EDBConnectInspector

Inherits From: IBInspector

Declared In: EDBConnectInspector.h

Class Description

EDBConnectInspector ist, wie der Name schon sagt, ein Connection Inspektor, der den originalen Inspektor des InterfaceBuilders funktional erweitert. EDBConnectInspector erlaubt, von einem Outlet aus zu mehreren Zielen Verbindungen zu ziehen und weiters beliebige Outlets wie Targets aussehen zu lassen, d.h. mit Pfeil und der Mðglichkeit, Actions einzugeben.

Der einzige sonstige Unterschied zum Standardinspektor liegt darin begröndet, daû der Zugriff auf Outlets und Actions, so wie sie im InterfaceBuilder editiert werden kðnnen, in der derzeitigen Betriebssystemversion NS3.0 nicht dokumentiert und dadurch nicht mðglich ist. Das Protokoll EDBConnections definiert daher eine Methode, um die Outlets eines Objekts zu erhalten (sh. EDBConnections und EDBOutlet). Actions dagegen beziehen sich auf das Ziel einer Verbindung, und nur InterfaceBuilder selbst weiû die Actions aller anderen Objekte. Aus diesem Grund ist es

nur mðglich, den Namen der Action einzugeben statt die gewönschte Action aus einer Liste von Mðglichkeiten auszuwÙhlen.

Folgende Methoden mössen implementiert werden, um EDBConnectInspector verwenden zu kðnnen: getConnectInspectorClassName, welche z.B. "EDBConnectInspector" zuröckliefert und die Methoden aus dem Protokoll EDBConnections.

Zur Laufzeit werden schluûendlich Verbindungen in Aufrufe wie z.B. setDelegate:, wenn das Objekt ein Outlet namens delegate definiert hat, umgewandelt. Wie und in welcher Reihenfolge das funktioniert, ist in der Klassenbeschreibung zu EDBConnector nachzulesen.

EDBConnectInspector stellt Methoden zur Verfögung, um Outlets wÙhrend der Laufzeit von InterfaceBuilder zu verÙndern, hinzuzufögen oder zu entfernen. Siehe +obtainEDBConnectInspector, -outletOfObject:renamedFrom:to:, -outlet:removedFromObject:,

outletAddedToObject:.

Instance Variables

id	**actionMatrix**
id	**actionScrollView**
id	**actionTitleField**
id	**commentField**
id	**connectMatrix**
id	**connectScrollView**
id	**lowerBox**

id	**outletBrowser**
id	**upperBox**
id	***connections**
NXCoord	**minHeight**
List	***outlets**
u_long	**outletCount**
BOOL	**refreshOnRevert**

actionMatrix	User Interface Objekt
actionScrollView	User Interface Objekt
actionTitleField	User Interface Objekt

commentField	User Interface Objekt
connectMatrix	User Interface Objekt
connectScrollView	User Interface Objekt
lowerBox	User Interface Objekt
outletBrowser	User Interface Objekt
upperBox	User Interface Objekt
connections	Liste von allen derzeitigen Connections von diesem Objekt
minHeight	Die Mindesthöhe der Boxes in der SplitView

outlets	Liste aller Outlets des aktiven Objekts
outletCount	Zahl der Outlets
refreshOnRevert	NO, nachdem ein Outlet oder eine Connection angeklickt wurde. Optimierung.

Method Types

Initializing and freeing	- init - free + obtainEDBConnectInspector

Overridden Methods	- ok: - revert: - wantsButtons
Actions	- clickAction: - clickConnection: - clickOutlet: - doubleClickOutlet:
Handling Connections	- getConnections: - displayConnections: - setSequenceNumberOf: - fillTargetMatrixForConnection:andOutlet: - createConnectorForOutlet:withNr:andSel:

Handling Outlets	- selectOutlet - outletAt:inMatrix:setConnected:andDisplay: - indexOfOutlet: - indexOfFirstFreeOutlet - outletIsConnected:
Dynamically Change of Outlets	- outletOfObject:renamedFrom:to: - outlet:removedFromObject: - outletAddedToObject:
Outlets	- addEDBOutlet:at: - removeEDBOutletWithName: - removeEDBOutletAt:

	- outletRenamedFrom:to:
Browser delegate	- browser:fillMatrix:inColumn:

Class Methods

obtainEDBConnectInspector
+ **obtainEDBConnectInspector**

Returniert die einzige Instanz dieser Klasse oder kreiert eine Instanz, falls es noch keine gibt und returniert diese.

