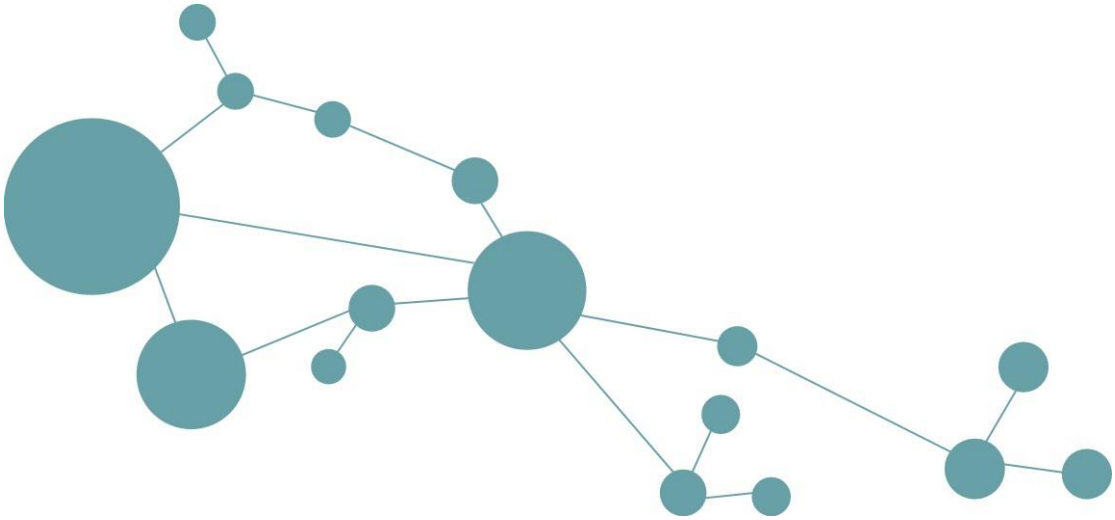


Net-Navigator Dokumentation

i·views

Inhaltsverzeichnis

1 Verzeichnisstruktur

Sämtliche Dateien, die das Net-Navigator-Applet verwendet, müssen in einer speziellen Verzeichnisstruktur abgelegt sein, damit das Programm sie findet.

Den Ausgangspunkt dazu bildet ein Verzeichnis mit dem Namen „**Net-Navigator_Resources**".

Es muss ein Unterverzeichnis von dem Verzeichnis sein, in dem sich die Datei „**NetNavigator.jar**" befindet (bzw. dem Ort, den das Net-Navigator-Applet als Codebase annimmt).

1.1 Net-Navigator_Resources

In diesem Verzeichnis lagert die Konfigurationsdatei „**NetNavigator.dat**", in der sämtliche Einstellungen für das Programm vorgenommen werden können. (Prinzipiell können auch unterschiedliche Konfigurationsdateien bei dem Applet Aufruf durch Parameter angegeben werden.)
Ausserdem kann in diesem Verzeichnis eine Datei „**Volumename_Kategorien.dat**" angelegt werden, die Kategorieninformationen für ein spezielles Volume enthält (Farbe, Kategorienname, etc.).Dabei ist „Volumename" durch den tatsächlichen Namen des Volumes zu ersetzen (z.B. „seruba_Kategorien.dat").

1.2 Net-Navigator_Resources/data/

Soll der Net-Navigator seine Daten nicht über eine Netzwerkverbindung von einer XML-Bridge beziehen, müssen die notwendigen XML-Dateien in diesem Verzeichnis abgelegt werden.

1.3 Net-Navigator_Resources/HelpDocs

In diesem Verzeichnis werden die beiden Dateien **info.htm** sowie **hilfe.htm** untergebracht, die bei einem Klick auf das Info-Icon (bzw. auf das Hilfe-Icon) des Net-Navigators geöffnet werden.

1.4 Net-Navigator_Resources/HelpDocs/gfx

Hier werden alle Bilddateien abgelegt, die von den HTML-Dateien im Verzeichnis '**HelpDocs** verwendet werden.

1.5 Net-Navigator_Resources/eng/

Net-Navigator kann als Applet-Parameter die gewünschte Sprache übergeben werden. In diesem Fall sucht Net-Navigator in einem Unterverzeichnis mit dem entsprechenden Sprachkürzel (z.B. „eng") nach einer Datei namens '**language.dat'**, in der sämtliche Zeichenketten in der

entsprechenden Übersetzung enthalten sein müssen.

Zusätzlich wird auch hier nach der **'_Kategorien.dat'** gesucht.
Parameter, die darin enthalten sind, überschreiben die Parameter aus der allgemeinen _Kategorien.dat, die sich direkt unter Net-Navigator_Resources befindet.
(Es bietet sich an in der sprachspezifischen_Kategorien.dat nur noch die Parameter _Display-Name zu überschreiben).

1.6 Net-Navigator_Resources/eng/IconDocs

In diesem Verzeichnis werden die entsprechend übersetzten Versionen der beiden Dateien **info.htm** sowie **hilfe.htm** untergebracht.

1.7 Net-Navigator_Resources/eng/IconDocs/gfx

Hier werden alle Bilddateien abgelegt, die von den HTML-Dateien im Verzeichnis **eng/IconDocs** verwendet werden.

1.8 Net-Navigator_Resources/Images/

Sämtliche Buttons, Hintergrundgrafiken und Icons die im Net-Navigator dargestellt werden, müssen in diesem Verzeichnis als Bilddatei im GIF- oder JPG-Format vorliegen.

1.9 Net-Navigator_Resources/eng/Images/

Falls eine Sprache über den Applet-Parameter <**language**> spezifiziert worden ist, wird der Net-Navigator zunächst versuchen, die Bildresourcen aus diesem Verzeichnis zu laden.
Erst nachdem festgestellt worden ist, daß sich die entsprechende Bilddatei nicht in diesem Verzeichnis befindet, wird im Images-Verzeichnis, das sich direkt unterhalb von Net-Navigator_Resources befindet nach ihr gesucht.

2 Konfiguration

2.1 Konfigurationsmöglichkeiten

Das Net-Navigator-Applet wertet zunächst Parameter aus, die beim Start übergeben werden. Beim Aufruf des Applets aus einer HTML-Datei kann zum setzen dieser Parameter das PARAM-Tag verwendet werden.

Beispiel:
<APPLET ARCHIVE="NetNavigator.jar" CODE="Graphtool.GraphApplet.class" NAME="nn" WIDTH="1" HEIGHT="1">
<PARAM name="language" value="eng">

<PARAM name="volume" value="Seruba">
</APPLET>

Nachdem das Applet gestartet ist, liesst es die Datei „NetNavigator.dat“ im Unterverzeichnis „Net-Navigator_Resources“ und optional eine Datei „Volume_Kategorien.dat“. „Volume“ muss hierbei durch den Namen des verwendeten Volumes ersetzt werden. In diesen Dateien sind eine Vielzahl von Einstellungen durch key/value-Paare angegeben. Ein key/value-Paar wird immer durch ein Leerzeichen getrennt.

Für den Fall, das Parameter nicht in der Konfigurationsdatei definiert sind, wurden fast alle Parametern Default-Werte zugewiesen. Der Parameter „Volume“ bildet dabei eine Ausnahme. Er muss in jedem Fall entweder als Applet-Parameter oder als Parameter in der „Net-Navigator.dat“ definiert sein. Ist ein Schlüssel sowohl als Applet-Parameter, als auch in der NetNavigator.dat definiert, wird der Wert aus der Konfigurationsdatei verwendet.
Zu allen in der Dokumentation angesprochenen Konfigurationsdateien („NetNavigator.dat“, „Volume_Kategorien.dat“ und „language.dat“) existieren vollständige Beispiele, die in jedem Fall als Vorlage verwendet werden sollten.

Der Applet Parameter "ParamFile" dient zur Angabe der Konfigurationsdateinamens, defaultmäßig ist „NetNavigator.dat“ eingestellt.

Voraussetzung im Wissensnetz - die netNavCategory

Damit der Net-Navigator interpretieren kann, mit welchen Topics er in spezifischer Weise agieren soll, müssen diesen das Attribut "netNavCategory" im Wissensnetz zugewiesen werden. (Das Attribut netNavCategory benötigt dabei zwingende den internen Namen netNavCategory.) Die Benamung des gesetzten Attributs ist frei. Bevor das Attribut z.B. unter den Filterkriterien im Net-Navigator Applet erscheint - also Änderungen auch im Net-Navigator ankommen, muss im Wissensnetz folgende Aktion ausgeführt werden:

Werkzeuge -> www -> Net-Navigator Kategorien

Tip:

Um die Änderungen der Konfiguration im Net-Navigator angezeigt zu bekommen muß der Browser **geschlossen** und wieder geöffnet werden. Ein Reload (F5 drücken) reicht nicht aus!

