## Annotations (org.apache.isis.applib.annotations)

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>@DomainService</code></td>
<td>Indicates a class is a domain service singleton that contributes Actions to the application’s menu and/or to the REST backend.</td>
</tr>
<tr>
<td><code>@DomainServiceLayout</code></td>
<td>Domain service layout customization. When present, overridden by <code>menubars.layout.xml</code>.</td>
</tr>
<tr>
<td><code>@HomePage</code></td>
<td>Indicates which view-model should be used to render as homepage.</td>
</tr>
<tr>
<td><code>@DomainObject</code></td>
<td>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 <code>ViewModel</code> and optionally <code>java.io.Serializable</code>.</td>
</tr>
<tr>
<td><code>@DomainObjectLayout</code></td>
<td>Domain object layout customization. When present, overridden by <code>Xxx.layout[.layoutName].xml &gt; Xxx.layout.xml &gt; Xxx.layout.fallback.xml</code>.</td>
</tr>
<tr>
<td><code>@Action</code></td>
<td>Indicates that a method contributes an Action. Example: <code>@Action .. placeOrder(X y, Y Z)</code></td>
</tr>
<tr>
<td><code>@Property</code></td>
<td>Indicates that a field or method contributes a Property. If annotated with <code>@Title</code>, then used for (part of) the title of the object. Typically also used with <code>@lombok.Getter @lombok.Setter</code> Example: <code>@Property .. Email email();</code></td>
</tr>
<tr>
<td><code>@Collection</code></td>
<td>Indicates which Property or Properties make up the object title.</td>
</tr>
<tr>
<td><code>@CollectionLayout</code></td>
<td>Indicates which view-model should be used to render as collection.</td>
</tr>
<tr>
<td><code>@Domain.Include</code></td>
<td>Indicates that a field or method must contribute to the metamodel.</td>
</tr>
<tr>
<td><code>@Domain.Exclude</code></td>
<td>Indicates that a field, method or type must not contribute to the metamodel.</td>
</tr>
<tr>
<td><code>@LogicalTypeName</code></td>
<td>Assigns a logical type name to an interface.</td>
</tr>
<tr>
<td><code>@Value</code></td>
<td>Value semantics used with <code>@Value</code>.</td>
</tr>
</tbody>
</table>

## Services (most common)

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RepositoryService</td>
<td>Access to persistence layer</td>
</tr>
<tr>
<td>MessageService</td>
<td>UI notifications</td>
</tr>
<tr>
<td>FactoryService</td>
<td>Object construction</td>
</tr>
<tr>
<td>ClockService</td>
<td>Provides virtual clock</td>
</tr>
<tr>
<td>WrapperFactory</td>
<td>Enforce domain rules on domain objects</td>
</tr>
<tr>
<td>EventBusService</td>
<td>Emit custom events</td>
</tr>
<tr>
<td>BookmarkService</td>
<td>Bookmark = object identity</td>
</tr>
<tr>
<td>InteractionService</td>
<td>Action execution and Property modification</td>
</tr>
<tr>
<td>TransactionService</td>
<td>Request transactions</td>
</tr>
<tr>
<td>MetaModelService</td>
<td>Export domain model</td>
</tr>
</tbody>
</table>

## Object Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getAction()</code></td>
<td>Returns the <code>Action</code> for the current object.</td>
</tr>
<tr>
<td><code>getTitle()</code></td>
<td>Returns the title of the object.</td>
</tr>
<tr>
<td><code>getIdentity()</code></td>
<td>Returns the object identity.</td>
</tr>
<tr>
<td><code>getVersion()</code></td>
<td>Returns the version of the object.</td>
</tr>
<tr>
<td><code>getCompositeId()</code></td>
<td>Returns the composite ID of the object.</td>
</tr>
<tr>
<td><code>toJson()</code></td>
<td>Returns the object as a JSON string.</td>
</tr>
<tr>
<td><code>export()</code></td>
<td>Returns the object as a string.</td>
</tr>
<tr>
<td><code>getExport()</code></td>
<td>Returns the export of the object.</td>
</tr>
<tr>
<td><code>importObject()</code></td>
<td>Imports an object.</td>
</tr>
<tr>
<td><code>importExport()</code></td>
<td>Imports an export.</td>
</tr>
</tbody>
</table>

