1 FEM – Function Evaluation Model

Das «Function Evaluation Model» oder kurz «FEM» beschreibt ein generisches Datenmodell zur Auswertung abstrakter Funktionen im Rahmen gegebener Parameterwerte und Kontextinformationen.

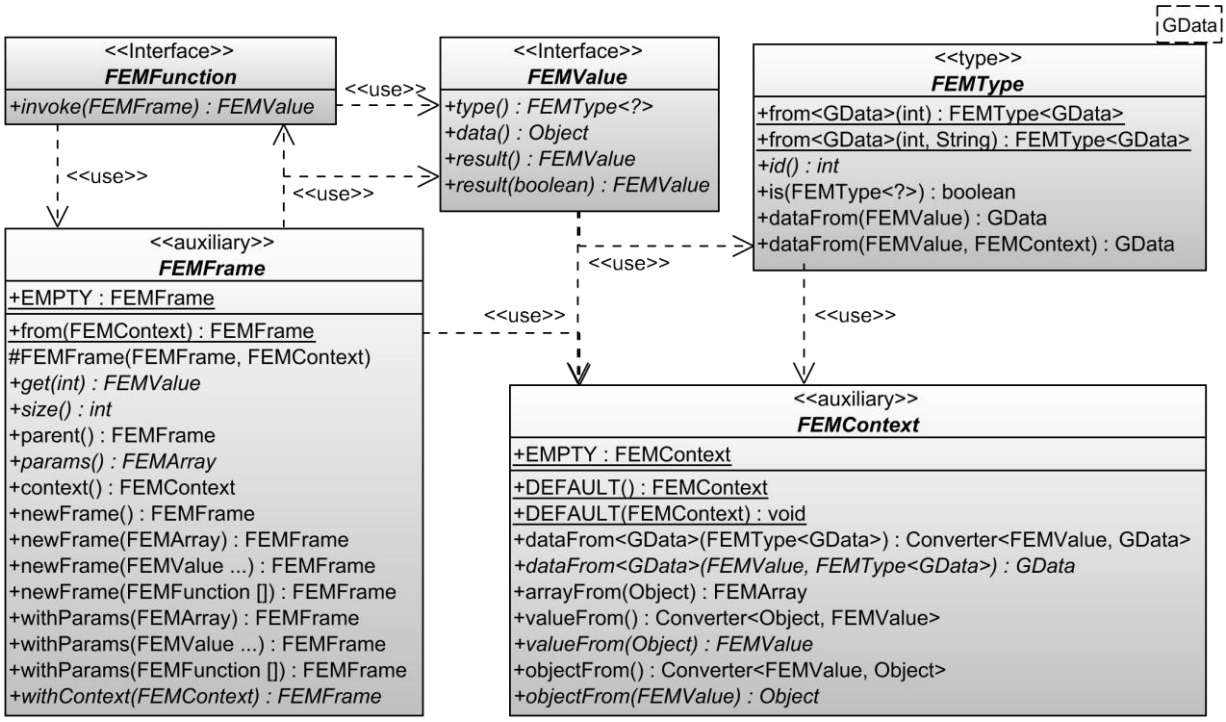

Abbildung 1 Function Evaluation Model

Eine Funktion («*FEMFunction*») wird mit einem Stapelrahmen («*FEMFrame*») aufgerufen und liefert einen Ergebniswert («*FEMValue*»). Werte besitzen einen Datentyp («*FEMType*») sowie Nutzdaten, welche über den impliziten bzw. ein expliziten Kontext («*FEMContext*») in andere Datenformate umgewandelt werden können.

1.1 FEMFunction

+ bee.creative.fem.FEMFunction

Diese Schnittstelle definiert eine Funktion, deren Berechnungsmethode mit einem Stapelrahmen aufgerufen wird und einen Ergebniswert liefert. Aus den Parameterwerten sowie dem Kontextobjekt der Stapelrahmens können hierbei Informationen für die Berechnungen extrahiert werden. Der Zustand des Kontextobjekts kann auch modifiziert werden.

+ invoke(frame: FEMFrame): FEMValue

Diese Methode führt Berechnungen mit dem gegebenen Stapelrahmen durch und gibt den ermittelten Ergebniswert zurück.

1.2 FEMValue

+ bee.creative.fem.FEMValue

Diese Schnittstelle definiert einen Wert, der als Ergebnis der Auswertung einer Funktion oder als Parameter in einem Stapelrahmen zur Auswertung einer Funktion verwendet werden kann. Ein solcher Wert besitzt dazu Nutzdaten mit einem bestimmten Datentyp. Die Konvertierung der Nutzdaten in einen gegebenen Datentyp «type» kann im Rahmen eines gegebenen Kontextobjekts «context» über den Befehl «*context.dataFrom(this, type)*» erfolgen.

`+ type(): FEMType<?>`

Diese Methode gibt den Datentyp zurück.

`+ data(): Object`

Diese Methode gibt die Nutzdaten zurück.

`+ result(): FEMValue`

Diese Methode ist eine Abkürzung für «*`this.result(false)`*».

`+ result(recursive: boolean): FEMValue`

Diese Methode gibt diesen Wert als ausgewerteten und optimierten Ergebniswert zurück. Bei rekursiver Auswertung werden auch die in diesem Wert enthaltenen Werte ausgewertet, z.B. bei einem «*`FEMArray`*». Andernfalls wird nur dieser Wert ausgewertet, z.B. bei einem «*`FEMResult`*».

1.3 FEMType

`+ bee.creative.fem.FEMType<GData>`

Diese Klasse implementiert den abstrakten Datentyp eines Werts («*`FEMValue`*»), analog zur Klasse eines Objekts. Ein solcher Datentyp besitzt Methoden zum Konvertieren der Nutzdaten eines gegebenen Werts sowie zur Prüfung der Kompatibilität zu anderen Datentypen. «GData» ist hierbei der Typ der von «*`FEMType.dataFrom(FEMValue)`*» bzw. «*`FEMType.dataFrom(FEMValue, FEMContext)`*» gelieferten Nutzdaten.

`+ from<GData>(id: int): FEMType<GData>`

Diese Methode gibt einen einfachen Datentyp mit dem gegebenen Identifikator zurück.

`+ from<GData>(id: int, toString: String): FEMType<GData>`

Diese Methode gibt einen einfachen Datentyp mit dem gegebenen Identifikator und der gegebenen Textdarstellung zurück.

`+ id(): int`

Diese Methode gibt den Identifikator dieses Datentyps zurück. Dieser sollte über eine statische Konstante definiert werden, um Fallunterscheidungen mit einem *switch*-Statement umsetzen zu können.

`+ is(type: FEMType<?>): boolean`

Diese Methode gibt nur dann «*`true`*» zurück, wenn ein *cast* in den gegebenen Datentyp zulässig ist. Dies kann der Fall sein, wenn der gegebene Datentyp gleich zu diesem oder ein Vorfahre dieses Datentyps ist. Wenn der gegebene Datentyp «*`null`*» ist, wird «*`false`*» geliefert.

`+ dataFrom(value: FEMValue): GData`

Diese Methode gibt die in den Datentyp «T» konvertierten Nutzdaten des gegebenen Werts zurück. Der Rückgabewert entspricht «FEM*`Context-.DEFAULT().dataFrom(value, this)`*».

`+ dataFrom(value: FEMValue, context: FEMContext): GData`

Diese Methode gibt die in den Datentyp «T» kontextsensitiv konvertierten Nutzdaten des gegebenen Werts zurück. Der Rückgabewert entspricht «*`context.dataFrom(value, this)`*».

1.4 FEMFrame

`+ bee.creative.fem.FEMFrame`

Diese Klasse implementiert einen Stapelrahmen (*stack-frame*), über welchen einer Funktion eine Liste von Parameterwerten sowie ein Kontextobjekt zur Verfügung gestellt wird. Über die Anzahl der zugesicherten Parameterwerte hinaus können von «*`FEMFrame.get(int)`*» auch zusätzliche Parameterwerte aus dem übergeordneten Stapelrahmen bereitgestellt werden. Diese Methode liefert für einen gegebenen «index» immer den gleichen Wert bzw. löst immer die gleiche Ausnahme aus.

Analoges gilt für «*`FEMFrame.size()`*», «*`FEMFrame.params()`*» und «*`FEMFrame.context()`*».

`+ EMPTY: FEMFrame`

Dieses Feld speichert den leeren Stapelrahmen, der keine Parameterwerte bereitstellt, das Kontextobjekt «*`FEMContext.EMPTY`*» verwendet und sich selbst als «*`FEMFrame.parent()`*» nutzt.

`+ from(context: FEMContext): FEMFrame`

Diese Methode gibt einen Stapelrahmen mit dem gegebenen Kontextobjekt zurück. Sie ist eine Abkürzung für «*`EMPTY.withContext(context)`*».

`+ get(index: int): FEMValue`

Diese Methode gibt den «index»-ten Parameterwert evaluiert zurück. Über die Anzahl der zugesicherten Parameterwerte hinaus, können auch zusätzliche Parameterwerte des übergeordneten Stapelrahmens bereitgestellt werden. Genauer wird für einen «index» größer oder gleich der Anzahl der zugesicherten Parameter der Parameterwert «*`this.parent().get(index - size())`*» des übergeordneten Stapelrahmens geliefert.

`+ size(): int`

Diese Methode gibt die Anzahl der Parameterwerte zurück, die zur Verwendung durch eine aufgerufene Funktion bestimmt sind. Über «*`FEMFrame.get(int)`*» werden mindestens so viele Parameterwerte bereitgestellt.

`+ parent(): FEMFrame`

Diese Methode gibt die übergeordneten Parameterdaten zurück.

`+ params(): FEMArray`

Diese Methode gibt eine Wertliste als Sicht auf die zugesicherten Parameterwerte zurück. Die Elemente dieser Wertliste können der *return-by-reference*-Semantik angehören.

`+ context(): FEMContext`

Diese Methode gibt das Kontextobjekt zurück. Funktionen können aus diesem Objekt Informationen für ihre Berechnungen extrahieren oder auch den Zustand dieses Objekts modifizieren. Das Kontextobjekt entspricht dem Kontext «*`this`*» in Java-Methoden.

`+ newFrame(): FEMFrame`

Diese Methode gibt einen neuen Stapelrahmen zurück, welcher keine zugesicherten Parameterwerte besitzt, das Kontextobjekt dieses Stapelrahmens übernimmt und diesen als übergeordneten Stapelrahmen nutzt.

`+ newFrame(params: FEMArray): FEMFrame`

Diese Methode gibt einen neuen Stapelrahmen zurück, welcher die gegebenen zugesicherten Parameterwerte besitzt, das Kontextobjekt dieses Stapelrahmens übernimmt und diesen als übergeordneten Stapelrahmen nutzt.

`+ newFrame(params: FEMValue...): FEMFrame`

Diese Methode gibt einen neuen Stapelrahmen zurück, welcher die gegebenen zugesicherten Parameterwerte besitzt, das Kontextobjekt dieses Stapelrahmens übernimmt und diesen als übergeordneten Stapelrahmen nutzt.

`+ newFrame(params: FEMFunction[]): FEMFrame`

Diese Methode gibt einen neuen Stapelrahmen zurück, welcher die gegebenen Parameterfunktionen zur Berechnung der zugesicherten Parameterwerte verwendet, das Kontextobjekt dieses Stapelrahmens übernimmt und diesen als übergeordneten Stapelrahmen nutzt.

