Annotations (org.apache.causeway.applib.annotations)			
@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.	@Title *field, method*	Indicates which *Property* or *Properties* make up the object title.
@DomainObject *type*	Indicates a class that has an *identity* and is *bookmarkable*, it is either a ***view-model*** or an ***entity***. Not allowed on *interfaces*. 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. Example: @Action .. placeOrder(X x, Y y, Z z)		
Action support methods	String[1] namedPlaceOrder() String[1] describedPlaceOrder() boolean hidePlaceOrder() String[1] disablePlaceOrder() String[1] validatePlaceOrder(X x, Y y, Z z)		
Action Parameter support methods	boolean hide{0|1|2}PlaceOrder(...) 0..3 args String[1] disable{0|1|2}PlaceOrder(...) 0..3 args X|Y|Z default{0|1|2}PlaceOrder(...) 0..3 args Collection<X|Y|Z> autoComplete{0|1|2}PlaceOrder(..., 0..3 args @MinLength(3) String search) +1 arg Collection<X|Y|Z> choices{0|1|2}PlaceOrder(...) 0..3 args String[1] validate{0|1|2}PlaceOrder(...) 0..3 args		
@ActionLayout *method, mixin*	*Action* layout customization. When present, overriden by Xxx.layout[.layoutName].xml > Xxx.layout.xml > Xxx.layout.fallback.xml.		
@Property *field, getter, mixin*	Indicates that a field or method contributes a *Property*. If annotated with @Title, then used for (part of) the title of the object. Typically also used with @lombok.Getter @lombok.Setter Example: @Property .. Email email, getEmail(), setEmail();		
Property support methods	String[1] namedEmail() String[1] describedEmail() boolean hideEmail() String[1] disableEmail() Email defaultEmail() Collection<Email> choicesEmail() Collection<Email> autoCompleteEmail(@MinLength(3) String search) String[1] validateEmail(Email)		
@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*	Indicates that a field or method contributes a *Collection*, meaning a non-scalar property. Typically used with @lombok.Getter @lombok.Setter Example: @Collection .. List<Order> orders, getOrders()		
Collection support methods	String[1] namedOrders() String[1] describedOrders() boolean hideOrders() String[1] disableOrders()		
@CollectionLayout *field, getter, mixin*	*Collection* layout customization. When present, overriden by Xxx.layout[.layoutName].xml > Xxx.layout.xml > Xxx.layout.fallback.xml.		
@Parameter *parameter*	Action parameter constraint and behavior customization.	@MinLength *parameter*	Search parameter's minimum required character count.
@ParameterLayout *parameter*	Action parameter layout customization.		
@ObjectSupport *method*	Indicates that a method supports its *Object*. Not allowed on *Mixins*.	@ObjectLifecycle *method*	Object lifecycle callback method. (no-arg, void)
@MemberSupport *method*	Indicates that a method supports an *Action*, *Property* or *Collection*. Usable with *Objects* and *Mixins*. Also allowed on the *Mixin's* main method.		

... 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.
@Value, @ValueSemantics ... see *ValueTypes*	

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

Object Support used with @ObjectSupport	
String[1] title()	imperative title literal
String iconName()	icon file suffix (UI)
FontAwesomeLayers iconFaLayers()	Font Awesome icon(s) **experimental [2.0.0-RC4]**
String cssClass()	additional CSS class (UI)
String layout()	layout file suffix (grid layout)
boolean hidden()	hide all members
String[1] disabled()	disable all members
Lifecycle Callback used with @ObjectLifecycle	
void created()	emitted by FactoryService
void loaded() void persisting() void persisted() void updating() void updated() void removing()	emitted by persistence layer integration (JDO/JPA) supported synonyms: saving() = persisting() saved() = persisted() deleting() = removing()

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
Temporal Types	java.util.Date, java.sql.{Date|Time|Timestamp}, java.time.*, Joda Time (via extension module)
Enum Types	any
Collection Types	Collection<T>, List<T>, Set<T>, Can<T>, T[]
Additional Types	BufferedImage, Blob, Clob, UUID, Url, LocalResourcePath, SSE (ServerSentEvents)
Annotations (org.apache.causeway.applib.annotations)	
@Value *type*	Indicates a concrete class that has immutable state and no *identity*. A *value-type*.

@ValueSemantics *field, getter, mixin, parameter* *since [2.0.0-M7]*	`class Order {`
	
	`  // custom ValueSemanticsProvider,`
	`  // selected by bean qualifier 'special-asciidoc'`
	`  @ValueSemantics(`
	`  provider = "special-asciidoc")`
	`  public AsciiDoc getDescription() { ... }`
	
	`  // custom localized temporal format modifiers`
	`  @ValueSemantics(`
	`  dateFormatStyle = FormatStyle.FULL,`
	`  timeFormatStyle = FormatStyle.SHORT,`
	`  timePrecision = TimePrecision.MINUTE)`
	`  public LocalDateTime getDateTime() { ... }`
	
	`  // custom decimal digit constraints`
	`  @ValueSemantics(`
	`  maxTotalDigits = 19,`
	`  maxFractionalDigits = 5)`
	`  public BigDecimal getCost() { ... }`
	
	`}`

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.

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)

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.

```
@Action
@RequiredArgsConstructor
public class Customer_placeOrder {

   private final Customer target;

    // typed tuple made of all the action parameters
   @lombok.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) { ... }
   // ...
}
```