUID, Url, LocalResourcePath, SSE (ServerSentEvents)

Error & Exception Handling

RecoverableException	anticipated error that results in an UI notification popup
NonRecoverableExeption	error that results in an error page showing the stacktrace

Spring Annotations and Standard API

@SpringBootApplication @SpringBootTest @Configuration @Import @ComponentScan @EntityScan	Bootstrapping
@Component @Service @Repository	Indicates that a class is a "component". Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.
@EventListener	Marks a method as a listener for application events. **Domain Events** .. subscribe using Spring's @EventListener @DomainObject(xxxLifecycleEvent=….) @DomainObjectLayout(xxxUiEvent=….) @Action(domainEvent=….) @Property(domainEvent=…) @Collection(domainEvent=…)
@Nullable (org.springframework.lang)	declares that annotated elements can be **null**
@javax.inject.Inject @Autowired	Marks a constructor, field, setter method, or config method as to be autowired by Spring's dependency injection facilities.
@PostConstruct	Marks a method to be executed after dependency injection is done to perform any initialization.
@PreDestroy	Marks a method as a listener for notification that the instance is in the process of being removed by the container .

New Programming Style: Parameters as Typed Tuple [2.0.0-M6]

Action parameters can now be collected into an immutable value type say Parameters, a typed tuple. The name is arbitrary. *Future releases might support Java records.* Instances of the Parameters type are passed to the various action-support methods, which need to be single-arg, except for autoComplete, which is required to be bi-arg.

For regular objects, action-support methods must reference parameters by index (0, 1, 2, …). However, with *Mixins* it is allowed to reference parameters by name.

Example: *Action Mixin*, with nested class Parameters, using parameter references by name.

```java
@Action
@RequiredArgsConstructor
public class Customer_placeOrder {

    private final Customer target;

     // typed tuple made of all the action parameters
    @Value @Accessors(fluent = true)
    public static class Parameters {
        Product product;
        int quantity;
    }

    public Customer act(
        @Parameter Product product,
        @Parameter int quantity) {
        // …
        return target;
    }

    // support methods (no action name reference required)
    public boolean hide() { … }
    public String disable() { … }
    public String validate(Parameters params) { … }

    // parameter support methods (exemplified on first parameter)
    public boolean hideProduct(Parameters params) { … }
    public String disableProduct(Parameters params) { … }
    public String validateProduct(Parameters params) { … }
    public Collection<Product> choicesProduct(Parameters params){ … }
    // note: additional search parameter required: search
    public Collection<Product> autoCompleteProduct(
        Parameters params, @MinLength(3) String search) { … }
    public Product defaultProduct(Parameters params) { … }

    // parameter supporting methods (exemplified on second parameter)
    public boolean hideQuantity(Parameters params) { … }
    // …
}
```

Entity and Viewmodel Annotations

JPA Entities (javax.persistence)

@Entity @Table @NamedQueries @Id @Version @Column @Transient

JDO Entities (javax.jdo.annotation)

@PersistenceCapable @DatastoreIdentity @Inheritance @Discriminator @Version @Queries @Uniques @Indices @NotPersistent @Column @PrimaryKey @Join @Element

JAXB View Models (javax.xml.bind.annotation)

@XmRootElement; @XmlType
@XmlAccessorType; @XmlTransient
referenced entities: @XmlJavaTypeAdapter(PersistentEntityAdapter.class)