Die zugesicherten Parameterwerte werden mit Hilfe dieses Stapelrahmens und der gegebener Parameterfunktionen ermittelt. Eine Parameterfunktion wird zur Ermittlung eines Parameterwerts einmalig mit diesem Stapelrahmen ausgewertet. Genauer entspricht der «index»-te zugesicherte Parameterwert dem Ergebnis von «*`functions[index].invoke(this)`*». Der Ergebniswert wird zur Wiederverwendung zwischengespeichert. Die über «*`FEMFrame.params()`*» bereitgestellte Liste der Parameterwerte des erzeugten Stapelrahmens liefert die noch nicht über «*`FEMFrame.get(int)`*» ermittelten Parameterwerte als «*`FEMResult`*».

`+ withParams(params: FEMArray): FEMFrame`

Diese Methode gibt diesen Stapelrahmen mit den gegebenen zugesicherten Parameterwerten zurück. Sie ist eine Abkürzung für «*`this.parent().newFrame(params).withContext(this.context())`*».

`+ withParams(params: FEMValue[]...): FEMFrame`

Diese Methode gibt diesen Stapelrahmen mit den gegebenen zugesicherten Parameterwerten zurück. Sie ist eine Abkürzung für «*`this.parent().newFrame(params).withContext(this.context())`*».

`+ withParams(params: FEMFunction[]): FEMFrame`

Diese Methode gibt diesen Stapelrahmen mit den gegebenen zugesicherten Parameterwerten zurück. Sie ist eine Abkürzung für «*`this.parent().newFrame(params).withContext(this.context())`*».

`+ withContext(context: FEMContext): FEMFrame`

Diese Methode gibt diesen Stapelrahmen mit dem gegebenen Kontextobjekt zurück.

1.5 FEMContext

+ bee.creative.fem.FEMContext

Diese Klasse implementiert ein abstraktes Kontextobjekt, das über einen Stapelrahmen der Auswertung von Funktionen bereitgestellt wird und in Funktionen zur Umwandlung von Werten genutzt werden kann.

+ EMPTY: FEMContext

Dieses Feld speichert das leere Kontextobjekt.

«*FEMContext.dataFrom(FEMValue, FEMType)*» dieses Kontextobjekts gibt die Nutzdaten des ihr übergebenen Werts «value» unverändert zurück, wenn sein Datentyp gleich oder einem Nachfahren des ihr übergebenen Datentyps «type» ist, d.h. wenn «*value.type().is(type)*». Andernfalls löst sie eine «*IllegalArgumentException*» aus.

«*FEMContext.valueFrom(Object)*» dieses Kontextobjekts gibt einen gegebenen «*FEMValue*» unverändert zurück und konvertiert «*null*» zu «*FEMVoid*», «*char[]*» und «*String*» zu «*FEMString*», «*byte[]*» zu «*FEMBinary*», «*Float*», «*Double*» und «*BigDecimal*» zu «*FEMDecimal*», alle anderen «*Number*» zu «*FEMInteger*», «*Boolean*» zu «*FEMBoolean*», «*Calendar*» zu «*FEMDatetime*», «*FEMFunction*» zu «*FEMHandler*» und alle anderen Eingaben via «*FEMContext.arrayFrom(Object)*» in ein «*FEMArray*». Im Fehlerfall löst sie eine «*IllegalArgumentException*» aus.

«*FEMContext.objectFrom(FEMValue)*» dieses Kontextobjekts konvertiert «*FEMVoid*» zu «*null*», «*FEMArray*» und die darin enthaltenen Werte rekursiv zu «*Object[]*», «*FEMBinary*» zu «*byte[]*», «*FEMString*» zu «*String*», «*FEMInteger*» und «*FEMDecimal*» zu «*Number*», «*FEMDatetime*» zu «*Calendar*», «*FEMBoolean*» zu «*Boolean*» und alle anderen Werte ihren Nutzdatensatz.

+ DEFAULT(): FEMContext

Diese Methode gibt das Kontextobjekt zurück, das als Rückfallebene für kontextfeie Datentypumwandlungen genutzt wird. Dieses Rückfallkontextobjekt wird u.a. in «*FEMType.dataFrom(FEMValue)*» verwendet.

+ DEFAULT(context: FEMContext): void

Diese Methode setzt den Rückfallkontextobjekt. Wenn das gegebene Kontextobjekt «*null*» ist, wird «*FEMContext.EMPTY*» verwendet.

+ dataFrom<GData>(type: FEMType<?: GData>): Converter<FEMValue, GData>

Diese Methode gibt einen «*Converter*» zurück, der seine Eingabe «input» über den Aufruf «*this.dataFrom(input, type)*» in seine Ausgabe überführt.

+ dataFrom<GData>(value: FEMValue, type: FEMType<GData>): GData

Diese Methode gibt die in Nutzdaten des gegebenen Werts im gegebenen Datentyp «GData» zurück. Hierbei werden die Nutzdaten «*value.data()*» in den geforderten Datentyp konvertiert.

+ arrayFrom(data: Object): FEMArray

Diese Methode konvertiert das gegebene Objekt in eine Wertliste und gibt diese zurück.

Wenn das Objekt ein «*FEMArray*» ist, wird es unverändert zurückgegeben. Wenn es ein natives Array ist, wird jedes seiner Elemente über «*FEMContext.valueFrom(Object)*» in einen Wert überführt und die so entstandene Wertliste geliefert. Wenn es eine «*Collection*» ist, wird diese in ein natives Array überführt, welches anschließend in eine Wertliste umgewandelt wird. Wenn es ein «*Iterable*» ist, wird dieses in eine «*Collection*» überführt, welche anschließend in eine Wertliste umgewandelt wird. Andernfalls wird eine «*IllegalArgumentException*» ausgelöst.

+ valueFrom(): Converter<Object, FEMValue>

Diese Methode gibt einen «*Converter*» zurück, der seine Eingabe «input» via «*this.valueFrom(input)*» in seine Ausgabe überführt.

+ valueFrom(object: Object): FEMValue

Diese Methode gibt einen Wert mit den gegebenen Nutzdaten zurück. Welcher Wert- und Datentyp hierfür verwendet wird, ist der Implementation überlassen.

+ objectFrom(): Converter<FEMValue, Object>

Diese Methode gibt einen «*Converter*» zurück, der seine Eingabe «input» via «*this.objectFrom(input)*» in seine Ausgabe überführt.

+ objectFrom(value: FEMValue): Object

Diese Methode gibt ein «*Object*» zurück, welches via «*FEMContext.valueFrom(Object)*» in einen Wert überführt werden kann, der zum gegebenen Wert äquivalenten ist.

2 FEM-Datentypen

2.1 FEMBaseValue

+ bee.creative.fem.FEMBaseValue: FEMValue, FEMFunction

Diese Klasse implementiert einen abstrakten Wert als «*FEMFunction*». Die «invoke(frame)»-Methode liefert «*this*», sodass Instanzen dieser Klassen das Einpacken in eine «*FEMValueFunction*» nicht benötigen.

+ data<GData>(type: FEMType<GData>): GData

Diese Methode gibt die in den gegebenen Datentyp «GData» kontextfrei konvertierten Nutzdaten dieses Werts zurück. Der Rückgabewert entspricht «*FEMContext.DEFAULT().dataFrom(this, type)*».

+ data<GData>(type: FEMType<GData>, context: FEMContext): GData

Diese Methode gibt die in den gegebenen Datentyp «GData» kontextsensitiv konvertierten Nutzdaten dieses Werts zurück. Der Rückgabewert entspricht «*context.dataFrom(this, type)*».

+ result(recursive: boolean): FEMValue

Diese Methode gibt «*this*» zurück.

+ invoke(frame: FEMFrame): FEMValue

Diese Methode gibt «*this*» zurück.

2.2 FEMNative

+ bee.creative.fem.FEMNative: FEMBaseValue

Diese Klasse implementiert einen Wert mit einem beliebigen nativen Objekt als Nutzdaten.

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<Object>

Dieses Feld speichert den Datentyp.

+ NULL: FEMNative

Dieses Feld speichert den Wert zu «*null*».

+ TRUE: FEMNative

Dieses Feld speichert den Wahrheitswert «*true*».

+ FALSE: FEMNative

Dieses Feld speichert den Wahrheitswert «*false*».

+ from(data: Object): FEMNative

Diese Methode gibt das native Objekt als Wert zurück. Wenn das Objekt bereits ein «*FEMNative*» ist, wird dieses geliefert.

+ FEMNative(data: Object)

Dieser Konstruktor initialisiert das native Objekt.

+ data(): Object

Diese Methode gibt das native Objekt zurück.

+ type(): FEMType<Object>

Diese Methode «*TYPE*» zurück.

2.3 FEMResult

+ bee.creative.fem.FEMResult: FEMBaseValue

Diese Klasse implementiert einen Wert, der als Ergebniswert einer Funktion mit *return-by-reference*-Semantik sowie als Parameterwert eines Aufrufs mit *call-by-reference*-Semantik eingesetzt werden kann.

Der Wert kapselt dazu eine gegebene Funktion sowie einen gegebenen Stapelrahmen und wertet diese Funktion erst dann mit dem diesem Stapelrahmen einmalig aus, wenn auf Datentyp oder Nutzdaten zugegriffen wird. Der von der Funktion berechnete Ergebniswert wird zur Wiederverwendung zwischengespeichert. Nach der einmaligen Auswertung der Funktion werden die Verweise auf Stapelrahmen und Funktion aufgelöst.

+ from(frame: FEMFrame, function: FEMFunction): FEMResult

Diese Methode den Ergebniswert des Aufrufs der gegebenen Funktion mit dem gegebenen Stapelrahmen mit *call-by-reference*-Semantik zurück. Der gelieferte Ergebniswert verzögert die Auswertung der Funktion bis zum ersten Lesen seines Datentyp bzw. seiner Nutzdaten.

+ data(): Object

Diese Methode gibt «*this.result().data()*» zurück.

+ type(): FEMType<?>

Diese Methode gibt «*this.result().type()*» zurück.

+ frame(): FEMFrame

Diese Methode gibt die Stapelrahmen oder «*null*» zurück. Der erste Aufruf von «*FEMResult.result(boolean)*» setzt die Stapelrahmen auf «*null*».

+ function(): FEMFunction

Diese Methode gibt die Funktion oder «*null*» zurück. Der erste Aufruf von «*FEMResult.result(boolean)*» setzt die Funktion auf «*null*».

+ result(recursive: boolean): FEMValue

Diese Methode gibt das Ergebnis der Auswertung der Funktion mit den Stapelrahmen zurück. Dieser Ergebniswert wird nur beim ersten Aufruf dieser Methode ermittelt und zur Wiederverwendung zwischengespeichert. Dabei werden die Verweise auf Stapelrahmen und Funktion aufgelöst.

2.4 FEMVoid

+ bee.creative.fem.FEMVoid: FEMBaseValue

Diese Klasse implementiert den unveränderlichen Leerwert.

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMVoid>

Dieses Feld speichert den Datentyp.

+ INSTANCE: FEMVoid

Dieses Feld speichert den Leerwert.

+ from(value: String): FEMVoid

Diese Methode gibt den Leerwert nur dann zurück, wenn die gegebene Zeichenkette gleich «*"void"*» ist.

+ from(value: FEMValue): FEMVoid

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMVoid

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ data(): FEMVoid

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMVoid>

Diese Methode «*TYPE*» zurück.

+ toString(): String

Diese Methode gibt «*"void"*» zurück.