2.2 Wie definiere ich Farbwerte?

Farbwerte werden in HTML-Schreibweise mit Rot-, Grün- und Blauanteil jeweils als zweistellige Hexadezimalzahlen angegeben.

Beispiel:
BackGround 10FF64

Dieses Beispiel definiert die Hintergrundfarbe mit einem Rot-Anteil von 10, einem Grün-Anteil von 255 und einem Blauanteil von 100.

2.3 Wie definiere ich Schriftarten?

Schriftarten werden in der Form

<**Fontname**>, <**Style**>, <**Groesse**>

angegeben,
wobei Style ein Wert zwischen 0 und 3 sein muss (0=normal, 1=fett, 2=kursiv, 3=fett+kursiv).

Beispiel:
DefaultFont Helvetica, 1, 5

Auf diese Weise wird die Schriftart Arial 5 Punkt Fett als Standardfont gewählt.

2.4 Browser-Einbindung

2.4.1 Einbettung in HTML-Frames

Über den Schlüssel

„FrameEmbedded" (false/true)

kann festgelegt werden, ob Net-Navigator in dem HTML-Frame dargestellt werden soll, der das Java-Applet enthält.
Wird dieser Parameter auf 'false' gesetzt, so wird ein neues Fenster erzeugt, und Net-Navigator darin dargestellt.

2.4.2 Frame zur Inhaltsdarstellung auswählen

Mit dem Parameter „**TargetFrame**" kann ein HTML-Frame ausgewählt werden, in dem Net-Navigator sowohl eigene HTML-Dokumente als auch Web-Verknüpfungen von Knoten anzeigt. Dabei sind die folgenden Werte möglich:

- "**_self**" zeigt Inhalte in dem Fenster und dem Frame an, dass auch das Applet enthält.
- "**_parent**" zeigt den Inhalt im Eltern-Frame des Applet-Frames. Existiert dieser Frame nicht, so gilt das gleiche wie bei „_self".
- "**_top**" zeit den Inhalt im Top-Level-Frame des Applet-Fensters. Befindet sich das Applet bereits im Top-Level-Frame, so gilt das gleich wie bei "_self".
- "**_blank**" zeigt Inhalte in einem neuen, unbenannten Top-Level-Fenster.
- "**name**" zeigt Inhalte im Frame oder Fenster mit dem Namen "name". Existiert kein Fenster oder Frame mit diesem Namen, wird ein neues Fenster mit dem Namen erzeugt und die Inhalt dort angezeit.

2.5 Konfiguration der Datenquelle

Unabhängig von der Datenquelle muss immer der Parameter „**Volume**“ gesetzt werden.
Er gibt an, auf welches Wissensnetz zugegriffen werden soll.

Um festzulegen, welche Knoten beim Start des Net-Navigator geöffnet werden sollen, können über den Applet-Parameter <**StartNodes**> ein oder mehrere Knoten-IDs angegeben werden (getrennt durch ein Semikolon „;“). Diese beiden Parameter können sowohl als Applet-Parameter, als auch durch Eintrag in die Datei „**NetNavigator.dat**“ definiert werden. Wird der Parameter „StartNode“ nicht definiert, so nimmt er automatisch den Wert „default“ an. Durch diesen Wert wird ein im Wissensnetz definierter Default-Knoten geöffnet.
Der Applet Parameter <**ParamFile**> dient zur Angabe der Konfigurationsdateinamens, defaultmäßig ist „NetNavigator.dat“ eingestellt.

Der Schlüssel „**DownloadGraphAtOnce**“ (false/true) bestimmt, ob Net-Navigator seine XML-Daten aus einer Datei von einem lokalen Datenträger beziehen soll.
Ist der Wert dieses Schlüssels „true“, so wird versucht, eine XML-Datei mit dem Namen des Volumes im Unterverzeichnis Net-Navigator_Resources/data zu laden.
Ist dieser Wert auf false gesetzt, wird eine Anfrage an eine URL gesendet, die über den Parameter „**RequestURL**“ (NNRequestURL bei Übergabe als Applet-Parameter) angegeben werden kann. Es muss entweder die Adresse eines Net-Navigator Request-Mappers oder die Adresse der Net-Navigator-XML-Bridge angegeben werden.

Über den Parameter „**ID-Request**“ kann die Art des Requests verändert werden, der beim Öffnen von Knoten an die XML-Bridge gesendet wird. Ist dieser Parameter nicht definiert, so wird als Standardwert „id“ verwendet.

Über den Parameter „**UseShortTags**“ kann festgelegt werden, ob in den XML-Daten kurze (true) oder lange (false) Attributnamen verwendet werden. In der Regel werden lange Attributnamen verwendet.

2.6 Mehrsprachigkeit

Über den Applet-Parameter „**language**“ kann ein Sprachkürzel (z.B. „eng“) an den Net-Navigator übergeben werden.
Ist dies der Fall, versucht Net-Navigator die Datei „**Language.dat**“ im entsprechenden Unterverzeichnis von „**Net-Navigator_Resources**“ (z..B. „Net-Navigator_Resources/eng“) zu laden. In diese Datei können sämtliche von Net-Navigator verwendeten Zeichenketten eingetragen werden. Ausserdem wird bei einem Klick auf das Hilfe- oder Info-Icon die in der Language.dat als URL1 (Hilfe) bzw. URL2 (Info) angegebene URL geöffnet.

2.7 Look & Feel

2.7.1 Fenstergröße und Rand

Breite und Höhe des Net-Navigator Fensters können über die Parameter „**AppletPreferredFrameWidth**“ und „**AppletPreferredFrameHeight**“ eingestellt werden.
Diese Parameter besitzen nur für JAVA-Frames Relevanz.

Das Verhältnis von Rand zu Darstellungsfläche kann prozentual jeweils für die X- und die Y-Achse angegeben werden (**ViewPaintRandXProtz**, **ViewPaintRandYProz**).
Wird ViewPaintRandXProtz auf 40 gesetzt, so verwendet Net-Navigator 60 Prozent der Fen-

sterbreite für die Netzdarstellung, und jeweils 20 Prozent der Fensterbreite für den linken und den rechten Rand (insgesamt 40 Prozent).

Da auf diese Weise der linke und der rechte Rand die gleiche Größe haben, gibt es noch den Parameter „**ViewPaintProtzRandXRechts**“ (für die Y-Achse „ViewPaintProtzRandYUnten“), über den sich der rechte Rand separat einstellen lässt.

Über die Parameter „**ViewPaintMinRandX**“ und „**ViewPaintMinRandXRechts**“ (für die Y-Achse „ViewPaintMinRandY“ und „ViewPaintMinRandYUnten“) lässt sich außerdem noch ein absoluter minimaler Rand in Pixeln einstellen.

2.7.2 Fensterhintergrund

Die Hintergrundfarbe des Fensters wird über den Parameter „**BackGround**“ eingestellt.
Es kann auch eine GIF-Bilddatei als Fensterhintergrund verwendet werden
(Parameter „**ImageName1**“).

Man sollte den Parameter „**DarkBackground**“ entsprechend dem Hintergrundbild oder der Hintergrundfarbe setzen. Der Net-Navigator verwendet diesen Parameter, um selbstständig aktive Farben mit einer guten Kontrastwirkung zum Hintergrund zu bestimmen.

2.7.3 Kopfbereich, Icons und Suchfeld

Im Kopfbereich des Net-Navigator-Fenster können drei Farbflächen, jeweils unterteilt durch eine Trennlinie, beliebig in Farbe und Höhe variiert werden.
Die Farbflächen bilden den Hintergrund für die Icons der History-, Info- und Hilfefunktion, das Sucheingabefeld und die Farblegende.

Die Parameter, mit denen man die **Höhe der Farbflächen** bestimmen kann sind
„BarHeight1“, „BarHeight2“ und „BarHeight3“.
Die Parameter für die **Farbe** sind entsprechend
„BarColor1“, „BarColor2“ und „BarColor3“.

Die **Farbe der drei Trennlinien** lässt sich über die Parameter
„BarColor4“ (Linie zwischen erster und zweiter Farbfläche),
„BarColor5“ (Linie zwischen zweiter und dritter Fläche) und
„BarColor6“ (Linie zwischen dritter Fläche und Arbeitsfläche) verändern.

Setzt man den Parameter „ShowButtonBar“ auf „true“, so werden die **Buttons für History, Info und Hilfe** angezeigt.

