Contents

1 Package `core`

The core package is the root package for the storydriven core meta-model. It defines several abstract super classes which implement an extension mechanism as well as recurring structural features like, e.g., names of elements. The classes in this package are intended to be sub-classed by any meta-model element.

Namespace `http://www.storydriven.org/core/0.3.1`

1.1 Abstract Class `CommentableElement`

Abstract super class for all meta-model elements that may carry a comment in form of a string.

Subclasses

- `Expression` (see section 2.1 on page 5)

Superclasses

- `ExtendableElement` (see section 1.2 on page 2)

Attributes

`comment : EString` The comment string that can be used to attach arbitrary information to `CommentableElements`.

1.2 Abstract Class `ExtendableElement`

Abstract base class for the whole story diagram model. The `ExtendableElement` specifies the extension mechanism that can be used to extend an object by an `Extension` containing additional attributes and references.

Subclasses

- `CommentableElement` (see section 1.1 on page 2)
- `Extension` (see section 1.3 on page 3)
- `NamedElement` (see section 1.4 on page 4)
- `TypedElement` (see section 1.5 on page 4)

Operations

`getExtension() : Extension` No detailed documentation provided.

`provideExtension() : Extension` No detailed documentation provided.

`getAnnotation(EString) :` No detailed documentation provided.

`provideAnnotation(EString) :` No detailed documentation provided.

Containments

annotation : No detailed documentation provided.

extension : Extension No detailed documentation provided.

1.3 Abstract Class `Extension`

Abstract super class for an `Extension` that can be defined for an object.

Superclasses

- `ExtendableElement` (see section 1.2 on page 2)

References

base : No detailed documentation provided.

modelBase : No detailed documentation provided.

owningAnnotation : No detailed documentation provided.

extendableBase : ExtendableElement No detailed documentation provided.

1.4 Abstract Class `NamedElement`

Abstract super class for all meta-model elements that carry a name.

Superclasses

- `ExtendableElement` (see section 1.2 on page 2)

Attributes

`name : EString` The name attribute of a meta-model element.

1.5 Abstract Class `TypedElement`

Abstract super class for all meta-model elements that are typed by means of an `EClassifier` or an `EGenericType`.

Superclasses

- `ExtendableElement` (see section 1.2 on page 2)

Containments

genericType : No detailed documentation provided.

References

type : No detailed documentation provided.

2 Package `core::expressions`

The base package for all expressions which can be used for modeling activities and patterns.

Namespace `http://www.storydriven.org/core/expressions/0.3.1`

Prefix `expr`

2.1 Abstract Class `Expression`

Represents any expression in an embedded textual language, e.g. OCL or Java. An expression's type is dynamically derived by an external mechanism (see `TypedElement`).

Subclasses

- `TextualExpression` (see section 2.2 on page 5)
- `UnaryExpression` (see section 3.5 on page 8)
- `BinaryExpression` (see section 3.6 on page 8)
- `LiteralExpression` (see section 3.10 on page 10)

Superclasses

- `CommentableElement` (see section 1.1 on page 2)

2.2 Class `TextualExpression`

Represents any expression in a textual language embedded into Story Diagrams, e.g. OCL or Java .

Superclasses

- `Expression` (see section 2.1 on page 5)

Attributes

`expressionText : EString` Holds the expression, e.g. in OCL or Java.

`language : EString` String representation of the used language which has to be unique. Examples are OCL and Java.

`languageVersion : EString` String representation of the used language's version. The format is <Major>.<Minor>[.<Revision>[.<Build>]]. Examples: `1.4` or `3.0.1` or `1.0.2.20101208`.

3 Package `core::expressions::common`

No detailed documentation provided.

Namespace `http://www.storydriven.org/core/expressions/common/0.3.1`

Prefix `sdcec`

3.1 Enumeration `LogicOperator`

Defines the operators for binary logic expressions. The unary logic expression representing negated expressions is reflected by the `NotExpression`.

AND No detailed documentation provided.

OR No detailed documentation provided.

XOR No detailed documentation provided.

IMPLY No detailed documentation provided.

EQUIVALENT No detailed documentation provided.

3.2 Enumeration `ComparingOperator`

Defines the operators for comparing expressions.

LESS No detailed documentation provided.

LESS_OR_EQUAL No detailed documentation provided.

EQUAL No detailed documentation provided.

GREATER_OR_EQUAL No detailed documentation provided.

GREATER No detailed documentation provided.

UNEQUAL No detailed documentation provided.

REGULAR_EXPRESSION For comparison of a String with a regular expression.

3.3 Enumeration `ArithmeticOperator`

Defines the operators for arithmetic expressions.

PLUS No detailed documentation provided.

MINUS No detailed documentation provided.

TIMES No detailed documentation provided.

DIVIDE No detailed documentation provided.

MODULO No detailed documentation provided.

3.4 Enumeration `UnaryOperator`

No detailed documentation provided.

NOT No detailed documentation provided.

PLUS No detailed documentation provided.

MINUS No detailed documentation provided.

INCREMENT No detailed documentation provided.

DECREMENT No detailed documentation provided.

3.5 Class `UnaryExpression`

Represents an unary expression.

Superclasses

- `Expression` (see section 2.1 on page 5)

Attributes

`operator : UnaryOperator` Represents the operator of the expression.

Containments

enclosedExpression : Expression Represents the operand of a `NotExpression`, e.g. `a < 5` in `NOT(a < 5)`.

3.6 Abstract Class `BinaryExpression`

Represents any binary expression like `v < 5` or `x + 7`.

Subclasses

- `ComparisonExpression` (see section 3.7 on page 9)
- `ArithmeticExpression` (see section 3.8 on page 9)
- `LogicalExpression` (see section 3.9 on page 10)

Superclasses

- `Expression` (see section 2.1 on page 5)

Containments

leftExpression : Expression Represents the first operand of a binary expression, e.g. x in the expression `x < 5`.

rightExpression : Expression Represents the second operand of a binary expression, e.g. `5` in the expression `x < 5`.

3.7 Class `ComparisonExpression`

Represents comparing expressions like `a < 5` or `a >= 7`.

Superclasses

- `BinaryExpression` (see section 3.6 on page 8)

Attributes

`operator : ComparingOperator` Specifies the expression's comparing operator, e.g. `<`, `>=`, `!=`.

3.8 Class `ArithmeticExpression`

Represents arithmetic expressions like `a + 5` or `a * 7`.

Superclasses

- `BinaryExpression` (see section 3.6 on page 8)

Attributes

`operator : ArithmeticOperator` Specifies the expression's arithmetic operator, e.g. `+`, `-`, `*`, `/`, or `MODULO`.

3.9 Class `LogicalExpression`

Represents binary, logic expressions like `a AND b` and `a OR b`.

Superclasses

- `BinaryExpression` (see section 3.6 on page 8)

Attributes

`operator : LogicOperator` Specifies the expression's logic operator, e.g. `AND`, `OR`, or `XOR`.

3.10 Class `LiteralExpression`

Represents any literal, i.e. a value whose type is an `EDataType`. Literals are, for example, `5`, `3.14`, `'c'`, `"text"`, `true`.

Superclasses

- `Expression` (see section 2.1 on page 5)

Attributes

`value : EString` String representation of the value, e.g. "5", "3.14", "c", "text", or "true".