2.5 FEMArray

+ bee.creative.fem.FEMArray: FEMBaseValue, Items<FEMValue>, Iterable<FEMValue>

Diese Klasse implementiert eine unveränderliche Liste von Werten sowie Methoden zur Erzeugung solcher Wertlisten aus nativen Arrays und «*Iterable*».

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMArray>

Dieses Feld speichert den Datentyp.

+ EMPTY: FEMArray

Dieses Feld speichert die leere Wertliste.

+ from(items: FEMValue[]): FEMArray

Diese Methode konvertiert die gegebenen Werte in eine Wertliste und gibt diese zurück. Das gegebene Array wird Kopiert, sodass spätere Änderungen am gegebenen Array nicht auf die erzeugte Wertliste übertragen werden.

+ from(item: FEMValue, length: int): FEMArray

Diese Methode gibt eine uniforme Wertliste mit der gegebenen Länge zurück, deren Werte alle gleich dem gegebenen sind.

+ from(items: Iterable<?: FEMValue>): FEMArray

Diese Methode konvertiert die gegebenen Werte in eine Wertliste und gibt diese zurück.

+ from(items: Collection<?: FEMValue>): FEMArray

Diese Methode konvertiert die gegebenen Werte in eine Wertliste und gibt diese zurück.

+ from(value: FEMValue): FEMArray

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMArray

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ data(): FEMArray

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMArray>

Diese Methode «*TYPE*» zurück.

+ value(): FEMValue[]

Diese Methode konvertiert diese Wertliste in ein «*FEMValue[]*» und gibt dieses zurück.

+ get(index: int): FEMValue

Diese Methode gibt den «index»-ten Wert zurück. Wenn «index» ungültig ist, wird eine «*IndexOutOfBoundsException*» ausgelöst.

+ length(): int

Diese Methode gibt die Länge, d.h. die Anzahl der Werte in der Wertliste zurück.

+ concat(that: FEMArray): FEMArray

Diese Methode gibt eine Sicht auf die Verkettung dieser Wertliste mit der gegebenen Wertliste zurück.

+ section(offset: int, length: int): FEMArray

Diese Methode gibt eine Sicht auf einen Abschnitt dieser Wertliste zurück. Wenn der Abschnitt nicht innerhalb dieser Wertliste liegt oder eine negative Länge hätte, wird eine «*IllegalArgumentException*» ausgelöst.

`+ reverse(): FEMArray`

Diese Methode gibt eine rückwärts geordnete Sicht auf diese Wertliste zurück.

`+ compact(): FEMArray`

Diese Methode gibt die Werte dieser Wertliste in einer performanteren oder zumindest gleichwertigen Wertliste zurück.

`+ find(that: FEMValue, offset: int): int`

Diese Methode gibt die Position des ersten Vorkommens des gegebenen Werts «that» innerhalb dieser Wertliste zurück. Die Suche beginnt an der gegebenen Position «offset». Bei einer erfolglosen Suche wird «*-1*» geliefert. Wenn «offset» ungültig ist, wird eine «*IllegalArgumentException*» ausgelöst.

`+ find(that: FEMArray, offset: int): int`

Diese Methode gibt die Position des ersten Vorkommens der gegebenen Wertliste «that» innerhalb dieser Wertliste zurück. Die Suche beginnt an der gegebenen «offset». Bei einer erfolglosen Suche wird «*-1*» geliefert. Wenn «offset» ungültig ist, wird eine «*IllegalArgumentException*» ausgelöst.

`+ export(target: Collector): boolean`

Diese Methode fügt alle Werte dieser Wertliste vom ersten zum letzten geordnet an den gegebenen «*Collector*» «target» an. Das Anfügen wird vorzeitig abgebrochen, wenn dessen «*push(value)*»-Methode «*false*» liefert.

`+ hash(): int`

Diese Methode gibt den Streuwert zurück.

`+ equals(that: FEMArray): boolean`

Diese Methode gibt nur dann «*true*» zurück, wenn diese Wertliste gleich der gegebenen ist.

`+ compare(that: FEMArray, order: Comparator<FEMValue>): int`

Diese Methode gibt «*-1*», «*0*» bzw. «*+1*» zurück, wenn die lexikographische Ordnung dieser Wertliste kleiner, gleich oder größer als die der gegebenen Wertliste ist. Die Werte werden über den gegebenen «*Comparator*» «order» verglichen.

`+ toList(): List<FEMValue>`

Diese Methode gibt diese Wertliste als «*List*» zurück.

`+ toString(): String`

Diese Methode gibt die Textdarstellung zurück. Diese besteht aus den in eckige Klammern «*"["*» und «*"]"*» eingeschlossenen sowie mit Semikolon und Leerzeichen «*"; "*» separierten Textdarstellungen der Elemente.

2.6 FEMArray.Collector

`+ bee.creative.fem.FEMArray.Collector`

Diese Schnittstelle definiert ein Objekt zum geordneten Sammeln von Werten einer Wertliste in der Methode «*FEMArray.export(Collector)*».

`+ push(value: FEMValue): boolean`

Diese Methode fügt den gegebenen Wert an das Ende der Sammlung an. Der Rückgabewert ist «*true*», wenn das Sammeln fortgeführt werden soll, bzw. «*false*», wenn es abgebrochen werden soll.

2.7 FEMBinary

`+ bee.creative.fem.FEMBinary: FEMBaseValue, Iterable<Byte>`

Diese Klasse implementiert eine unveränderliche Bytefolge, deren Verkettungen, Anschnitte und Umkehrungen als Sichten auf die grundlegenden Bytefolgen realisiert sind.

`+ ID: int`

Dieses Feld speichert den Identifikator von «*TYPE*».

`+ TYPE: FEMType<FEMBinary>`

Dieses Feld speichert den Datentyp.

`+ EMPTY: FEMBinary`

Dieses Feld speichert die leere Bytefolge.

`+ from(items: byte[]): FEMBinary`

Diese Methode gibt eine Bytefolge mit den gegebenen Bytes zurück. Das gegebene Array wird hierbei kopiert.

`+ from(item: byte, length: int): FEMBinary`

Diese Methode gibt eine uniforme Bytefolge mit der gegebenen Länge zurück, deren Bytes alle gleich dem gegebenen sind.

`+ from(string: String): FEMBinary`

Diese Methode gibt eine neue Bytefolge mit dem in der gegebenen Zeichenkette kodierten Wert zurück. Das Format der Zeichenkette entspricht dem der Textdarstellung.

`+ from(array: MMFArray): FEMBinary`

Diese Methode gibt eine Bytefolge mit den gegebenen Zahlen zurück. Wenn diese Zahlen nicht als «*UNI8*» oder «*UINT8*» vorliegt, wird eine «*IllegalArgumentException*» ausgelöst.

`+ from(value: FEMValue): FEMBinary`

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

`+ from(value: FEMValue, context: FEMContext): FEMBinary`

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

`+ toChar(hexDigit: int): char`

Diese Methode gibt das Zeichen («*'0'…'9'*», «*'A'…'F'*») zur gegebenen hexadezimalen Ziffer («*0…15*») zurück. Wenn «hexDigit» ungültig ist, wird eine «*IllegalArgumentException*» ausgelöst.

`+ toDigit(hexChar: int): int`

Diese Methode gibt die hexadezimale Ziffer («*0…15*») zum gegebenen Zeichen («*'0'…'9'*», «*'A'…'F'*») zurück. Wenn «hexChar» ungültig ist, wird eine «*IllegalArgumentException*» ausgelöst.

`+ data(): FEMBinary`

Diese Methode gibt «*this*» zurück.

`+ type(): FEMType<FEMBinary>`

Diese Methode «*TYPE*» zurück.

`+ value(): byte[]`

Diese Methode konvertiert diese Bytefolge in ein «*byte[]*» und gibt dieses zurück.

`+ get(index: int): byte`

Diese Methode gibt das «index»-te Byte zurück. Wenn «index» ungültig ist, wird eine «*IndexOutOfBoundsException*» ausgelöst.

`+ length(): int`

Diese Methode gibt die Länge, d.h. die Anzahl der Bytes in der Bytefolge zurück.

`+ concat(that: FEMBinary): FEMBinary`

Diese Methode gibt eine Sicht auf die Verkettung dieser Bytefolge mit der gegebenen Bytefolge zurück.

`+ section(offset: int, length: int): FEMBinary`

Diese Methode gibt eine Sicht auf einen Abschnitt dieser Bytefolge zurück. Wenn der Abschnitt nicht innerhalb dieser Bytefolge liegt oder eine negative Länge hätte, wird eine «*IllegalArgumentException*» ausgelöst.

`+ reverse(): FEMBinary`

Diese Methode gibt eine rückwärts geordnete Sicht auf diese Bytefolge zurück.

`+ compact(): FEMBinary`

Diese Methode gibt die Bytes dieser Bytefolge in einer performanteren oder zumindest gleichwertigen Bytefolge zurück.

+ find(that: byte, offset: int): int

Diese Methode gibt die Position des ersten Vorkommens des gegebenen Bytewerts «that» innerhalb dieser Bytefolge zurück. Die Suche beginnt an der gegebenen Position «offset». Bei einer erfolglosen Suche wird «*-1*» geliefert. Wenn «offset» ungültig ist, wird eine «*IllegalArgumentException*» ausgelöst.

+ find(that: FEMBinary, offset: int): int

Diese Methode gibt die Position des ersten Vorkommens der gegebenen Bytefolge «that» innerhalb dieser Bytefolge zurück. Die Suche beginnt an der gegebenen «offset». Bei einer erfolglosen Suche wird «*-1*» geliefert. Wenn «offset» ungültig ist, wird eine «*IllegalArgumentException*» ausgelöst.

+ export(target: Collector): boolean

Diese Methode fügt alle Bytes dieser Bytefolge vom ersten zum letzten geordnet an den gegebenen «*Collector*» «target» an. Das Anfügen wird vorzeitig abgebrochen, wenn dessen «*Collector.push(byte)*»-Methode «*false*» liefert.

+ hash(): int

Diese Methode gibt den Streuwert zurück.

+ equals(that: FEMBinary): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn diese Bytefolge gleich der gegebenen ist.

+ compare(that: FEMBinary): int

Diese Methode gibt «*-1*», «*0*» bzw. «*+1*» zurück, wenn die lexikographische Ordnung dieser Bytefolge kleiner, gleich oder größer als die der gegebenen Bytefolge ist. Die Bytewerte werden als «*UINT8*» verglichen.

+ toString(): String

Diese Methode gibt die Textdarstellung dieser Bytefolge zurück. Die Textdarstellung besteht aus der Zeichenkette «*"0x"*» und den Bytes dieser Bytefolge vom ersten zum letzten geordnet in hexadezimalen Ziffern («*'0'…'9'*», «*'A'…'F'*»).

2.8 FEMBinary.Collector

+ bee.creative.fem.FEMBinary.Collector

Diese Schnittstelle definiert ein Objekt zum geordneten Sammeln von Bytes einer Bytefolge in der Methode «*FEMBinary.export(Collector)*».

+ push(value: byte): boolean

Diese Methode fügt den gegebenen Bytewert an das Ende der Sammlung an. Der Rückgabewert ist «*true*», wenn das Sammeln fortgeführt werden soll, bzw. «*false*», wenn es abgebrochen werden soll.

2.9 FEMString

+ bee.creative.fem.FEMString: FEMBaseValue, Iterable<Integer>

Diese Klasse implementiert eine Zeichenkette, deren Verkettungen, Anschnitte und Umkehrungen als Sichten auf die grundlegenden Zeichenketten realisiert sind.