Die **Position der Icons** kann verändert werden, indem für jedes Icon eine Bezugsecke sowie ein Pixelangabe relativ zu dieser Ecke angegeben wird.
Die Bezugsecken für die verschiedenen Icons werden über die Parameter
„InfoCornerX“, „InfoCornerY“, „HilfeCornerX“, „HilfeCornerY“, „BackButtonCornerX“, „BackButtonCornerY“, „ForeButtonCornerX“ und „ForeButtonCornerY“ gesetzt.

Gültige Werte für die X-Parameter sind „LEFT“ und „RIGHT“,

für die Y-Parameter „TOP“ und „BOTTOM“.

Die relative Position der Icons wird dann über die Parameter „InfoX“, „InfoY“, „HilfeX“, „HilfeY“, „BackButtonX“, „BackButtonY“, „ForeButtonX“ und „ForeButtonY“ gesetzt.

Im oberen Bereich des Net-Navigator wird ein **Eingabefeld für Suchanfragen** eingeblendet, wenn man den Parameter „ShowSearchField“ auf „true“ setzt.
Die Position und Breite des Eingabefeldes kann über die Parameter „SearchFieldX“, „SearchFieldY“ und „SearchNumberOfColumns“ verändert werden.
Ausserdem kann die Rahmenfarbe („SearchFieldFrameColor“), die Hintergrundfarbe („SearchFieldBackground“) und die Textfarbe („SearchFieldForeground“) eingestellt werden.

Es besteht die Möglichkeit, einen speziellen **Logo-Button** einzublenden, der mit einer URL verknüpft werden kann.
Mit dem Paramter „PaintLogo“ kann diese Funktion aktiviert („true“) oder deaktiviert werden („false“).

Wird PaintLogo aktiviert, müssen zwei Bilddateien für das Logo angegeben werden (Aktiv: „ImageName21“, Passiv: „ImageName20“).
Die Positionierung erfolgt wie bei anderen Icons über eine Bezugsecke („CornerX“, „CornerY“) und eine relative Position („LogoX“ und „LogoY“).
Ausserdem kann das Logo auf eine gewünschte Größe skaliert werden („LogoWidth“, „LogoHeight“).

Die **URL**, die bei einem Klick auf das Logo geöffnet werden soll, wird über den Parameter „LogoURL“ festgelegt.

Sollen **Veränderungen der Filterkriterien im Kopfbereich des Net-Navigator** vorgenommen werden, geschieht dies in der _Kategorien.dat und heißt dort Farblegende (_ShowInColorDescription).

```
### Soll die Kategorie in der Farblegende aufgeführt werden? ###
### default: true ###
#KategorieName_ShowInColorDescription true
```

Soll ein Konzept nicht angezeigt werden, setzt man die spezifischeKategorieName_ShowInColorDescription auf "false":

```
product_ShowInColorDescription false
product_DisplayName Produkt
product_ConceptImage nn-kubus-32.gif
product_InstanceImage nn-kubus-32.gif
```

Tip:
Im Kopfbereich werden Konzepte angezeigt, die dort nicht gewünscht sind (und nicht mit vorherig genannten Befehl ausgeblendet werden können)? Grund ist, dass an einem Begriff im Wissensnetz das Attribut "netNavCategory" mit dem im Kopfbereich angezeigten Namen vergeben wurde. Um auf einen Blick alle diese "Fälle" zu sehen kopiere man nachfolgenden Reguest aus der Java-Konsole

"kp-fertigung1" ist das für dieses Beispiel benutzte Volume - hier muss der jeweils benutzte Volumename stehen.

und lasse sich das Ergebnis im Browser anzeigen

```
<category catname="product3" rootnodename="Lebensmittel" rootnodei
instancecolor="#F9DF84"/>
<category catname="date" rootnodename="Geschäftsereignis" rootnodei
instancecolor="#EBFFEB"/>
<category catname="product" rootnodename="Produkt" rootnodeid="1
instancecolor="#7FBFDF"/>
<category catname="process" rootnodename="Produktionsprozess" root
instancecolor="#F9DF84"/>
```

2.7.4 Farblegende

Der Parameter

„ShowColorDescription“

bestimmt, ob die Farblegende angezeigt werden soll.
Für jede Kategorie kann festgelegt werden, ob sie aus der Farblegende ausgeschlossen werden soll, indem der jeweilige Parameter **_ShowInColorDescription** in der **_Kategorien.dat** auf false gesetzt wird.
Die Namen, unter denen die default-Kategorien (Konzepte, Dokumente, Instanzen) parametrisiert werden können, entsprechen denen, die in der NetNavigator.dat als „ConceptString“, „DocumentString“ und „InstanceString“ festgelegt worden sind.

Über den Parameter

„CDDown“

kann festgelegt werden, ob die Farblegende am oberen (false) oder am unteren Bildschirmrand (true) dargestellt wird.

Die dritte Farbfläche (BarColor3, BarHeight3, ...) wandert im Falle von „true“ automatisch mit an den unteren Bildschirmrand. Sie ist speziell als Hintergrundfläche für die Farblegende gedacht.

Die in der Legende verwendete Schriftart, kann über den Parameter **„BarFont3**“ eingestellt werden.

2.7.5 Kontextmenü (Tool-Tip)

Das Kontextmenü (Tooltip) erscheint, wenn der Benutzer über einem Knoten die rechte Maustaste drückt.
Es kann nur farblich verändert werden, da die Grafiken nicht als Bitmap vorliegen (sie werden gezeichnet).
Die Hintergrundfarbe (**ToolTipBackground**) und die Vordergrundfarbe (**ToolTipTextColor**) sind über Parameter einstellbar.

2.7.6 Suchergebnisdialog

Dieses Dialogfenster ist der Overload-List sehr ähnlich. Es wird angezeigt, wenn der Parameter „**ShowSearchField**“ auf true gesetzt ist, und mehr als ein Treffer auf eine Suchanfragen zurückgeliefert wird.
Es werden die gleichen Bitmaps verwendet wie bei der Overload-List, mit Ausnahme des „Alle markieren“-Buttons, für den sich eigene Bitmaps definieren lassen (passiv: „ImageName22“, aktiv: „ImageName23“).

2.7.7 Knoten und Kanten

Sofern Knoten keiner Kategorie zugeordnet sind, werden sie entsprechend ihres Typs (Konzept, Instanz oder Dokument) eingefärbt.

Die entsprechenden Parameter sind „**DefaultConceptColor**“, „**DefaultInstanceColor**“ und „**DocColor**“.
Für Dokumente kann außerdem noch eine Rollover-Farbe („**DocRolloverColor**“) festgelegt werden.
Die passenden Rollover-Farben der übrigen Knotentypen werden anhand des Parameter „**DarkBackground**“ automatisch vom Programm ermittelt.

Die Größe der Knoten relativ zur Fenstergröße wird über die beiden Parameter „**NodeWidth**“ (Instanz- und Konzeptknoten) und „**DocWidth**“ (Dokumentknoten) festgelegt.

Über den Wert „**MaxNodeWidth**“ bzw. „**MaxDocWidth**“ kann die Maximalgröße der Knoten in Pixeln angegeben werden.

Über den Parameter „**SmallerIndividuums**“ kann festgelegt werden, ob Individuen (d.h. Instanzen) etwas kleiner dargestellt werden sollen als Konzepte. Das Größenverhältnis kann hierbei nicht konfiguriert werden.

Die Farbe der Kanten wird über die beiden Parameter „**EdgeColor**“ (passiv) und „**EdgeRolloverColor**“ (aktiv) bestimmt. Wird die Maus über eine Kante bewegt, so wird die Beziehung zwischen den beiden Knoten, die eine Kante verbindet, angezeigt. Wird die Maus über einen Knoten bewegt, so werden alle Kanten, die von diesem Knoten wegführen, in der aktiven Farbe angezeigt.

2.7.8 Ausblenden von Knoten

Der Net-Navigator hat aus Gründen der Übersichtlichkeit eine Obergrenze für die Anzahl gleichzeitig sichtbarer Knoten („**MaxNode**“).

Sind mehr Knoten sichtbar, blendet Net-Navigator Knoten aus, die schon länger sichtbar sind.

Damit Knoten, die gerade erst zur Ansicht hinzugefügt wurden nicht sofort wieder ausgeblendet werden, kann über den Parameter „**HidingDepth**“ festgelegt werden, wie viele Mausklicks die Knoten mindestens sichtbar bleiben müssen.