Instance Methods

browser:fillMatrix:inColumn:
- (int)**browser:***sender* **fillMatrix:***matrix* **inColumn:**(int)*column*

Föllt den Outletbrowser mit allen Outlets des gerade inspizierten Objekts. Die Outlets werden in der Liste outlets erwartet. Returniert die Anzahl der Outlets.

See also: EDBOutlet

clickAction:
- **clickAction:***sender*

Föhrt nur ein [okButton performClick:self] durch und returniert self.

clickConnection:
- **clickConnection:***sender*

Zeigt die Connection an und selektiert das entsprechende Outlet. Returniert self.

See also: - **displayConnectionBetween:and:** (IB)

clickOutlet:
- **clickOutlet:***sender*

Selektiert den Text im Actionbrowser, falls es sich um ein Target handelt. Wenn es kein Multiuse-Outlet ist und eine Verbindung för dieses Outlet existiert, wird die Verbindung selektiert und angezeigt. Returniert self.

See also: - **isTarget** (EDBOutlet), - **multiUse** (EDBOutlet)

createConnectorForOutlet:withNr:andSel:

- **createConnectorForOutlet:**(NXAtom)*outletName* **withNr:**(int)*outletNr* **andSel:**(const char *)*sel*

Kreiert eine neue Instanz der Klasse EDBConnecotor und setzt deren Argumente entsprechend den Parametern dieser Methode. Source und Destination werden allerdings direkt durch [NXApp

connectSource] und [NXApp connectDestination] ermittelt. Der Returnwert ist die neue Instanz.

See also: EDBConnector

displayConnections:
- **displayConnections:***list*

Föllt die Matrix der Connections entsprechend dem Inhalt der Liste *list*. In *list* sollten alle Connections mit dem inspizierten Objekt als source enthalten sein (alle diese Objekte haben dem Protokoll IBConnectors zu gehorchen). Die Anzeige einer Connection entspricht der des Standardconnectinspectors, d.h. links der Name des Outlets (plus einer Nummer, falls es sich um ein MultiUse Outlet handelt) und rechts entweder der Name des ZielObjekts oder dessen Klasse.

Returniert self.

See also: IBConnectors**,** EDBConnector

doubleClickOutlet:
- **doubleClickOutlet:***sender*

Föhrt die Action des Ok-Buttons aus und returniert self.

fillTargetMatrixForConnection:andOutlet:
- **fillTargetMatrixForConnection:***connection* **andOutlet:**(int)*outletNr*

Setzt den Inhalt des TextFields im Targetbrowser entweder auf den Namen der Action, wenn das Outlet ein Target ist oder auf NULL. Returniert self.

See also: - **isTarget** (EDBOutlet)

free

- **free**

Freed die Resourcen dieser Klasse und setzt die Klassenvariable onlyInspector auf nil, falls self gleich onlyInspector ist. Es ist daher för Unterklassen mðglich, mehrere Instanzen zu erzeugen, ohne den Wert der privaten Klassenvariable zu verÙndern.

See also: - **init**

getConnections:

- **getConnections:***list*

Holt alle Connections mit dem inspizierten Objekt als Source (mittels [[NXApp activeDocument] listConnectors:*list* forSource:object]) und speichert sie in Liste list. Diese Liste wird anschlieûend nach Outletnamen sortiert. Returniert self.

indexOfFirstFreeOutlet

- (int)**indexOfFirstFreeOutlet**

Liefert den Index des ersten Outlets in der Liste outlets, för das entweder keine Connection existiert oder das ein MultiUse-Outlet ist, oder -1 andernfalls.