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMString>

Dieses Feld speichert den Datentyp.

+ EMPTY: FEMString

Dieses Feld speichert die leere Zeichenkette.

+ from(items: int[]): FEMString

Diese Methode eine Zeichenkette mit den gegebenen UTF32-kodierten Codepoints zurück.

+ from(items: byte[]): FEMString

Diese Methode eine Zeichenkette mit den gegebenen UTF8-kodierten Codepoints zurück.

`+ from(items: char[]): FEMString`

Diese Methode eine Zeichenkette mit den gegebenen UTF16-kodierten Codepoints zurück.

`+ from(string: String): FEMString`

Diese Methode eine Zeichenkette mit den gegebenen Codepoints zurück.

`+ from(item: int, length: int): FEMString`

Diese Methode gibt eine uniforme Zeichenkette mit der gegebenen Länge zurück, deren Codepoints alle gleich dem gegebenen sind.

`+ from(array: MMFArray): FEMString`

Diese Methode gibt eine Zeichenkette mit den gegebenen Zahlen zurück. Abhängig davon, ob die Zahlenliste aus INT8/UINT8-, INT16/UINT16- oder INT32-Zahlen besteht, werden diese als UTF8-, UTF16- bzw. UTF32-kodierte Codepoints interpretiert.

`+ from(value: FEMValue): FEMString`

Diese Methode ist eine Abkürzung für «*`FEMContext.DEFAULT().dataFrom(value, TYPE)`*».

`+ from(value: FEMValue, context: FEMContext): FEMString`

Diese Methode ist eine Abkürzung für «*`context.dataFrom(value, TYPE)`*».

`+ data(): FEMString`

Diese Methode gibt «*`this`*» zurück.

`+ type(): FEMType<FEMString>`

Diese Methode gibt «*`TYPE`*» zurück.

`+ value(): int[]`

Diese Methode gibt die Codepoint in UTF32-Kodierung zurück.

`+ get(index: int): int`

Diese Methode gibt das «index»-te Zeichen als Codepoint zurück.

`+ length(): int`

Diese Methode gibt die Länge, d.h. die Anzahl der Zeichen in der Zeichenkette zurück.

`+ concat(that: FEMString): FEMString`

Diese Methode gibt eine Sicht auf die Verkettung dieser Zeichenkette mit der gegebenen Zeichenkette zurück.

`+ section(offset: int, length: int): FEMString`

Diese Methode gibt eine Sicht auf einen Abschnitt dieser Zeichenkette zurück. Wenn der Abschnitt nicht innerhalb dieser Zeichenkette liegt oder eine negative Länge hätte, wird eine «*`IllegalArgumentException`*» ausgelöst.

`+ reverse(): FEMString`

Diese Methode gibt eine rückwärts geordnete Sicht auf diese Zeichenkette zurück.

`+ compact(): FEMString`

Diese Methode gibt die Codepoints dieser Zeichenkette in einer performanteren oder zumindest gleichwertigen Zeichenkette zurück.

`+ find(that: int, offset: int): int`

Diese Methode gibt die Position des ersten Vorkommens des gegebenen Zeichens «that» innerhalb dieser Zeichenkette zurück. Die Suche beginnt an der gegebenen Position «offset». Bei einer erfolglosen Suche wird «*`-1`*» geliefert. Wenn «offset» ungültig ist, wird eine «*`IllegalArgumentException`*» ausgelöst.

`+ find(that: FEMString, offset: int): int`

Diese Methode gibt die Position des ersten Vorkommens der gegebenen Zeichenkette «that» innerhalb dieser Zeichenkette zurück. Die Suche beginnt an der gegebenen Position «offset». Bei einer erfolglosen Suche wird «*`-1`*» geliefert. Wenn «offset» ungültig ist, wird eine «*`IllegalArgumentException`*» ausgelöst.

`+ export(target: Collector): boolean`

Diese Methode fügt alle Codepoints dieser Zeichenkette vom ersten zum letzten geordnet an den gegebenen «*`Collector`*» an. Das Anfügen wird vorzeitig abgebrochen, wenn dessen «*`Collector.push(int)`*»-Methode «*`false`*» liefert.

+ hash(): int

Diese Methode gibt den Streuwert zurück.

+ equals(that: FEMString): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn diese Zeichenkette gleich der gegebenen ist.

+ compare(that: FEMString): int

Diese Methode «*-1*», «*0*» bzw. «*+1*» zurück, wenn die lexikographische Ordnung dieser Zeichenkette kleiner, gleich oder größer als die der gegebenen Zeichenkette ist.

+ toBytes(): byte[]

Diese Methode gibt die Codepoint in UTF8-Kodierung zurück.

+ toChars(): char[]

Diese Methode gibt die Codepoint in UTF16-Kodierung zurück.

+ toString(): String

Diese Methode gibt diesen Zeichenkette als «*String*» zurück.

2.10 FEMString.Collector

+ bee.creative.fem.FEMString.Collector

Diese Schnittstelle definiert ein Objekt zum geordneten Sammeln von Codepoints einer Zeichenkette in der Methode «*FEMString.export(Collector)*».

+ push(value: int): boolean

Diese Methode fügt den gegebenen Codepoint an das Ende der Sammlung an. Der Rückgabewert ist «*true*», wenn das Sammeln fortgeführt werden soll, bzw. «*false*», wenn es abgebrochen werden soll.

2.11 FEMObject

+ bee.creative.fem.FEMObject: FEMBaseValue, Comparable<FEMObject>

Diese Klasse implementiert eine unveränderliche Referenz auf ein logisches Objekt, welches im Rahmen seines Besitzers über einen Objektschlüssel identifiziert wird. Datentyp und Besitzer des Objekts werden über eine Typkennung bzw. Besitzerkennung angegeben. Die Besitzerkennung kann beispielsweise eine über den «*FEMContext*» erreichbare Objektliste identifizieren, deren Elemente die referenzierten Objekte darstellen. Der Objektschlüssel könnte hierbei der Position eines Objekts in solch einer Liste entsprechen. Alternativ zur Besitzerkennung könnte hierbei auch die Typkennung genutzt werden.

Die Wertebereiche für Objektschlüssel, Typkennungen und Besitzerkennungen sind «*0…2147483647*», «*0…65535*» bzw. «*0…65535*».

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMObject>

Dieses Feld speichert den Datentyp.

+ EMPTY: FEMObject

Dieses Feld speichert die Referenz, deren Komponenten alle «*0*» sind.

+ from(string: String): FEMObject

Diese Methode gibt eine neue Referenz mit dem in der gegebenen Zeichenkette kodierten Wert zurück. Das Format der Zeichenkette entspricht dem der Textdarstellung.

+ from(ref: int, type: int, owner: int): FEMObject

Diese Methode gibt eine neue Referenz mit den gegebenen Eigenschaften zurück.

+ from(value: FEMValue): FEMObject

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMObject

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ FEMObject(value: long)

Dieser Konstruktor initialisiert die interne Darstellung der Referenz.

+ data(): FEMObject

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMObject>

Diese Methode «*TYPE*» zurück.

+ value(): long

Diese Methode gibt die interne Darstellung der Referenz zurück. Die 64 Bit von MBS zum LSB sind:
«0» - 1 Bit, «refValue» - 31 Bit, «ownerValue» - 16 Bit, «typeValue» - 16 Bit

+ refValue(): int

Diese Methode gibt den Objektschlüssel zurück.

+ typeValue(): int

Diese Methode gibt die Typkennung zurück.

+ ownerValue(): int

Diese Methode gibt die Besitzerkennung zurück.

+ withRef(ref: int): FEMObject

Diese Methode gibt diese Referenz mit dem gegebenen Objektschlüssel zurück.

+ withType(type: int): FEMObject

Diese Methode gibt diese Referenz mit der gegebenen Typkennung zurück.

+ withOwner(owner: int): FEMObject

Diese Methode gibt diese Referenz mit der gegebenen Besitzerkennung zurück.

+ hash(): int

Diese Methode gibt den Streuwert zurück.

+ equals(that: FEMObject): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn diese Referenz gleich der gegebenen ist.

+ compare(that: FEMObject): int

Diese Methode gibt «*-1*», «*0*» bzw. «*+1*» zurück, wenn die Ordnung dieser Referenz kleiner, gleich bzw. größer als die der gegebenen Referenz ist.

+ toString(): String

Diese Methode gibt die Textdarstellung dieser Referenz zurück. Das Format der Textdarstellung ist «#REF.TYPE:OWNER».

2.12 FEMInteger

+ bee.creative.fem.FEMInteger: FEMBaseValue, Comparable<FEMInteger>

Diese Klasse implementiert eine unveränderliche Dezimalzahl. Intern wird die Dezimalzahl als «*Long*» dargestellt.

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMInteger>

Dieses Feld speichert den Datentyp.

+ EMPTY: FEMInteger

Dieses Feld speichert die Dezimalzahl «*0*».

+ from(value: long): FEMInteger

Diese Methode gibt eine neue Dezimalzahl mit dem gegebenen Wert zurück.

+ from(value: Number): FEMInteger

Diese Methode gibt eine neue Dezimalzahl mit dem gegebenen Wert zurück.

+ from(value: String): FEMInteger

Diese Methode gibt eine neue Dezimalzahl mit dem in der gegebenen Zeichenkette kodierten Wert zurück. Das Format der Zeichenkette entspricht dem der Textdarstellung.

+ from(value: FEMValue): FEMInteger

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMInteger

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ FEMInteger(value: long)

Dieser Konstruktor initialisiert die interne Darstellung der Dezimalzahl.

+ data(): FEMInteger

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMInteger>

Diese Methode «*TYPE*» zurück.

+ value(): long

Diese Methode gibt die interne Darstellung der Dezimalzahl zurück.

+ hash(): int

Diese Methode gibt den Streuwert zurück.

+ equals(that: FEMInteger): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn diese Dezimalzahl gleich der gegebenen ist.

+ compare(that: FEMInteger): int

Diese Methode gibt «*-1*», «*0*» bzw. «*+1*» zurück, wenn diese Dezimalzahl kleiner, gleich oder größer als die gegebene Dezimalzahl ist.

+ toNumber(): Number

Diese Methode gibt diese Dezimalzahl als «*Number*» zurück.

+ toString(): String

Diese Methode gibt die Textdarstellung dieser Dezimalzahl zurück.

2.13 FEMDecimal

+ bee.creative.fem.FEMDecimal: FEMBaseValue, Comparable<FEMDecimal>

Diese Klasse implementiert einen unveränderlichen Dezimalbruch. Intern wird der Dezimalbruch als «*double*» dargestellt.

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMDecimal>

Dieses Feld speichert den Datentyp.

+ EMPTY: FEMDecimal

Dieses Feld speichert den Dezimalbruch «*NaN*».

+ from(value: double): FEMDecimal

Diese Methode gibt einen neuen Dezimalbruch mit dem gegebenen Wert zurück.

+ from(value: Number): FEMDecimal

Diese Methode gibt einen neuen Dezimalbruch mit dem gegebenen Wert zurück.

+ from(value: String): FEMDecimal

Diese Methode gibt einen neuen Dezimalbruch mit dem in der gegebenen Zeichenkette kodierten Wert zurück. Das Format der Zeichenkette entspricht dem der Textdarstellung.

+ from(value: FEMValue): FEMDecimal

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMDecimal

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ FEMDecimal(value: double)

Dieser Konstruktor initialisiert die interne Darstellung des Dezimalbruchs.