Über den Parameter „**UseTransparentGraphics**“ wird festgelegt, ob ein Transparenz-Effekt genutzt werden soll, wenn Knoten ausgeblendet werden. Dieser Effekt ist nicht auf allen Java-Plattformen verfügbar (Internet-Explorer und Netscape 4-Browser unterstützen diese Funktion nicht).

2.7.9 Knoten-Auswahldialog (Overload-List)

Dieser Auswahldialog wird eingeblendet, wenn zu viele Knoten auf einmal sichtbar gemacht werden sollen. Der Schwellenwert für die Anzeige des Auswahldialogs wird über den Parameter **„OpenAtOnceThreshold“** eingestellt.
Für den Auswahldialog lässt sich die Rahmenfarbe (OverloadListFrameColor), die Hintergrundfarbe (OverloadListBackground) und die Vordergrundfarbe (OverloadListTextColor) einstellen.

Im Dialogfenster werden drei Buttons zum Öffnen von Knoten, zur Auswahl aller Knoten sowie zum Abbrechen des Dialogs angzeigt. Das Erscheinungsbild dieser Buttons kann jeweils für den aktiven (Roll-Over) und passiven Zustand separat bestimmt werden. Dazu muss bei den folgenden Parametern der Dateiname einer Bilddatei im GIF- oder JPEG-Format angegeben werden:

- ImageName13 (Öffnen-Button passiv)
- ImageName14 (Öffnen- Button aktiv)
- ImageName15 (Abbrechen- Button passiv)
- ImageName16 (Abbrechen- Button aktiv)
- ImageName17 („Alle markieren“-Button passiv)
- ImageName18 („Alle markieren“-Button aktiv)

2.7.10 Konfiguration von Kategorien

Kategorien werden in einer speziellen Datei namens „**Volume_Kategorien.dat**“ konfiguriert („Volume“ muss durch einen konkreten Volumenamen ersetzt werden). Die Parameter werden einer bestimmten Kategorie zugeordnet, indem die Bezeichnung der Kategorie als Prefix vorangestellt wird. Leerzeichen im Kategoriennamen muss ein Backslash-Zeichen vorangestellt werden („\“).

Kategorien beeinflussen das Erscheinungsbild von Knoten. Dabei ist die einfachste Form die Einfärbung . Zu diesem Zweck kann jeder Kategorie eine Instanzfarbe („InstanceColor“) und

eine Konzeptfarbe („ConceptColor“) zugeordnet werden.
Für den Fall dass eine einfache farbliche Kennzeichnung nicht ausreicht, besteht die Möglichkeit, einer Kategorie spezielle Bilddateien zuzuordnen (Instanzen: „InstanceImage“, Konzepte: „ConceptImage“).

Diese beiden Möglichkeiten können kombiniert werden, indem der Parameter „UseCatColorsInCatIcons“ auf den Wert „true“ gesetzt wird. In diesem Fall werden Knoten als entsprechend eingefärbte Bilddateien angezeigt.

Beispiel für den Inhalt einer vollwertigen Kategorien-Datei:

Erste\ Kategorie_InstanceColor 0000FF
Erste\ Kategorie_ConceptColor 1A1AFF

Kategorie2_InstanceImage instance_kat2.gif
Kategorie2_ConceptImage concept_kat2.gif

KategorieDrei_DisplayName Herrenunterwäsche
KategorieDrei_ConceptColor DDA419
KategorieDrei_ConceptColorActive DDA419
KategorieDrei_InstanceColor DDA419
KategorieDrei_InstanceColorActive DDA419
KategorieDrei_ConceptImage concept_kat3.gif
KategorieDrei_InstanceImage instance_kat3.gif

Dokumente_ShowInColorDescription false
Instanzen_ShowInColorDescription false
Konzepte_ShowInColorDescription false

UseCatColorsInCatIcons false

ImageScalingRaster 10

2.8 Layoutregeln (Constraints)

2.8.1 Layoutregeln zur Laufzeit ein-/ausschalten

Standardmäßig wird im Net-Navigator ein Button zum aktivieren bzw. deaktivieren der Layoutregeln (Constraints) angezeigt. Die Anzeige dieses Buttons kann über den Konfigurationsschlüssel "ShowConstraintsOnOffButton" bestimmt werden. Um den Button auszublenden, muss die folgende Zeile in die NetNavigator.dat eingefügt werden:

ShowContraintsOnOffButton false

3 Die HTTP-Schnittstelle des NetNavigators

3.1 Anfragen des NetNavigators an das Wissensnetz

3.1.1 Einführung

Der NetNavigator ist als Java-Applet realisiert und aufgrund der JavaSecurity Mechanismen darauf angewiesen mit dem Server zu kommunizieren, vom dem er geladen wurde.
Um seine dynamische Funktion möglichst ungehindert durch Firewalls und Proxies zu ermöglichen, sendet er HTTP-GET Requests an den Server und den Port von dem er geladen wurde.
Die Requests, deren Parameter und die Bedeutung derselben sind Bestandteil dieses Abschnitts.
Alle Requests haben das Format (mit einer variablen Anzahl)

http://server/nn?request=name&volume=name&param1=wert&param2=wert

Die zwei Standardparameter haben folgende Bedeutung:

- **request** der Bezeichner des zu bearbeitenden Requests.
  Die Requests werden unten detailliert beschrieben.
- **volume** der Name des Wissensnetzes.

Da der Server mehrere Wissensnetze anbieten kann, teilt der NetNavigator mit, von welchem Netz er Daten abfragen möchte.
Die beiden Parameter **volume** und **lang** werden dem NetNavigator üblicherweise beim Aufruf als Applet-Parameter übergeben und nicht in der Konfigurationsdatei abgelegt, wie das für einige Parameter möglich ist. Die Reihenfolge der Parameter ist nicht festgelegt, wird daher also nicht immer in der hier vorgestellten Reihenfolge vom NetNavigator übergeben.

Folgende optionale Parameter werden in bestimmten Konfigurationen vom NetNavigator gesendet:

- **lang** die gewünschte Sprache der Ausgabe.
  Da Wissensnetze mehrsprachig ausgelegt sein können, fragt der NetNavigator mit diesem Parameter die Werte für eine bestimmte Sprache ab.
- **user** ein Benutzer des Wissensnetzes
  In Wissensnetzen kann das Rechtesystem aktiviert sein, dass für den NetNavigator Lesebeschränkungen definieren kann. Falls der NetNavigator diesen Parameter nicht sendet, geht er davon aus, dass die Rechte eines Standardnutzers für die Generierung der Antworten genutzt werden.
- **densexml** ein boolescher Wert (Standard: false)
  Mit diesem Parameter kann der NetNavigator die Brücke anweisen, eine verkürzte Form des XML zu senden (spart ca. 25% Volumen, ist beschrieben in der NetNavigator-DTD Request-ID.dtd).
- **maxrelations** eine Ganzahl (Standard: unbeschränkt)
  Dieser Parameter steuert, wie viele Relationen pro Knoten maximal an den NetNavigator zurück geliefert werden sollen. Dies ist besonders sinnvoll in Netzen, bei denen einige Knoten sehr viele Verknüpfungen ausweisen.

Die beiden Parameter volume und lang werden dem NetNavigator üblicherweise beim Aufruf übergeben. Die Reihenfolge der Parameter ist nicht festgelegt, wird daher also nicht immer in der hier vorgestellten Reihenfolge vom NetNavigator übergeben.

Wird während der Bearbeitung eines Requests ein Fehler ausgelöst, so wird eine Standardfehlermeldung der Bridge (erkennbar am umschliessenden XML-Element <**KAResponse**>)

an den NetNavigator gesendet.

3.1.2 id-Request

Parameter:

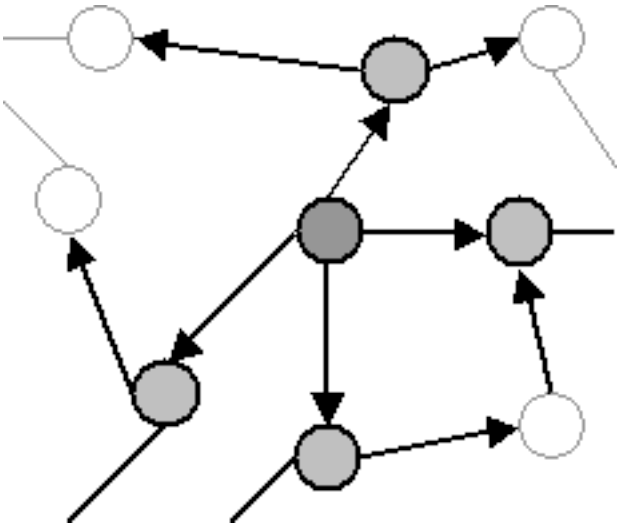
- **id**
  Identifikator des gewünschten Knotens.
- **nodocuments**
  (boolean, Standard: false)
  bestimmt, ob Dokumente an den NetNavigator gesendet werden sollen.
- **noinstances**
  (boolean, Standard: false)
  bestimmt, ob Individuen an den NetNavigator gesendet werden sollen.
- **showtrivials**(boolean, Standard: false) bestimmt, ob triviale Umgebungen angezeigt werden sollen.