## Lifecycle Callbacks

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>creating()</code></td>
<td>Called before creating an object.</td>
</tr>
<tr>
<td><code>created()</code></td>
<td>Called after creating an object.</td>
</tr>
<tr>
<td><code>updated()</code></td>
<td>Called after updating an object.</td>
</tr>
<tr>
<td><code>updating()</code></td>
<td>Called before updating an object.</td>
</tr>
<tr>
<td><code>removed()</code></td>
<td>Called after removing an object.</td>
</tr>
<tr>
<td><code>removing()</code></td>
<td>Called before removing an object.</td>
</tr>
</tbody>
</table>

## Translation

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Types</td>
<td>byte, Byte, short, Short, int, Long, Long, float, Float, double, Double, BigInteger, BigDecimal</td>
</tr>
<tr>
<td>Boolean Types</td>
<td>boolean, Boolean</td>
</tr>
<tr>
<td>Text Types</td>
<td>char, Character, String, Password</td>
</tr>
<tr>
<td>Markup Types</td>
<td>Markup, AsciiDoc, Markdown</td>
</tr>
<tr>
<td>Temporal Types</td>
<td>java.util.Date, java.sql.{Date</td>
</tr>
<tr>
<td>Enum Types</td>
<td>any</td>
</tr>
<tr>
<td>Collection Types</td>
<td>Collection&lt;T&gt;, List&lt;T&gt;, Set&lt;T&gt;, Can&lt;T&gt;, T[]</td>
</tr>
<tr>
<td>Additional Types</td>
<td>BufferedImage, Blob, Clob, UUID, Url, LocalResourcePath; SSE (ServerSentEvents)</td>
</tr>
</tbody>
</table>

### Annotations (org.apache.isis.applib.annotations)

- **@Value type**
  - Indicates a concrete class that has immutable state and no identity. A value-type.

- **@ValueSemantics**
  - Field, getter, mixin, parameter
    - *class Order {*
    - // custom ValueSemanticsProvider,
      - @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

- **NonRecoverableException**
  - error that results in an error page showing the stacktrace

### Spring Annotations and Standard API

- **@Target**
  - Designates the argument types for the annotated method or constructor.

- **@Retention**
  - Specifies how the annotation is retained through the compilation and linking processes.

- **@Documented**
  - Indicates that the annotation should be documented in the javadoc tool.

- **@InputParameters**
  - Indicates that the annotated method has input parameters.

- **@OutputParameters**
  - Indicates that the annotated method has output parameters.

### Spring Annotations and Standard API

- **@Component**
  - Indicate that a class is a component. Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.

- **@ComponentScan**
  - Marks a method as a listener for application events.

- **@DomainObject**
  - Subscribe using Spring’s @EventListener

- **@DomainObjectLayout**
  - Marks a method as a listener for application events.

- **@Action**
  - 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**
  - Designates a class as an entity.

- **@Table**
  - Specifies the name of the table in the database.

- **@NamedQueries**
  - Specifies named queries for this entity.

- **@Id**
  - Specifies the primary key of the entity.

- **@Version**
  - Specifies the version of the entity.

- **@Column**
  - Specifies the column of the entity.

- **@Transient**
  - Marks a field as not persistent.

#### JDO Entities (javax.jdo.annotation)

- **@PersistenceCapable**
  - Specifies that the class is a persistence-capable class.

- **@DatastoreIdentity**
  - Specifies the datastore identity of the entity.

- **@Inheritance**
  - Specifies the inheritance of the entity.

- **@Discriminator**
  - Specifies the discriminator of the entity.

- **@Version**
  - Specifies the version of the entity.

- **@Queries**
  - Specifies the queries for this entity.

- **@Uniques**
  - Specifies the unique constraints for this entity.

- **@Indices**
  - Specifies the indices for this entity.

- **@NotPersistent**
  - Marks a field as not persistent.

- **@PrimaryKey**
  - Specifies the primary key of the entity.

- **@Join**
  - Specifies a join between entities.

- **@Element**
  - Specifies an element of the entity.

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

- **@XmRootElement**
  - Indicates that the element is a root element in an XML document.

- **@XmlType**
  - Specifies the type of the element.

- **@XmlAccessorType**
  - Specifies the access to the elements of the type.

- **@XmlTransient**
  - Marks a field as not serializable.

- **@XmJavaTypeAdapter**
  - Specifies the adapter for the Java type.

- **@XmlAccessorType**
  - Specifies the accessor type for the Java type.

- **@XmlTransient referenced entities**
  - Specifies the referenced entities for the Java type.
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
    @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) { ... }
    // ...}
```