+ data(): FEMDecimal

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMDecimal>

Diese Methode «*TYPE*» zurück.

+ value(): double

Diese Methode gibt die interne Darstellung des Dezimalbruchs zurück.

+ hash(): int

Diese Methode gibt den Streuwert zurück.

+ equals(that: FEMDecimal): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn dieser Dezimalbruch gleich dem gegebenen ist.

+ compare(that: FEMDecimal, undefined: int): int

Diese Methode gibt «*-1*», «*0*» bzw. «*+1*» zurück, wenn dieser Dezimalbruch kleiner, gleich oder größer als der gegebene Dezimalbruch ist. Wenn die Dezimalbrüche nicht vergleichbar sind, wird «undefined» geliefert.

+ toNumber(): Number

Diese Methode gibt diesen Dezimalbruch als «*Number*» zurück.

+ toString(): String

Diese Methode gibt die Textdarstellung dieses Dezimalbruchs zurück.

2.14 FEMHandler

+ bee.creative.fem.FEMHandler: FEMBaseValue, TracerInput

Diese Klasse implementiert einen unveränderlichen Funktionszeiger, d.h. eine als «*FEMValue*» verpackte Funktion. Intern wird der Funktionszeiger als «*FEMFunction*» dargestellt.

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMHandler>

Dieses Feld speichert den Datentyp.

+ from(data: FEMFunction): FEMHandler

Diese Methode gibt die gegebene Funktion als Funktionszeiger zurück.

+ from(value: FEMValue): FEMHandler

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMHandler

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ FEMHandler(value: FEMFunction)

Dieser Konstruktor initialisiert die Funktion.

+ data(): FEMHandler

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMHandler>

Diese Methode «*TYPE*» zurück.

+ value(): FEMFunction

Diese Methode gibt die Funktion zurück.

+ hash(): int

Diese Methode gibt den Streuwert zurück.

+ equals(that: FEMHandler): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn dieser Funktionszeiger gleich der gegebenen ist.

2.15 FEMBoolean

+ bee.creative.fem.FEMBoolean: FEMBaseValue, Comparable<FEMBoolean>

Diese Klasse implementiert einen unveränderlichen Wahrheitswert. Intern wird der Wahrheitswert als «*boolean*» dargestellt.

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMBoolean>

Dieses Feld speichert den Datentyp.

+ TRUE: FEMBoolean

Dieses Feld speichert den Wahrheitswert «*true*».

+ FALSE: FEMBoolean

Dieses Feld speichert den Wahrheitswert «*false*».

+ from(data: boolean): FEMBoolean

Diese Methode gibt einen neuen Wahrheitswert mit dem gegebenen Wert zurück.

+ from(data: Boolean): FEMBoolean

Diese Methode gibt einen neuen Wahrheitswert mit dem gegebenen Wert zurück.

+ from(value: String): FEMBoolean

Diese Methode gibt einen neuen Wahrheitswert mit dem in der gegebenen kodierten Wert zurück. Das Format der Zeichenkette entspricht dem der Textdarstellung.

+ from(value: FEMValue): FEMBoolean

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMBoolean

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ data(): FEMBoolean

Diese Methode gibt «*this*» zurück.

`+ type(): FEMType<FEMBoolean>`

Diese Methode «*`TYPE`*» zurück.

`+ value(): boolean`

Diese Methode gibt die interne Darstellung des Wahrheitswerts zurück.

`+ hash(): int`

Diese Methode gibt den Streuwert zurück.

`+ equals(that: FEMBoolean): boolean`

Diese Methode gibt nur dann «*`true`*» zurück, wenn dieser Wahrheitswert gleich dem gegebenen ist.

`+ compare(that: FEMBoolean): int`

Diese Methode gibt «*`-1`*», «*`0`*» bzw. «*`+1`*» zurück, wenn dieser Wahrheitswert kleiner, gleich oder größer als der gegebene Wahrheitswert ist.

`+ toBoolean(): Boolean`

Diese Methode gibt diesen Wahrheitswert als «*`Boolean`*» zurück.

`+ toString(): String`

Diese Methode gibt die Textdarstellung dieses Wahrheitswerts zurück. Für die Wahrheitswerte «*`true`*» und «*`false`*» sind die Textdarstellungen «*`"true"`*» und «*`"false"`*».

2.16 FEMDatetime

`+ bee.creative.fem.FEMDatetime: FEMBaseValue, Comparable<FEMDatetime>`

Diese Klasse implementiert eine Zeitangabe mit Datum, Uhrzeit und/oder Zeitzone im Gregorianischen Kalender. Intern wird die Zeitangabe als ein «*`Long`*» dargestellt.

Das Datum kann unspezifiziert sein oder aus Jahr, Monat sowie Tag bestehen und im Bereich «*`15.10.1582…31.12.9999`*» liegen. Die Uhrzeit kann unspezifiziert sein oder aus Stunde, Minute, Sekunde sowie Millisekunde bestehen und im Bereich «*`00:00:00.000…24:00:00.000`*» liegen. Die Zeitzone kann unspezifiziert sein oder aus Stunde sowie Minute bestehen und im Bereich «*`-14:00…+14:00`*» liegen.

Jahr

Der Zahlenwert für das Jahr entspricht der Anzahl der Jahre seit dem Beginn des Gregorianischen Kalenders erhöht um «*`1582`*». Unterstützte Zahlenwerte für das Jahr sind «*`1582…9999`*». Ein reguläres Jahr hat «*`365`*» Tage, ein Schaltjahr hat «*`366`*» Tage.

Monat

Der Zahlenwert für den Monat entspricht der Anzahl der Monate seit Beginn des Jahres erhöht um «*`1`*». Unterstützte Zahlenwerte für den Monat sind «*`1…12`*».

Januar = «*`Calendar.JANUARY + 1`*»
Februar = «*`Calendar.FEBRUARY + 1`*»
März = «*`Calendar.MARCH + 1`*»
April = «*`Calendar.APRIL + 1`*»
Mai = «*`Calendar.MAY + 1`*»
Juni = «*`Calendar.JUNE + 1`*»
Juli = «*`Calendar.JULY + 1`*»
August = «*`Calendar.AUGUST + 1`*»
September = «*`Calendar.SEPTEMBER + 1`*»
Oktober = «*`Calendar.OCTOBER + 1`*»
November = «*`Calendar.NOVEMBER + 1`*»
Dezember = «*`Calendar.DECEMBER + 1`*»

Tag (Tag in einem Monat)

Der Zahlenwert für den Tag entspricht der Anzahl der Tage seit Beginn des Monats erhöht um «*`1`*». Unterstützte Zahlenwerte für den Monat sind «*`1…31`*», wobei einige Monate auch abhängig von Schaltjahren geringere Obergrenzen besitzen.

Jahrestag (Tag in einem Jahr)

Der Zahlenwert für den Jahrestag entspricht der Anzahl der Tage seit dem Beginn des Jahres erhöht um «*`1`*». Unterstützte Zahlenwerte für den Jahrestag sind «*`1…366`*».

Wochentag (Tag in einer Woche)

Der Zahlenwert für den Wochentag entspricht der Anzahl der Tage seit Beginn der Woche erhöht um «*`1`*». Unterstützte Zahlenwerte für den Wochentag sind «*`1…7`*».

Sonntag = «*Calendar.SUNDAY*»
Montag = «*Calendar.MONDAY*»
Dienstag = «*Calendar.TUESDAY*»
Mittwoch = «*Calendar.WEDNESDAY*»
Donnerstag = «*Calendar.THURSDAY*»
Freitag = «*Calendar.FRIDAY*»
Samstag = «*Calendar.SATURDAY*»

Kalendertag (Tag im Kalender)

Der Zahlenwert für den Kalendertag entspricht der Anzahl der Tage seit dem Beginn des Gregorianischen Kalenders am Freitag dem «*15.10.1582*». Unterstützte Zahlenwerte für den Kalendertag sind «*0…3074323*».

Tagesmillis (Millisekunden am Tag)

Der Zahlenwert für die Tagesmillis entspricht der Anzahl der Millisekunden seit «*00:00:00.000*». Unterstützte Zahlenwerte für die Tagesmillis sind «*0…86400000*».

Datum

Das Datum einer Zeitangabe kann entweder als Kalendertag oder als Kombination von Jahr, Monat und Tag angegeben werden.

Uhrzeit

Die Uhrzeit einer Zeitangabe kann als Tagesmillis oder als Kombination von Stunden, Minuten, Sekunden und Millisekunden angegeben werden. Unterstützte Zahlenwerte für Stunden, Minuten, Sekunden und Millisekunden sind «*0…24*», «*0…59*», «*0…59*» bzw. «*0…999*».

Zeitzone (Zeitzonenverschiebung)

Der Zahlenwert für die Zeitzone entspricht der Zeitzonenverschiebung gegenüber UTC in Minuten. Unterstützte Zahlenwerte für die Zeitzone sind «*-840…840*».

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMDatetime>

Dieses Feld speichert den Datentyp.

+ EMPTY: FEMDatetime

Dieses Feld speichert die leere Zeitangabe ohne Datum, ohne Uhrzeit und ohne Zeitzone.

+ now(): FEMDatetime

Diese Methode gibt den aktuellen Zeitpunkt zurück.

+ from(millis: long): FEMDatetime

Diese Methode gibt einen Zeitpunkt zurück, der die gegebene Anzahl an Millisekunden nach dem Zeitpunkt «*1970-01-01T00:00:00Z*» liegt.

+ from(string: String): FEMDatetime

Diese Methode gibt eine Zeitangabe mit den in der gegebenen Zeitangabe kodierten Komponenten zurück. Das Format der Zeichenkette entspricht dem der Textdarstellung.

+ from(calendar: Calendar): FEMDatetime

Diese Methode gibt eine Zeitangabe mit dem Datum, der Uhrzeit und der Zeitzone des gegebenen «*Calendar*» zurück und ist eine Abkürzung für «*FEE_Datetime.EMPTY.withDate(calendar).withTime(calendar).withZone(calendar)*».

+ from(value: FEMValue): FEMDatetime

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMDatetime

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ fromDate(calendarday: int): FEMDatetime

Diese Methode gibt eine Zeitangabe mit dem Datum zum gegebenen Kalendertag zurück und ist eine Abkürzung für «*FEE_Datetime.EMPTY.withDate(calendarday)*».

+ fromDate(year: int, month: int, date: int): FEMDatetime

Diese Methode gibt eine Zeitangabe mit dem gegebenen Datum zurück und ist eine Abkürzung für «*FEE_Datetime.EMPTY.withDate(year, month, date)*».

+ fromTime(daymillis: int): FEMDatetime

Diese Methode gibt eine Zeitangabe mit der Uhrzeit zu den gegebenen Tagesmillis zurück und ist eine Abkürzung für «*FEE_Datetime.EMPTY-.withTime(daymillis)*».

+ fromTime(hour: int, minute: int, second: int, millisecond: int): FEMDatetime

Diese Methode gibt eine Zeitangabe mit der gegebenen Uhrzeit zurück und ist eine Abkürzung für «*FEE_Datetime.EMPTY.withTime(hour, minute, second, millisecond)*».

+ fromZone(zone: int): FEMDatetime

Diese Methode gibt eine Zeitangabe mit der gegebenen Zeitzone zurück und ist eine Abkürzung für «*FEE_Datetime.EMPTY.withZone(zone)*».

+ fromZone(zoneHour: int, zoneMinute: int): FEMDatetime

Diese Methode gibt eine Zeitangabe mit der gegebenen Zeitzone zurück und ist eine Abkürzung für «*FEE_Datetime.EMPTY.withZone-(zoneHour, zoneMinute)*».