Dies ist der meist genutzte Request des NetNavigators und dient zum Lesen eines Ausschnitts des Wissensnetzes um einen oder mehrere bestimmte Knoten herum.

Im Parameter **id** übergibt der NetNavigator typischerweise eine einzige eindeutige Identifikation eines Knotens im Wissensnetzes. Als Antwort erwartet er eine XML-Ausgabe, die der Request-ID.dtd genügt.

Der NetNavigator erwartet, dass der Inhalt der empfangenen Antwort den (oder die) Knoten und seine Umgebung im Wissensnetz beschreibt. Bei diesem Request ist das die Umgebung der Distanz 1 inklusive der Relationen an den äußersten Knoten, also:

- der Knoten selbst,
- alle Kanten (=Relationen) zu umliegenden Knoten, die von diesem Knoten ausgehen,
- alle umliegenden Knoten,
- sowie alle Kanten, die von diesen umliegenden Knoten abgehen zu weiteren Knoten des Wissensnetzes.



Da im Wissensnetz die meisten Relationen gerichtet sind, wird dem NetNavigator nur die Primärrichtung einer Relation übergeben. Bei symmetrischen Relationen werden beide Hälften der Relation übergeben.

Als Besonderheit erlaubt der id-Request auch, keinen oder mehrere durch Komma getrennte Identifier anzugeben. Wird keine Referenz angegeben, so erwartet der NetNavigator den Wurzelknoten als erstes Zentralobjekt in der Ausgabe. Werden mehrere Knoten angegeben, so soll für alle angegebenen Knoten die oben beschriebene Umgebung berechnet und zurück geliefert werden.

Wichtig bei den gesendeten Knoten und Kantenlisten ist, dass der oder die angefragten Knoten (bzw. der Wurzelknoten) am Anfang der Liste der Knoten zurück geliefert werden.

Der NetNavigator speichert die Ergebnisse solcher Anfragen pro Knoten und Sitzung, damit nicht für jeden Interaktionsschritt des Nutzers eine neue Anfrage gestartet werden muss.

Ältere Versionen des NetNavigators verwenden statt der Version mit leerem id-Parameter den default-Request.

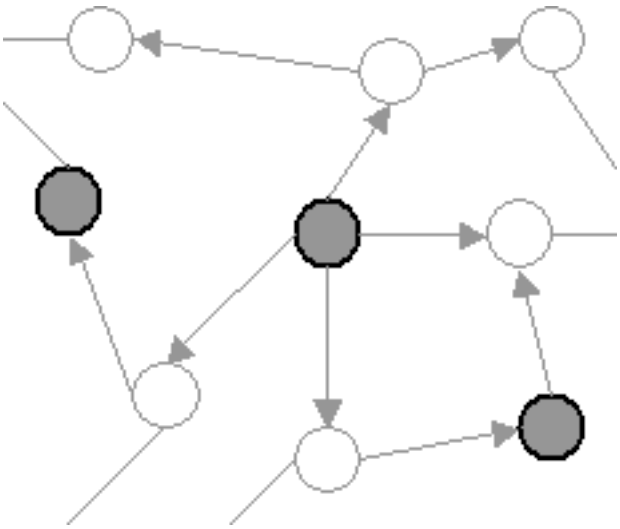
3.1.3 id_trivial-Request

Parameter:

- **id**
  Identifikator des gewünschten Knotens.
- **nodocuments**
  (boolean, Standard: false)
  bestimmt, ob Dokumente an den NetNavigator gesendet werden sollen.
  Dieser Request wird nur dann vom NetNavigator verwendet, wenn in einem id-Request Elemente als trivial gekennzeichnet wurden.

Da an manchen Stellen das Wissensnetz sehr viele Verbindungen zwischen den Knoten haben kann, die alle für den Nutzer trivial verständlich sind (z.B. ist es für Individuen des Begriffs Person meist deutlich, dass es sich hierbei um eine Person handelt), gibt es die Möglichkeit, dem NetNavigator mitzuteilen, dass bestimmte Relationen trivial sind. Solche Mengen von Relationen zeigt der NetNavigator nicht sofort an, sondern erst auf explizite Anweisung des Nutzers hin. Da die Daten dann nicht zum NetNavigator übertragen wurden, stellt dieser den id_trivial-Request, um die weiteren trivialen Daten zu einem Knoten zu erhalten.

Bei diesem Request werden nur die trivialen Anteile übertragen, die in einem normalen id-Request ausgelassen würden.



Die Brücke betrachtet momentan die Individuen der Konzepte als trivial, denen eine Kategorie direkt zugewiesen wurde. Hierbei ist zu beachten, dass ein Konzept zwar zu einer Kategorie gehört, wenn eines seiner Oberkonzepte als Kategorie gekennzeichnet ist, die Trivialmarkierung aber nur erfolgt, wenn die Kategoriezuweisung auch direkt an diesem Konzept stattgefunden hat.

3.1.4 conceptcategories-Request

Der NetNavigator sendet diesen Request einmalig beim Start, um **Kategorien von Konzepten** im Wissensnetz zu erhalten.
Wird er mit einer leeren Liste gemäß der NetNavigator-DTD **Request-Categories.dtd** beantwortet, so geht der NetNavigator davon aus, dass es in diesem Netz keine Kategorien gibt und verwendet nur die standardmäßig vorhandenen, abstrakten Kategorien „Konzepte“, „Individuen“ und „Dokumente“.

Die Kategorien sind orthogonal zum eigentlichen Wissensnetz definierbar, es können also mehrere Teilbäume des Wissensnetzes zu einer Kategorie zugeordnet werden. Der NetNavigator erhält bei folgenden id-Requests bei jedem Knoten die Angabe, zu welcher Kategorie dieser gehört, falls für den Knoten eine Kategorie definiert wurde. Die Kategorie verwendet der Navigator, um das Layout der Knoten zu bestimmen, da für jede Kategorie eigene Farben und Icons definiert werden können.

Zusätzlich dienen die Kategorien dazu, Einsprungpunkte in das Wissensnetz anzubieten, die der NetNavigator anzeigen kann. Über diese Anzeige wird auch eine sichtbare Legende für den Benutzer realisiert, damit zu jedem Knoten im NetNavigator leicht nachzuvollziehen ist, zu welcher Kategorie dieser gehört.

Falls keine Farbverteilung explizit in der Layoutbeschreibung des NetNavigators vereinbart ist, können hier übertragene Farbwerte für Konzepte und Individuen das Layout der Netzes einfärben.

Da dieses Verhalten mehrfach überarbeitet wurde, gibt es mehrere ältere NetNavigator Versionen, die diesen Request auf andere Weise stellen. Ältere NetNavigatoren stellen eventuell nur den concepts-Request, da sie diese noch keine **Relationskategorien** (siehe dort) kennen.

Für den Fall, dass in dem Netz das Rechtesystem aktiviert ist, werden für die Kategorien die Rechte des anfragenden Nutzers überprüft und nur die lesbaren Kategorien zurückgegeben.

Die Berechnung der möglichen Kategorien findet nach deren Eintrag oder Änderung im Wissensnetz im KnowledgeBuilder durch den Menüpunkt **Werkzeuge - WWW - NetNavigatorKategorien** statt.

3.1.5 relationcategories-Request

Äquivalent zum conceptcategories-Request fragt der NetNavigator hier nach **Relationskategorien**.

Die Kategorisierung der Relationen dient dazu, das Layout der verschiedenen Relationsarten zu bestimmen. Als Antwort erwartet er hier nur die Liste der im Wissensnetz erwähnten Kategorien von Relationen.

Die Zusatzinformation zu Einsprungpunkten und Farben entfällt bei Relationskategorien. Wird eine leere Liste an den NetNavigator übergeben, so stellt er alle Relationen auf die gleiche Weise dar. Auch für diesen Request findet die Berechnung möglicher Kategorien nur auf expliziten Wunsch des Nutzer im KnowledgeBuilder statt.