See also: - **multiUse** (EDBOutlet)

indexOfOutlet:

- (int)**indexOfOutlet:**(NXAtom)*outlet*

Liefert den Index des Outlets mit Namen *outlet* in der Liste outlets oder -1, falls kein solches Outlet existiert.

init

- init

Falls bereits ein Objekt der Klasse EDBConnectInspector existiert und [self class]==[EDBConnectInspector class] ist, so wird die ID des existierenden Objekts zuröckgegeben. Andernfalls wird die neue Instanz initialisiert und self oder nil (wenn EDBConnectInspector.nib nicht gefunden wurde) returniert. D.h. wenn eine Unterklasse [super init] aufruft (oder init nicht implementiert), wird auf jeden Fall eine neue Instanz initialisiert. Es liegt also im Ermessen der Unterklassen, ob mehr als eine Instanz existieren darf.

See also: - **free**, + **obtainEDBConnectInspector**

ok:

- **ok:***sender*

Je nachdem, ob die gerade angezeigte Verbindung bereits existiert oder nicht, wird sie gelðscht bzw. neu erzeugt. Returniert self.

See also: - **createConnectorForOutlet:withNr:andSel:**

outlet:removedFromObject:

- **outlet**:(NXAtom)*name* **removedFromObject:***sender*

Entfernt alle Connections för das Outlet mit Namen *name* und schickt sich selbst -revert:, falls das gerade inspizierte Objekt mit *sender* öbereinstimmt.

outletAddedToObject:

- **outletAddedToObject**:*sender*

Schickt sich selbst die Methode -revert:, falls das gerade inspizierte Objekt mit *sender* öbereinstimmt.

outletAt:inMatrix:setConnected:andDisplay:

- **outletAt**:*cell* **inMatrix**:*matrix* **setConnected:**(BOOL)*flag* **andDisplay:**(BOOL)*disp*

Stattet Outlet *cell* im Outletbrowser mit einer entprechenden Markierung aus, die anzeigt, ob das

Outlet verbunden (*flag*==YES) oder nicht verbunden (*flag*==NO) ist. Retuniert self.

outletIsConnected:

- (BOOL)**outletIsConnected:**(int)*ind*

Returniert YES, wenn das Outlet mit Index *ind* in der Liste outlets verbunden ist und kein MultiUse-Outlet ist. Retuniert NO andernfalls.

See also: - **multiUse** (EDBOutlet)

outletOfObject:renamedFrom:to:

- **outletOfObject:***sender* **renamedFrom:**(NXAtom)*from* **to**:(NXAtom)*to*

Sucht das Outlet mit Namen *form* in der Liste outlets und ersetzt dessen Namen durch *to*. Retuniert self.

See also: - **setOutletName:** (EDBOutlet)

removeSequenceNumberOf:

- **removeSequenceNumberOf:***connector*

Die Sequencenummer ist durchgehend und bestimmt die Reihenfolge der Connections för ein MultiUse-Outlet (entspricht der Reihenfolge, in der die Verbindungen erstellt werden). Diese Methode schlieûst die Löcke in der Nummerierung, die entsteht, wenn eine Connection gelðscht

wird. Returniert self.

See also: - **setSequence:** (EDBConnection)

revert:

- **revert:***sender*

Kopiert alle Connections mit dem inspizierten Objekt als Source in die Liste connections, zeigt die Daten erneut an und selektiert das erste freie Outlet. Returniert [super revert:*sender*].

See also: - **getOutlets** (EDBConnections), - **getConnections:**, - **displayConnections:**, - **selectOutlet**

selectOutlet
- **selectOutlet**

Untersucht, ob InterfaceBuilder gerade eine Connection anzeigt und selektiert entsprechend im Connection- und Outletbrowser, oder selektiert das nÙchste freie Outlet. Returniert self.

setSequenceNumberFor:
- **setSequenceNumberFor:***connector*

Setzt *Connectors* Sequencenummer auf die nÙchste Freie. Returniert self.

See also: - **setSequence:** (EDBConnector)

wantsButtons

- (BOOL)**wantsButtons**

Returniert YES.