+ leapOf(year: int): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn das gegebene Jahr «year» ein Schaltjahr im Gregorianischen Kalender ist.

+ lengthOf(month: int, year: int): int

Diese Methode gibt die Länge des gegebenen Monats «month» im gegebenen Jahr «year» zurück («*1…31*»), d.h. die Anzahl der Tage im Monat.

+ lengthOf(month: int, leap: boolean): int

Diese Methode gibt die Länge des gegebenen Monats «month» in einem regulären oder Schaltjahr zurück («*1…31*»), d.h. die Anzahl der Tage im Monat. Wenn «leap» «*true*» ist, wird ein Schaltjahr betrachtet.

+ yeardayOf(calendarday: int): int

Diese Methode gibt den Jahrestag zum gegebenen Kalendertag zurück («*1…366*»).

+ weekdayOf(calendarday: int): int

Diese Methode gibt den Wochentag zum gegebenen Kalendertag zurück («*1…7*»).

+ daymillisOf(hour: int, minute: int, second: int, millisecond: int): int

Diese Methode gibt die Tagesmillis zur gegebenen Uhrzeit zurück («*0…86400000*»).

+ calendardayOf(year: int, month: int, date: int): int

Diese Methode gibt den Kalendertag zum gegebenen Datum zurück («*0…3074323*»).

+ FEMDatetime(value: long)

Dieser Konstruktor initialisiert die interne Darstellung.

+ data(): FEMDatetime

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMDatetime>

Diese Methode «*TYPE*» zurück.

+ value(): long

Diese Methode gibt die interne Darstellung der Zeitangabe zurück. Die 64 Bit von MBS zum LSB sind:

«yearValue» - 14 Bit, «monthValue» - 4 Bit, «minuteValue» - 6 Bit, «secondValue» - 6 Bit, «hasDate» - 1 Bit, «hasTime» - 1 Bit, «hasZone» - 1 Bit, «zoneValue» - 11 Bit, «dateValue» - 5 Bit, «hourValue» - 5 Bit, «millisecondValue» - 10 Bit

+ yearValue(): int

Diese Methode gibt das Jahr zurück. Sie löst eine «*IllegalStateException*» aus, wenn diese Zeitangabe kein Datum besitzt.

+ dateValue(): int

Diese Methode gibt den Tag zurück. Sie löst eine «*IllegalStateException*» aus, wenn diese Zeitangabe kein Datum besitzt.

+ monthValue(): int

Diese Methode gibt den Monat zurück. Sie löst eine «*IllegalStateException*» aus, wenn diese Zeitangabe kein Datum besitzt.

`+ hourValue(): int`

Diese Methode gibt die Stunde zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe keine Uhrzeit besitzt.

`+ minuteValue(): int`

Diese Methode gibt die Minute zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe keine Uhrzeit besitzt.

`+ secondValue(): int`

Diese Methode gibt die Sekunde zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe keine Uhrzeit besitzt.

`+ millisecondValue(): int`

Diese Methode gibt die Millisekunde zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe keine Uhrzeit besitzt.

`+ zoneValue(): int`

Diese Methode gibt die Zeitzonenverschiebung zur UTC in Minuten zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe keine Zeitzone besitzt.

`+ hasDate(): boolean`

Diese Methode gibt nur dann «*`true`*» zurück, wenn diese Zeitangabe ein Datum besitzt.

`+ hasTime(): boolean`

Diese Methode gibt nur dann «*`true`*» zurück, wenn diese Zeitangabe eine Uhrzeit besitzt.

`+ hasZone(): boolean`

Diese Methode gibt nur dann «*`true`*» zurück, wenn diese Zeitangabe eine Zeitzone besitzt.

`+ yeardayValue(): int`

Diese Methode gibt den Jahrestag zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe kein Datum besitzt.

`+ weekdayValue(): int`

Diese Methode gibt den Wochentag zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe kein Datum besitzt.

`+ daymillisValue(): int`

Diese Methode gibt die Tagesmillis zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe kein Datum besitzt.

`+ calendardayValue(): int`

Diese Methode gibt den Kalendertag zurück. Sie löst eine «*`IllegalStateException`*» aus, wenn diese Zeitangabe kein Datum besitzt.

`+ withDate(calendarday: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit dem Datum zum gegebenen Kalendertag zurück.

`+ withDate(year: int, month: int, date: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit dem gegebenen Datum zurück.

`+ withDate(calendar: Calendar): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit dem Datum des gegebenen «*`Calendar`*» zurück. Die gelieferte Zeitangabe besitzt nur dann ein Datum, wenn am gegebenen «*`Calendar`*» die Felder «*`Calendar.YEAR`*», «*`Calendar.MONTH`*» und «*`Calendar.DATE`*» definiert sind. Andernfalls hat die gelieferte Zeitangabe kein Datum.

`+ withDate(datetime: FEMDatetime): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit dem Datum der gegebenen Zeitangabe zurück. Wenn die gegebene Zeitangabe kein Datum besitzt, hat die gelieferte Zeitangabe auch kein Datum.

`+ withTime(daymillis: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der Uhrzeit zu den gegebenen Tagesmillis zurück.

`+ withTime(hour: int, minute: int, second: int, millisecond: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der gegebenen Uhrzeit zurück.

`+ withTime(calendar: Calendar): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der Uhrzeit des gegebenen «*`Calendar`*» zurück. Die gelieferte Zeitangabe besitzt nur dann eine Uhrzeit, wenn am gegebenen «*`Calendar`*» die Felder «*`Calendar.HOUR_OF_DAY`*», «*`Calendar.MINUTE`*» und «*`Calendar.SECOND`*» definiert sind. Andernfalls hat die gelieferte Zeitangabe keine Uhrzeit.

`+ withTime(datetime: FEMDatetime): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der Uhrzeit der gegebenen Zeitangabe zurück. Wenn die gegebene Zeitangabe keine Uhrzeit besitzt, hat die gelieferte Zeitangabe auch keine Uhrzeit.

`+ withZone(zone: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der gegebenen Zeitzone zurück. Wenn diese Zeitangabe bereits eine Zeitzone besitzt, werden Uhrzeit und Datum sofern vorhanden entsprechend angepasst.

`+ withZone(zoneHour: int, zoneMinute: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der gegebenen Zeitzone zurück. Wenn diese Zeitangabe bereits eine Zeitzone besitzt, werden Uhrzeit und Datum sofern vorhanden entsprechend angepasst.

`+ withZone(calendar: Calendar): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der Zeitzone des gegebenen «*`Calendar`*» zurück. Wenn diese Zeitangabe bereits eine Zeitzone besitzt, werden Uhrzeit und Datum sofern vorhanden entsprechend angepasst. Wenn am gegebenen «*`Calendar`*» das Feld «*`Calendar.ZONE_OFFSET`*» undefiniert ist, hat die gelieferte Zeitangabe keine Zeitzone.

`+ withZone(datetime: FEMDatetime): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit der Zeitzone der gegebenen Zeitangabe zurück. Wenn diese Zeitangabe bereits eine Zeitzone besitzt, werden Uhrzeit und Datum sofern vorhanden entsprechend angepasst. Wenn die gegebene Zeitangabe keine Zeitzone besitzt, hat die gelieferte Zeitangabe auch keine Zeitzone.

`+ withoutDate(): FEMDatetime`

Diese Methode gibt diese Zeitangabe ohne Datum zurück.

`+ withoutTime(): FEMDatetime`

Diese Methode gibt diese Zeitangabe ohne Uhrzeit zurück.

`+ withoutZone(): FEMDatetime`

Diese Methode gibt diese Zeitangabe ohne Zeitzone zurück.

`+ move(duration: FEMDuration): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit verschobenem Zeitpunkt zurück. Sie ist eine Abkürzung für «*`this.moveDate(duration).moveTime(duration)`*».

`+ moveDate(years: int, months: int, days: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit verschobenem Datum zurück. Die Verschiebung erfolgt gemäß «XML Schema Part 2: §E Adding durations to dateTimes»:

(1) Jahr und Monat werden gemäß der gegebenen Anzahl an Jahren «years» («*`-8417…8417`*») und Monaten «months» («*`-101015…101015`*») verschoben. (2) Der Tag wird gemäß dem ermittelten Jahr und Monat korrigiert, sodass der Tag nicht größer als die Anzahl der Tage im Monat ist. (3) Der Tag wird gemäß der gegebenen Anzahl an Tagen «days» («*`-3074323…3074323`*») verschoben. Dadurch können sich Jahr und Monat nochmals ändern.

`+ moveDate(duration: FEMDuration): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit verschobenem Datum zurück.

`+ moveTime(hours: int, minutes: int, seconds: int, milliseconds: int): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit verschobener Uhrzeit zurück. Wenn die Zeitangabe ein Datum besitzt, wird dieses falls nötig ebenfalls verschoben. Die Verschiebung erfolgt gemäß «XML Schema Part 2: §E Adding durations to dateTimes».

`+ moveTime(hours: int, minutes: long, seconds: long, milliseconds: long): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit verschobener Uhrzeit zurück. Wenn die Zeitangabe ein Datum besitzt, wird dieses falls nötig ebenfalls verschoben. Die Verschiebung erfolgt gemäß «XML Schema Part 2: §E Adding durations to dateTimes».

`+ moveTime(duration: FEMDuration): FEMDatetime`

Diese Methode gibt diese Zeitangabe mit verschobener Uhrzeit zurück.

+ moveZone(hours: int, minutes: int): FEMDatetime

Diese Methode gibt diese Zeitangabe mit verschobener Zeitzone zurück. Wenn die Zeitangabe eine Uhrzeit besitzt, wird diese falls nötig ebenfalls verschoben.

+ hash(): int

Diese Methode gibt den Streuwert zurück.

+ equals(that: FEMDatetime): boolean

Diese Methode gibt nur dann «true» zurück, wenn diese Zeitangabe effektiv gleich der gegebenen ist.

+ compare(that: FEMDatetime, undefined: int): int

Diese Methode gibt «*-1*», «*0*» bzw. «*+1*» zurück, wenn diese Zeitangabe früher, gleich bzw. später als die gegebene Zeitangabe ist. Wenn die Zeitangaben nicht vergleichbar sind, wird «undefined» geliefert. Der Vergleich erfolgt für Zeitangaben mit Datum und/oder Uhrzeit gemäß «XML Schema Part 2: 3.2.7.3 Order relation on dateTime»:

(1) Verschieben beider Zeitangaben auf Zeitzone 00:00. Zeitangaben mit Datum und ohne Uhrzeit werden hierbei so behandelt, als hätten sie die Uhrzeit «00:00:00». Damit sinkt der «*FEMDatetime.calendardayValue()*» nur dann um «1», wenn der «*FEMDatetime.zoneValue()*» größer als «0» ist. (2) Wenn nur eine der Zeitangaben ein Datum besitzt, wird «undefined» geliefert. (3) Wenn beide ein Datum besitzen und die Differenz von «*FEMDatetime.calendardayValue()*» ungleich «0» ist, wird das Vorzeichen dieser Differenz geliefert. (4) Wenn nur eine der Zeitangaben eine Uhrzeit besitzt, wird «undefined» geliefert. (5) Wenn beide eine Uhrzeit besitzen, wird das Vorzeichen der Differenz von «*FEMDatetime.daymillisValue()*» geliefert. (6) Andernfalls wird «0» geliefert.