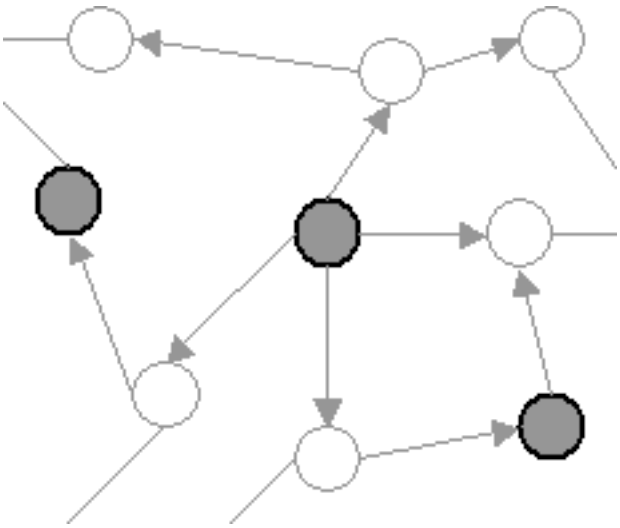
3.1.6 search-Request

Parameter:

- **search**
  Im Wissensnetz zu suchender Text.

Dieser Request dient dem NetNavigator zum Ausführen einer Suche im Wissensnetz. Der Benutzer kann im Interface des NetNavigators auf diese Weise nach bestimmten Knoten, die er anzeigen möchte, suchen.

Als Parameter übergibt der NetNavigator den eingegebenen String und erwartet als Antwort eine Liste der gefundenen passenden Knoten, ohne dass deren Umgebung oder Kanten mit angegeben würde.



Der NetNavigator bietet dem Nutzer dann gegebenenfalls einen Dialog an, aus dem er die relevanten Knoten zur Anzeige auswählen kann und stellt danach einen oder mehrere id-Requests, um die Umgebung der Auswahl zu erhalten.

Da hier nur Knoten mit ihren Namen ausgegeben werden, wird die spezielle NetNavigator-DTD **Request-Search.dtd** als Antwort-Format verwendet.

3.2 Anhang DTD's

3.2.1 Request-ID.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!--  Response format of all ID-Style-Requests.
      Valid: Jan 2003
      -->

<!-- ========================================================== -->

<!ELEMENT KNetNavigator ((Node | Edge)*)>
      <!-- main enclosing element, just a list structure -->

<!-- ========================================================== -->

<!ELEMENT Node (meta*)>
<!ATTLIST Node
      id          ID          #REQUIRED
      name        CDATA       #REQUIRED
      type        (concept|instance|document) #REQUIRED
      nncategory  CDATA       #IMPLIED
      tooltip     CDATA       #IMPLIED
      menuinfo    CDATA       #IMPLIED
      instancecount     CDATA #IMPLIED
      trivialcount      CDATA #IMPLIED
      lecategory  CDATA       #IMPLIED
      >
      <!--  id: internal unique id for this node
            name: name of the node
            type: specifies the type of the node to be displayed
            nncategory: name of the category, this node belongs to
            tooltip: a tooltip string to show in the netnavigator
            menuinfo: an additional string to present in selection
            boxes inside the netnavigator
            instanceCount: number of instances of a concept node
            trivialCount: number of trivial elements connected to
            this node (only valid for concept nodes, which were
            directly assigned to a category)
            lecategory: name of a layout-category this node
            belongs to (only useful, if NN is used with the
            LayoutEngine)
            -->

<!-- ========================================================== -->
```

```
<!ELEMENT Edge EMPTY>
<!ATTLIST Edge
      type        CDATA #REQUIRED
      from        CDATA #REQUIRED
      to          CDATA #REQUIRED
      nncategory  CDATA #IMPLIED
      trivial     (true|false)      #IMPLIED
      >
      <!-- Edge tags represent directed edges between Nodes
            type: contains the type of Edge used to display as
                  name of the edge
            from: id of source node
            to:   id of target node
            nncategory: layout category for this kind of edge
            trivial: signals, if this edge is trivial, default
            false
            -->

<!-- ========================================================== -->

<!ELEMENT meta EMPTY>
<!ATTLIST meta
      name  CDATA #REQUIRED
      value CDATA #REQUIRED
      >
      <!-- used only for document nodes, to transport titles,
           abstracts, urls, etc.
            name:  contains the name of a meta data attribute
            value: value of this meta data attribute -->

<!-- ========================================================== -->
```

```
<!--  long and short version of the tag and attribute names
      and attribute values -->
<!--  long version      short version
      ------------      -------------
      instancecount     ic
      trivialcount      tc
      nncategory        c
      url               u
      instance          instance
      value             v
      type              t
      Meta              M
      name              n
      lecategory        s
      KNetNavigator     NN
      tooltip           tt
      concept           concept

      document          document
      to                to
      menuinfo          mi
      trivial           tr
      Node              No
      from              f
      id                id
      Edge              E
      -->
```

3.2.2 Request-Categories.dtd

```
<!--  DTD for the transfer of categories of topics
      from the knowledge net (volume) to the NetNavigator.
      Valid: Jan 2003
      -->

<!-- ========================================================== -->

<!ELEMENT categorytable (category*)>
<!ATTLIST categorytable
      volume            CDATA #REQUIRED
      >
      <!-- volume: the name of the refering volume -->

<!-- ========================================================== -->

<!ELEMENT category EMPTY>
<!ATTLIST category
      catname          CDATA #REQUIRED
      rootnodename     CDATA #IMPLIED
      rootnodeid       CDATA #IMPLIED
      conceptcolor     CDATA #IMPLIED
      instancecolor    CDATA #IMPLIED
      >
      <!-- catname:        the identifier to display to the user
           rootnodename:   the real name of the root node for this
                           category from the knowledge net used as
                           a starting point for browsing this
                           category
           rootnodeid:     the id of the rootnode
           conceptcolor:   the coloring of concept-nodes
           instancecolor:  the coloring of instance-nodes
                           colors are hex values of the
                           format "#RRGGBB". These colors just
                           state the definition in the volume
                           and can be ignored by the NetNavigator
           -->

<!-- ========================================================== -->
```

3.2.3 Request-Search.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!--  DTD for the responses to search requests.
      Valid: Jan 2003
      -->

<!-- ========================================================== -->

<!ELEMENT KNetNavigator (Node* | TooMuch)>
      <!-- enclosing element -->

<!-- ========================================================== -->

<!ELEMENT TooMuch EMPTY>
      <!-- used, if the result would contain too many items -->

<!-- ========================================================== -->

<!ELEMENT Node EMPTY>
<!ATTLIST Node
      id          ID    #REQUIRED
      name        CDATA #IMPLIED
      type        (concept|instance|document) #REQUIRED
      nncategory  CDATA #IMPLIED
      tooltip           CDATA #IMPLIED
      menuinfo    CDATA #IMPLIED
      >
      <!--  id:   unique identifying id of the found node
            name: name of the node
            type: specifies the type of the node to be displayed
            nncategory: name of the category, this node belongs to
            tooltip: a tooltip string to show in the netnavigator
            menuinfo: an additional string to present in selection
            boxes inside the netnavigator
            -->

<!-- ========================================================== -->
```

4 Java Skript API Dokumentation

4.1 Format der Übergabeparameter und Rückgabewerte

Aufgrund der Schwierigkeiten Arrays von JavaScript nach Java zu übergeben und umgekehrt, wird auf ihre Verwendung verzichtet.
Statt ihrer werden **Strings** übergeben, die die Information durch **Semikolons** getrennt enthalten. Hierbei ist, wenn ein String Werte unterschiedlicher Aussagen codiert auf die Reihenfolge zu achten!

Beispielsweise sollte der Parameter, der der Methode **setCategoriesToFilter**(String szNamesAndValues) übergeben wird immer erst einen Kategorienamen und dann entweder „true“ oder „false“ enthalten:

- Branchen;true;Produkte;false;

Kommen in den Teilstrings selbst Semikolons vor, müssen sie durch einen **vorangestellten Backslash (\)** markiert werden.
Das selbe gilt für in Teilstrings vorkommende Backslashs:

- „Zukunft; zukünftiges\werdendes“ wird als
  „Zukunft\; zukünftiges\\werdendes“ dargestellt.

4.2 Get-Methoden, die es ermöglichen von außerhalb des Applets Parameter zu lesen

public String getSelectedNode()
- Liefert eine Beschreibung des Knotens, der zuletzt aktiv (durch click oder als Resultat einer eindeutigen Suchanfrage) selektiert wurde.
Der zurückgegebene String enthält:
ID;Name;Typ;SuperName;

public String getSelectedNodeName()
- Liefert den Namen des Knotens, der zuletzt aktiv (durch click oder als Resultat einer eindeutigen Suchanfrage) selektiert wurde.

public String getSelectedNodeID()
- Liefert die ID des Knotens, der zuletzt aktiv (durch click oder als Resultat einer eindeutigen Suchanfrage) selektiert wurde.

public String getVisibleNodeNames()
- Liefert einen String, der die Namen der zur Zeit angezeigten Knoten enthält.

public String getVisibleNodeIDs()
- Liefert einen String, der die IDs der zur Zeit angezeigten Knoten enthält.

public boolean getIsNodeVisible(String szID)
- Liefert zurück, od der mit szID spezifizierte Knoten zur Zeit angezeigt wird.

public String getCategories()
- Liefert die Namen aller Kategorien.

public boolean getIsCategoryFiltered(String catName)
- Liefert zurück, ob die mit catName spezifizierte Kategorie zur Zeit ausgefiltert ist.

public boolean getHasOpenFrames()
- Liefert true, wenn mindestens ein JavaFrame zur Darstellung des Graphen existiert.

public int getMaxNumberOfNodes()
- Liefert die Anzahl der Knoten, bis zu der bei Hinzukommen von neuen Knoten keine alten Knoten aus der Darstellung entfernt werden.

public boolean getIsFrameEmbeddedStandby()
- Liefert true, wenn in der Appletansicht, die in den Browserframe eingebettet ist, anstelle des Graphen das StandbyImage (NetNavigator.dat: ImageName24) dargestellt wird.

4.3 Set-Methoden, die es ermöglichen von außerhalb des Applets Parameter zu setzen

public void **setNodeToOpen(String szNodeID)**
Fall 1: Der Knoten wird noch nicht angezeigt
- Es existiert keine Verbindung zwischen dem angezeigten Graphen und dem Knoten.
Der bisher angezeigte Graph verschwindet.
- Der Knoten hat weniger als „OpenAtOnceThreashold“ (NetNavigator.dat Parameter) Nachbarn.
Es wird ein neuer Graph mit dem Knoten als Mittelpunkt geöffnet. Der Knoten wird selektiert.
- Der Knoten hat mehr als „OpenAtOnceThreashold“ Nachbarn.
Der Knoten wird in geschlossenem Zustand gezeigt und selektiert.
- Es existiert eine Verbindung zwischen dem angezeigten Graphen und dem Knoten.
Der bisher angezeigte Graph wird um den Knoten ergänzt.
Der Knoten wird selektiert. Wenn die Anzahl der dargestellten Knoten „MaxNode“ (NetNavigator.dat Parameter) überschreitet, wird ein anderer Knoten entfernt.

Fall 2: Der Knoten wird bereits angezeigt.
- Der Knoten hat weniger als „OpenAtOnceThreashold“ (NetNavigator.dat Parameter) Nachbarn.
Der Knoten wird geöffnet und selektiert.
- Der Knoten hat mehr als „OpenAtOnceThreashold“ Nachbarn.
Es erscheint eine Liste der Kotennachbarn (OverloadList).

public void **setNodesToOpen**(**String nodeIDs)**
- Der bisher angezeigte Graph verschwindet.
Die Knoten, deren IDs in nodeIDs enthalten sind und die Kanten zwischen ihnen werden angezeigt. Dokumente, die mit diesen Knoten verbunden sind werden zusätzlich angezeigt.
Wenn nodeIDs die IDs von Dokumenten enthält, wird deren vollständige Nachbarschaft dargestellt.

public void **setNodesToSelect(String nodeIDs)**
- Selektiert die Knoten deren IDs in nodeIDs ehthalten sind, wenn sie zur Zeit angezeigt werden.

public void **setCategorysToFilter(String namesAndValues)**
- namesAndValues sollte abwechselnd einen Kategorienamen und “true” oder “false” enthalten.
Setzt die Filter der Kategorien auf den in namesAndValues ihrem Namen folgenden Wert.

public void **setSearchString(String searchString)**
- Setzt searchString in das Suchen-Textfeld und startet eine Suche

public void **setOpenFrame(String startNodeIDs, int FrameWidth, int FrameHeight, int FrameX, int FrameY)**
- Öffnet ein neues NetNavigator Fenster und öffnet die Knoten, deren IDs in startNodeIDs angegeben sind. FrameWidth und FrameHeight steuern die Größe des Frames,
FrameX und FrameY legen den linken oberen Bildschirmpunkt des Frames fest. Das Setzen der ints auf -1 bewirkt das Verwenden der DefaultWerte.

public void **setCloseFrames()**
- Veranlasst das Schließen aller zur Zeit existierender JavaFrames des NetNavigators, in denen der Graph dargestellt wird.

public void **setNodesToPin(String szIDs)**
- Pinned die Knoten, deren IDs in szIDs angegeben sind, wenn sie zur Zeit sichtbar sind.

public void **setToFront()**
- Fall 1: Es existieren bereits NetNavigator-Frames
Holt alle NetNavigator-Frames in den Vordergrund.
- Fall 2: Es existiert kein NetNavigator-Frame
Veranlasst die Erzeugung eines neuen NetNavigator-Frames, holt ihn in den Vordergrund und öffnet den Startgraphen.

public void **setToBack()**
- Lässt alle zur Zeit existierenden JavaFrames des NetNavigators, in denen der Graph dargestellt wird, in der Reihenfolge ihrer Erzeugung in den Hintergrund treten.

public void **setStandByFrameEmbedded()**
- Bewirkt, daß in der Appletansicht, die in einen BrowserFrame eingebettet ist anstelle des Graphen ein Bild, das auf die Größe des Applets skaliert wird dargestellt wird. Sollte dann, wenn sich die Größe des Applets ändert und es bereits im Standbymodus ist, erneut aufgerufen werden, um die Größe des Bildes anzupassen. Das Bild kann in der NetNavigator.dat unter dem Eintrag ImageName24 zugewiesen werden. Es ist darauf zu achten daß der .dat-Parameter NumberOfImages mindestens 24 beträgt.

public void **setReactivateFrameEmbedded()**
- Bewirkt, daß falls die in einen Browserframe eingebettete Ansicht des Applets existiert und sich im Standbymodus befindet, diese wieder auf die Graphenansicht umgeschaltet wird.

public void **clearSearchField()**
- Löscht das Suchfeld des aktuellen NetNavigator-Java-Frames.

4.4 JavaSkript-Methoden, die vom NetNavigator aus aufgerufen werden

function eventNodeOpened(id,name,type,superName)
- Wird aufgerufen, wenn ein Knoten für den Browser geöffnet wird.

function eventNodeNavigation(id,name,type,superName)
- Wird aufgerufen, wenn ein Knoten beim Navigieren geöffnet wird.

function eventMultipleNodesOpened(ids, names)
- Wird aufgerufen wenn mehrere Knoten gleichzeitig geöffnet werden.
ids ist ein String, der die IDs der geöffneten Knoten enthält.
names ist ein String, der die Namen der geöffneten Knoten enthält.
Delimiter sowohl für ids , als auch für names ist ; (Semikolon).
Enthält der Name selbst ein Semikolon, wird dieses als \; (Backslash-Semicolon) codiert. Enthält der Name einen Backslash, wird dieser als \\ (Backslash-Backslash) codiert.

function eventNodeSelected(id,name,type,superName)
- Wird aufgerufen, wenn ein Knotens aktiv (durch click oder als Resultat einer eindeutigen Suchanfrage) selektiert wurde.

function eventSearchStarted(searchString)
- Wird aufgerufen, wenn eine Suchanfrage gestellt wird.

function eventSearchResult(ids, names)
- Wird aufgerufen, wenn das Ergebnis einer Suche eintrifft.
ids ist ein String, der die IDs der gefundenen Knoten enthält.
names ist ein String, der die Namen der gefundenen Knoten enthält.
Delimiter sowohl für ids , als auch für names ist ; (Semikolon).
Enthält der Name selbst ein Semikolon, wird dieses als \; (Backslash-Semicolon) codiert. Enthält der Name einen Backslash, wird dieser als \\ (Backslash-Backslash) codiert.

function eventNodeURLClicked(id,name,url)
- Wird aufgerufen, wenn die mit einem Knoten verlinkte URL aufgerufen wird.

function eventFilterActivated(categoryName)
- Wird aufgerufen, wenn eine Kategorie ausgeblendet wird.

function eventFilterDeactivated(categoryName)
- Wird aufgerufen, wenn eine Kategorie wieder eingeblendtet wird.

function eventHistoryBack()
- Wird aufgerufen, wenn in der History zurückgegangen wird.

function eventHistoryFore()
- Wird aufgerufen, wenn in der History wieder vor gegangen wird.

function eventFrameClosed(id,name,type,superName, visibleNodesIDs)
- Wird aufgerufen, wenn ein JavaFrame des NetNavigators, in dem der Graph dargestellt worden ist geschlossen wurde. id, name, type und superName beziehen sich auf den zuletzt selektierten Knoten dieses Frames, visibleNodesIDs entält die IDs der Knoten, die in dem Frame zuletzt angezeigt worden sind.

function eventFrameResized(width, height)
- Wird aufgerufen, wenn die aktive Graphenansicht ein resizeEvent empfängt.

function eventLoadingFinished()
- Wird aufgerufen, wenn der Ladevorgang des NetNavigators beendet ist.