Der Vergleich für Zeitangaben ohne Datum und ohne Uhrzeit erfolgt über folgende Schritte:

(1) Wenn nur eine der Zeitangaben eine Zeitzone besitzt, wird «undefined» geliefert. (2) Wenn beide eine Zeitzone besitzen, wird das Vorzeichen der Differenz von «*FEMDatetime.zoneValue()*» geliefert. (3) Andernfalls wird «0» geliefert.

+ toCalendar(): GregorianCalendar

Diese Methode gibt diese Zeitangabe als «*Calendar*» zurück.

+ toString(): String

Diese Methode gibt die Textdarstellung dieser Zeitangabe zurück. Diese Textdarstellung entspricht der des Datentyps «*xs:dateTime*» aus «XML Schema Part 2: Datatypes Second Edition», beschränkt auf maximal drei Nachkommastellen für die Sekunden.

2.17 FEMDuration

+ bee.creative.fem.FEMDuration: FEMBaseValue, Comparable<FEMDuration>

Diese Klasse implementiert eine unveränderliche Zeitspanne aus Jahren, Monaten, Tagen, Stunden, Minuten, Sekunden und Millisekunden. Intern wird die Zeitspanne als ein «*long*» dargestellt.

Gesamtanzahl der Monate

Der relative Anteil einer Zeitspanne ist durch die Jahre und Monate gegeben, welche zur Gesamtanzahl an Monaten zusammengefasst werden können. Diese Gesamtanzahl liegt im Bereich «*-101006…101006*». Wenn ein Datum um diese Monatsanzahl verschoben werden soll, entscheidet erst dieses Datum über die konkrete Anzahl an Tagen, um die das Datum verschoben wird. Man kann zu einer Monatsanzahl jedoch ermitteln, zu wie vielen Tagen diese minimal bzw. maximal führen kann.

Gesamtanzahl der Millisekunden

Der absolute Anteil einer Zeitspanne ist durch Tage, Stunden, Minuten, Sekunden und Millisekunden gegeben, welche zur Gesamtanzahl an Millisekunden zusammengefasst werden können. Diese Gesamtanzahl liegt im Bereich «*-265621593600000…265621593600000*».

+ ID: int

Dieses Feld speichert den Identifikator von «*TYPE*».

+ TYPE: FEMType<FEMDuration>

Dieses Feld speichert den Datentyp.

+ EMPTY: FEMDuration

Dieses Feld speichert die leere Zeitspanne, deren Komponenten «*0*» sind.

+ from(string: String): FEMDuration

Diese Methode gibt eine neue Zeitspanne mit dem in der gegebenen Zeichenkette kodierten Wert zurück. Das Format der Zeichenkette entspricht dem der Textdarstellung.

+ from(durationmonths: int, durationmillis: long): FEMDuration

Diese Methode gibt eine Zeitspanne mit den gegebenen Gesamtanzahlen an Monaten und Millisekunden zurück.

+ from(years: int, months: int, days: int, hours: int, minutes: long, seconds: long, milliseconds: long): FEMDuration

Diese Methode gibt eine Zeitspanne mit den Gesamtanzahlen an Monaten und Millisekunden zurück, die sich aus den gegebenen Anzahlen ergeben.

+ from(value: FEMValue): FEMDuration

Diese Methode ist eine Abkürzung für «*FEMContext.DEFAULT().dataFrom(value, TYPE)*».

+ from(value: FEMValue, context: FEMContext): FEMDuration

Diese Methode ist eine Abkürzung für «*context.dataFrom(value, TYPE)*».

+ between(datetime1: FEMDatetime, datetime2: FEMDatetime): FEMDuration

Diese Methode gibt die Zeitspanne zwischen den gegebenen Zeitangaben in Zeitzone «*00:00*» zurück. Wenn nur eine der Zeitangaben ein Datum bzw. eine Uhrzeit besitzt, wird eine «*IllegalArgumentException*» ausgelöst.

+ minLengthOf(months: int): int

Diese Methode gibt die minimale Anzahl an Tagen zurück, die durch die gegebene Anzahl an Monaten ausgedrückt werden kann.

+ maxLengthOf(months: int): int

Diese Methode gibt die maximale Anzahl an Tagen zurück, die durch die gegebene Anzahl an Monaten ausgedrückt werden kann.

+ durationmillisOf(days: int, hours: int, minutes: long, seconds: long, milliseconds: long): long

Diese Methode gibt die Gesamtanzahl der Millisekunden zurück, die sich aus den gegebenen Anzahlen ergeben («*-265621593600000…265621593600000*»).

+ durationmonthsOf(years: int, months: int): int

Diese Methode gibt die Gesamtanzahl der Monate zurück, die sich aus den gegebenen Anzahlen ergeben («*-101006…101006*»).

+ FEMDuration(value: long)

Dieser Konstruktor initialisiert die interne Darstellung der Zeitspanne.

+ data(): FEMDuration

Diese Methode gibt «*this*» zurück.

+ type(): FEMType<FEMDuration>

Diese Methode «*TYPE*» zurück.

+ value(): long

Diese Methode gibt die interne Darstellung der Zeitspanne zurück. Die 64 Bit von MBS zum LSB sind:

«yearsValue» - 14 Bit, «signValue» - 1 Bit, «hoursValue» - 5 Bit, «minutesValue» - 6 Bit, «secondsValue» - 6 Bit, «daysValue» - 18 Bit, «monthsValue» - 4 Bit, «millisecondsValue» - 10 Bit

+ signValue(): int

Diese Methode gibt das Vorzeichen dieser Zeitspanne zurück. Das Vorzeichen ist «*-1*», «*0*» oder «*+1*», wenn alle Komponenten der Zeitspanne kleiner als, gleich bzw. größer als «*0*» sind.

+ yearsValue(): int

Diese Methode gibt die Anzahl der Jahre zurück.

+ monthsValue(): int

Diese Methode gibt die Anzahl der Monate zurück.

+ daysValue(): int

Diese Methode gibt die Anzahl der Tage zurück.

+ hoursValue(): int

Diese Methode gibt die Anzahl der Stunden zurück.

`+ minutesValue(): int`

Diese Methode gibt die Anzahl der Minuten zurück.

`+ secondsValue(): int`

Diese Methode gibt die Anzahl der Sekunden zurück.

`+ millisecondsValue(): int`

Diese Methode gibt die Anzahl der Millisekunden zurück.

`+ durationmillisValue(): long`

Diese Methode gibt die Gesamtanzahl der Millisekunden zurück.

`+ durationmonthsValue(): int`

Diese Methode gibt die Gesamtanzahl der Monate zurück.

`+ negate(): FEMDuration`

Diese Methode gibt diese Zeitspanne mit umgekehrten Vorzeichen zurück.

`+ move(durationmonths: int, durationmillis: long): FEMDuration`

Diese Methode gibt diese Zeitspanne verschoben um die gegebenen Gesamtanzahlen an Monate und Millisekunden zurück.

`+ move(years: int, months: int, days: int, hours: int, minutes: long, seconds: long, milliseconds: long): FEMDuration`

Diese Methode gibt diese Zeitspanne verschoben um die Gesamtanzahlen an Monaten und Millisekunden zurück, die sich aus den gegebenen Anzahlen ergeben.

`+ move(duration: FEMDuration): FEMDuration`

Diese Methode gibt diese Zeitspanne verschoben um die Gesamtanzahlen an Monate und Millisekunden der gegebenen Zeitspanne zurück.

`+ hash(): int`

Diese Methode gibt den Streuwert zurück.

`+ equals(that: FEMDuration): boolean`

Diese Methode gibt nur dann «*`true`*» zurück, wenn diese Zeitspanne effektiv gleich der gegebenen ist.

`+ compare(that: FEMDuration, undefined: int): int`

Diese Methode gibt «*`-1`*», «*`0`*» bzw. «*`+1`*» zurück, wenn diese Zeitspanne kürzer, gleich bzw. länger als die gegebene Zeitspanne ist. Wenn die Zeitspannen nicht vergleichbar sind, wird «undefined» geliefert.

`+ toString(): String`

Diese Methode gibt die Textdarstellung dieser Zeitspanne zurück. Diese Textdarstellung entspricht der des Datentyps «*`xsd:duration`*» aus «XML Schema Part 2: Datatypes Second Edition», beschränkt auf maximal drei Nachkommastellen für die Sekunden.

3 FEM-Funktionen

3.1 FEMBaseFunction

+ bee.creative.fem.FEMBaseFunction: FEMFunction

Diese Klasse implementiert eine abstrakte Funktion mit einigen Methoden zur Umwandlung in einen Wert oder Funktionsaufruf.

+ toValue(): FEMHandler

Diese Methode gibt diese Funktion als «*FEMHandler*» zurück.

+ withParams(params: FEMFunction[]): FEMInvokeFunction

Diese Methode gibt eine neue «*FEMInvokeFunction*» zurück, welche diese Funktion direkt mit den gegebenen Parameterfunktionen aufruft.

3.2 FEMClosureFunction

+ bee.creative.fem.FEMClosureFunction: FEMBaseFunction

Diese Klasse implementiert eine Funktion, welche die zusätzlichen Parameterwerte des Stapelrahmens an eine gegebene Funktion bindet und diese gebundene Funktion anschließend als «*FEMHandler*» liefert.

+ from(function: FEMFunction): FEMClosureFunction

Diese Methode gibt die gegebene Funktion als «*FEMClosureFunction*» zurück. Wenn diese bereits eine «*FEMClosureFunction*» ist, wird sie unverändert geliefert.

+ FEMClosureFunction(function: FEMFunction)

Dieser Konstruktor initialisiert die Funktion, an welchen in «*FEMClosureFunction.invoke(FEMFrame)*» die zusätzlichen Parameterwerte des Stapelrahmens gebunden werden.

+ FEMClosureFunction(frame: FEMFrame, function: FEMFunction)

Dieser Konstruktor initialisiert den Stapelrahmen sowie die gebundene Funktion und sollte nur von «*FEMClosureFunction.invoke(FEMFrame)*» genutzt werden. In der so erzeugten Funktion delegiert die «*FEMClosureFunction.invoke(FEMFrame)*»-Methode die zugesicherten Parameterwerte des ihr übergebenen Stapelrahmens zusammen mit den zusätzlichen Parameterwerten des gebundenen Stapelrahmens an die gegebene Funktion und liefert deren Ergebniswert.

+ invoke(frame: FEMFrame): FEMValue

Diese Methode führt Berechnungen mit dem gegebenen Stapelrahmen durch und gibt den ermittelten Ergebniswert zurück.

Wenn diese Funktion über «*FEMClosureFunction(FEMFunction)*» erzeugt wurde, entspricht der Ergebniswert:
«*new FEMClosureFunction(frame, this.function()).toValue()*».
Damit werden der gegebene Stapelrahmen an die Funktion «*function()*» gebunden und als «*FEMHandler*» geliefert.

Wenn sie dagegen über «*FEMClosureFunction(FEMFrame, FEMFunction)*» erzeugt wurde, entspricht der Ergebniswert:
«*this.function().invoke(this.frame().withParams(frame.params()))*».
Damit werden die gebundene Funktion mit den zugesicherten Parameterwerten des gegebenen sowie den zusätzlichen Parameterwerten des gebundenen Stapelrahmens ausgewertet und der so ermittelte Ergebniswert geliefert.

+ frame(): FEMFrame

Diese Methode gibt die gebundene Stapelrahmen oder «*null*» zurück. Der Stapelrahmen ist «*null*», wenn diese «*FEMClosureFunction*» über dem Konstruktor «*FEMClosureFunction(FEMFunction)*» erzeugt wurde.

+ function(): FEMFunction

Diese Methode gibt die auszuwertende Funktion zurück.

3.3 FEMInvokeFunction

+ bee.creative.fem.FEMInvokeFunction: FEMBaseFunction, TracerInput

Diese Klasse implementiert eine Funktion, die den Aufruf einer gegebenen Funktion mit den Ergebniswerten mehrerer gegebener Parameterfunktionen berechnet.

+ CALL: FEMBaseFunction

Dieses Feld speichert eine Funktion mit der Signatur «*(method: FEMFunction, params: FEMArray): FEMValue*», deren Ergebniswert via «*method(params[0], params[1], ...)*» ermittelt wird, d.h. über den Aufruf der als ersten Parameter gegeben Funktion mit den im zweiten Parameter gegebenen Parameterwertliste.

+ APPLY: FEMBaseFunction

Dieses Feld speichert eine Funktion mit der Signatur «*(param1, ..., paramN: FEMValue, method: FEMFunction): FEMValue*», deren Ergebniswert via «*method(param1, ..., paramN)*» ermittelt wird, d.h. über den Aufruf der als letzten Parameter gegeben Funktion mit den davor liegenden Parametern.

+ from(function: FEMFunction, direct: boolean, params: FEMFunction[]): FEMInvokeFunction

Diese Methode gibt eine neue «*FEMInvokeFunction*» mit den gegebenen Eigenschaften zurück (siehe Konstruktor).

+ FEMInvokeFunction(function: FEMFunction, direct: boolean, params: FEMFunction[])

Dieser Konstruktor initialisiert die aufzurufende Funktion, die Aufrufart und die Parameterfunktionen.

Wenn die Aufrufart «direct» «*true*» ist, wird die aufzurufende Funktion direkt mit den Ergebnissen der Parameterfunktionen ausgewertet. Andernfalls wird diese zuerst mit dem gegebenen Stapelrahmen zu einer Funktion ausgewertet, welche dann mit den Ergebnissen der Parameterfunktionen ausgewertet wird.

+ invoke(frame: FEMFrame): FEMValue

Diese Methode führt Berechnungen mit dem gegebenen Stapelrahmen durch und gibt den ermittelten Ergebniswert zurück.

Bei der direkten Aufrufart, d.h. «*direct() == true*» entspricht der Ergebniswert:
«*this.function().invoke(frame.newFrame(this.params())*».
Damit wird die aufzurufende Funktion direkt mit den Parameterfunktionen aufgerufen. Andernfalls entspricht der Ergebniswert:

«*FEMHandler.from(this.function().invoke(frame), frame.context()).value().invoke(frame.newFrame(this.params())*».
Damit wird die aufzurufende Funktion mit dem gegebenen Stapelrahmen in eine Funktion ausgewertet, die anschließend mit den Parameterfunktionen aufgerufen wird.

+ direct(): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn die aufzurufende Funktion direkt mit den Ergebnissen der Parameterfunktionen aufgerufen wird. Andernfalls wird die aufzurufende Funktion mit den in «*FEMInvokeFunction.invoke(FEMFrame)*» gegebenen Stapelrahmen zu einer Funktion ausgewertet, welche dann mit den Ergebnissen der Parameterfunktionen aufgerufen wird.

+ params(): FEMFunction[]

Diese Methode gibt eine Kopie der Parameterfunktionen zurück.

+ function(): FEMFunction

Diese Methode gibt die aufzurufende Funktion zurück.

+ toResult(): FEMInvokeFunction

Diese Methode gibt eine zu dieser Funktion gleichwertige «*FEMInvokeFunction*» zurück, bei welcher die aufzurufende Funktion «*function()*» sowie jede Parameterfunktion in «*params()*» in eine «*FEMResultFunction*» konvertiert wurde.

3.4 FEMNativeFunction

+ bee.creative.fem.FEMNativeFunction: FEMBaseFunction

Diese Funktion kann zum Lesen und Schreiben von nativen Datenfeldern sowie zum Aufrufen von nativen Methoden und nativen Konstruktoren eingesetzt werden. Der dieser Funktion zugrundeliegende «*Member*» kann hierbei als «*Field*», «*Method*» oder «*Constructor*» gegeben sein.

Datenfelder

Native Funktionen zu klassengebundenen Datenfeldern nutzen zum Lesen die Signatur «*(): FEMNative*» und liefern den Ergebniswert «*new FEMNative(this.member().get(null))*». Zum Schreiben verwenden sie dagegen die Signatur «*(value: FEMNative): FEMNative*», führen «*this.member().set(null, value.data())*» aus und liefern «*FEMNative.NULL*».

Analog dazu nutzen die Funktionen zu instanzgebundenen Datenfeldern zum Lesen die Signatur «*(object: FEMNative): FEMNative*» und liefern den Ergebniswert «*new FEMNative(this.member().get(object.data()))*». Zum Schreiben verwenden sie dann die Signatur «*(object, value: FEMNative): FEMNative*», führen «*this.member().set(object.data(), value.data())*» aus und liefern ebenfalls «*FEMNative.NULL*».

Methoden

Native Funktionen zu klassengebundenen Methoden haben die Signatur «*(param1, …, paramN: FEMNative): FEMNative*» und liefern den Ergebniswert «*new FEMNative(this.member().invoke(null, param1.data(), …, paramN.data()))*».

Analog dazu haben die Funktionen zu instanzgebundenen Methoden die Signatur «*(object, param1, …, paramN: FEMNative): FEMNative*» und liefern den Ergebniswert «*new FEMNative(this.member().invoke(object.data(), param1.data(), …, paramN.data()))*».

Konstruktoren

Native Funktionen zu Konstruktoren haben die Signatur «*(param1, …, paramN: FEMNative): FEMNative*» und liefern den Ergebniswert «*new FEMNative(this.member().newInstance(param1.data(), …, paramN.data()))*».

+ from(memberPath: String): FEMFunction

Diese Methode gibt die native Funktion zur gegebenen Pfadangabe zurück.

Die Pfadangabe kodiert hierbei eine Funktion, die eine Klasse liefert, an eine Methode bzw. einen Konstruktor delegiert oder ein Datenfeld liest bzw. schreibt. Die folgenden Pfadangaben werden unterstützt:

Die Pfadangabe «*"CLASS_PATH.class"*» ergibt Funktion:
«*FEMNative.from(CLASS_PATH.class)*».

Die Pfadangabe «*"CLASS_PATH.FIELD_NAME"*» ergibt Funktion:
«*FEMNative.from(CLASS_PATH.class.getDeclaredField("FIELD_NAME"))*».

Die Pfadangabe «*"CLASS_PATH.new(TYPE_1,...,TYPE_N)"*» ergibt Funktion:
«*FEMNative.from(CLASS_PATH.class.getDeclaredConstructor(TYPE_1.class, ..., TYPE_N.class))*».

Die Pfadangabe «*"CLASS_PATH.METHOD_NAME(TYPE1_1,...,TYPE_N)"*» ergibt Funktion:
«*FEMNative.from(CLASS_PATH.class.getDeclaredMethod("METHOD_NAME", TYPE_1.class, ..., TYPE_N.class))*».

+ from(field: Field): FEMNativeFunction

Diese Methode gibt eine Funktion zurück, mit welcher der Wert des gegebenen Datenfelds gelesen sowie geschrieben werden kann.

+ from(method: Method): FEMNativeFunction

Diese Methode gibt eine Funktion zurück, die an die gegebene Methode delegiert.

+ from(constructor: Constructor<?>): FEMNativeFunction

Diese Methode gibt eine Funktion zurück, die an den gegebenen Konstruktor delegiert.

+ member(): Member

Diese Methode gibt den «*Member*» zurück, auf den sich «*invoke(FEMFrame)*» bezieht.

+ isStatic(): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn der Member statisch ist.

+ isField(): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn der Member ein «*Field*» ist.

+ isMethod(): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn der Member eine «*Method*» ist.

+ isConstructor(): boolean

Diese Methode gibt nur dann «*true*» zurück, wenn der Member ein «*Constructor*» ist.

3.5 FEMParamFunction

+ bee.creative.fem.FEMParamFunction: FEMBaseFunction

Diese Klasse implementiert eine projizierende Funktion, deren Ergebniswert einem der Parameterwerte des Stapelrahmens entspricht.

+ ITEM: FEMBaseFunction

Dieses Feld speichert eine Funktion mit der Signatur «*(index: FEMInteger): FEMValue*», deren Ergebniswert dem «index»-ten Parameterwert des Stapelrahmens entspricht.

+ COPY: FEMBaseFunction

Dieses Feld speichert eine Funktion, die eine Kopie der Parameterwerte des Stapelrahmens «frame» liefert, d.h. «*frame.params().result-(true)*».

+ VIEW: FEMBaseFunction

Dieses Feld speichert eine Funktion, die eine Sicht auf die Parameterwerte des Stapelrahmens «frame» liefert, d.h. «*frame.params()*».

`+ from(index: int): FEMParamFunction`

Diese Methode gibt eine Funktion zurück, welche den «index»-ten Parameterwert des Stapelrahmens als Ergebniswert liefert.

`+ FEMParamFunction(index: int)`

Dieser Konstruktor initialisiert den Index des Parameterwerts.

`+ invoke(frame: FEMFrame): FEMValue`

Der Ergebniswert entspricht «*`frame.get(this.index())`*».

`+ index(): int`

Diese Methode gibt den Index des Parameterwerts zurück.

3.6 FEMProxyFunction

`+ bee.creative.fem.FEMProxyFunction: FEMBaseFunction`

Diese Klasse implementiert den benannten Platzhalter einer Funktion, dessen «*`invoke(FEMFrame)`*»-Methode an eine gegebene Funktion delegiert.

`+ from(name: String): FEMProxyFunction`

Diese Methode gibt eine neue Platzhalterfunktion mit dem gegebenen Namen zurück.

`+ FEMProxyFunction(name: String)`

Dieser Konstruktor initialisiert den Namen.

`+ set(function: FEMFunction): void`

Diese Methode setzt die in «*`invoke(FEMFrame)`*» aufzurufende Funktion.

`+ name(): String`

Diese Methode gibt den Namen des Platzhalters zurück.

`+ function(): FEMFunction`

Diese Methode gibt die Funktion zurück, die in «*`invoke(FEMFrame)`*» aufgerufen wird. Diese ist «*`null`*», wenn «*`set(FEMFunction)`*» noch nicht aufgerufen wurde.

3.7 FEMResultFunction

`+ bee.creative.fem.FEMResultFunction: FEMBaseFunction`

Diese Klasse implementiert eine Funktion mit call-by-reference-Semantik, deren Ergebniswert ein FEMResult ist.

`+ from(function: FEMFunction): FEMResultFunction`

Diese Methode gibt die gegebene Funktion als «*`FEMResultFunction`*» zurück. Wenn sie bereits eine «FEMResultFunction» ist, wird sie unverändert zurückgegeben

`+ FEMResultFunction(function: FEMFunction)`

Dieser Konstruktor initialisiert die auszuwertende Funktion, die in FEMResultFunction.invoke(FEMFrame) zur Erzeugung eines FEMResult genutzt wird.

`+ function(): FEMFunction`

Diese Methode gibt die auszuwertende Funktion zurück.

`+ invoke(frame: FEMFrame): FEMResult`

Diese Methode gibt den Ergebniswert «*`FEMResult.from(frame, this.function())`*» zurück.