5 Eigenschaften der NetNavigator Bridge und Einstellungsmöglichkeiten

5.1 Einbettung der NetNavigator Anteile in die Bridge

Der NetNavigator kommuniziert per HTTP mit dem Server von dem er geladen wurde. Dieser Server wiederum leitet die Requests an die Bridge weiter, wartet auf deren XML-Antwort und schickt diese dann an den NetNavigator.

Seit Anfang 2004 wurde die alte Classic- oder MultiBridge, die nur per XML kommunizierte in großen Teilen durch die neue Bridge ersetzt. Diese kann, je nach Konfiguration, verschiedene Kommunikationsarten unterstützten, im Moment KEM (RPC-Aufrufe) und Classic. Soll eine solche Bridge auch NetNavigatoren bedienen können, so muss die Bridge entsprechend konfiguriert sein.

5.2 Parameter der NetNavigator Requests (bridge.ini)

Innerhalb der Konfigurationsdatei der Bridge, können verschiedene Parameter für den NetNavigator Anteil gesetzt werden.

Allen Parametern, die für NetNavigator Requests ausgewertet werden ist der **Präfix "netNav_"** vorangestellt (der Präfix ist in der Tabelle nicht aufgeführt).

Parametername	Bedeutung
maxRelations	Anzahl der Relationen, die pro Knoten maximal an den NetNavigator zurück geliefert werden sollen. Dies ist besonders sinnvoll in Netzen, bei denen einige Knoten sehr viele Verknüpfungen haben. Kann auch bei den Requests selbst durch den NetNavigator gesteuert werden. Default: unbegrenzt Wertebereich: Ganzzahlen
checkRelationRights	Dient zum Ausschalten der Rechtüberprüfung der Relationen (nicht der Topics). Dies ist nützlich bei Performanceproblemen durch die Rechteprüfung. Kann nicht in allen Konfigurationen eingesetzt werden. Default: true Wertebereich: true, false

writerClass	Legt die Klasse fest, mit der IDs ausgegeben werden. Diese wird verwendet, wenn der NetNavigator nicht die normalen Frame-IDs aus dem Netz ausgeben soll. Default: KNetNavWriter Wertebereich: KNetNavWriter, KNetNavWriterXID
idIdentifyOrder	Nur sinnvool, falls mit „writerClass“ die Ausgabeklasse auf „KNetNavWriterXID“ gesetzt wurde. Dann bestimmt dieser Parameter die Reihenfolge, in der die Bridge versuchen soll, die IDs auszugeben und bei Anfragen aufzulösen. Default: id Wertebereich: Komma getrennte Liste von: id (Frame-ID), eid (externe ID EID), name
withMetas	Sollen Metaeigenschaften betrachtet und traversiert werden? Sinnvoll ab K-Infinity 3.1 und einem Netz, in dem Relationen nicht zwischen den Topics oder Erweiterungen, sondern auch deren Eigenschaften gezogen sind. Default: false Wertebereich: true, false

5.3 Schemaelemente in K-Infinity-Netzen, die vom NetNavigator genutzt werden

Diverse Schemaelemente von K-Infinity Netzen dienen zum Steuern der Datenübertragung an den NetNavigator. Hierzu werden üblicherweise Attribute an den entsprechenden Elementen angebracht, deren interner Name (internalName) in bestimmter Weise gesetzt sein muss. Im folgenden sind die verschiedenen Schemaelemente und deren Funktion für den NetNavigator beschrieben.

Anmerkung: Es kann vorkommen, dass aus anderen Gründen bereits für ein Schemaelement ein interner Name vergeben ist, der auch so erhalten werden muss. Um dieses Element dann auch für den NetNavigator nutzen zu können, wird momentan eine Entwickler-Verision des KB benötigt, um einen Systemnamen (systemName) für das Proto-Attribut zu vergeben.

5.3.1 Kategorie

Interner Name: **netNavCategory**
Attributtyp: String

Die Kategorie eines Elements im Netz (Topic oder Relation) dient zur Festlegung der Darstellung des Elements im NetNavigator. Die konkrete visuelle Gestaltung wird in den **volume_kategorien.dat** Dateien vorgenommen und ist nicht Teil des Netzes.

Die Kategorie wird an Begriffen innerhalb des Netzes definiert, Individuen erben diese Information automatisch von den Begriffen. Für den Fall, dass ein Individuum zu mehreren Begriffen gehört, wird der in der Hierarchie am nächsten gelegene Begriff gesucht, der eine Kategoriendefinition enthält.

Die Kategorien für ein Netz ruft der NetNavigator beim Start einmal ab. Damit nach einer Änderung von Kategorien im Netz diese Übertragung die aktuellen Werte liefert, muss vorher im Menu Werkzeuge/WWW des KnowledgeBuilders der Punkt NetNavigator-Kategorien aufgerufen werden.

5.3.2 Triviale Knoten

Als trivial nimmt die Bridge alle Individuen an, bei denen an deren direktem Begriff (keine Oberbegriffe) eine Kategorie definiert ist.

Wenn die Anfrage des NetNavigators so gestaltet ist, dass triviale Knoten nicht übertragen werden sollen, so enthält die Umgebung eines Konzepts mit Kategorie keinerlei Individuen.

Umgekehrt wird aber das Konzept dem NetNavigator übertragen, wenn die Umgebung eines Individuums eines solchen Konzepts berechnet wird.

5.3.3 Tooltips

Interner Name: **netNavToolTip**
Attributtyp: String

Der NetNavigator kann zu allen Knoten sogenannte ToolTips einblenden, sofern dies in seiner Konfiguration eingeschaltet ist. Wenn der Mauszeiger kurz über einem Element in der Anzeige des NetNavigators ruht, wird dieser ToolTip angezeigt.

Hat ein Knoten ein Attribut mit diesem Namen, so werden diese Daten an den NetNavigator übertragen.

5.3.4 MenuInfo

Interner Name: **netNavMenuInfo**
Attributtyp: String

Wird innerhalb von Auswahlmenus, die im NetNavigator präsentiert werden, genutzt.
Diese Menus tauchen entweder auf, wenn zu viele Treffer bei einer Suche gefunden wurden oder an einem Knoten zu viele weitere Knoten in der Umgebung vorhanden sind und nicht gleichzeitig geöffnet werden sollen.
Mit MenuInfo kann ein zusätzlicher String in Klammern hinter dem eigentlichen Namen des gefundenen Topics angezeigt werden in diesen Auswahllisten angezeigt werden.

Wenn diese Funktion im NetNavigator aktiviert ist, so überträgt die Bridge diesen Wert. Falls an einem Individuum kein ToolTip angebracht ist, so überträgt die Bridge den Namen des Konzepts als ToolTip.

5.3.5 Knoten ausblenden

Interner Name: **netNavDisable**
Attributtyp: Boolean

Interner Name: **netNavDisableInstances**
Attributtyp: Boolean

In manchen Fällen reichen die Möglichkeiten des Rechtesystems nicht aus, um bestimmte Elemente den Netzes im NetNavigator auszublenden. Dies ist zum Beispiel der Fall, wenn im normalen Layout alle Informationen verfügbar sein sollen, innerhalb des NetNavigators bestimmte (meist in großer Zahl vorhandene) Elemente aber nicht angezeigt werden sollen. Zu dem Zweck des Ausblendens wurden daher die beiden Attribute eingeführt. Die verschiedenen Elemente des Netzes suchen unterschiedliche Attribute und Wege zu diesen, daher folgende Tabelle:

Begriff	Das **netNavDisable-**Attribut des Begriffs wird betrachtet. Oberbegriffe werden nicht herangezogen
Individuum und Dokument	Wenn am Individuum ein **netNavDisable-**Attribut angebracht ist, wird dessen Wert verwendet. Ansonsten wird beim Konzept nachgefragt, ob alle Individuen ausgeblendet werden sollen (netNavDisableInstances Attribut am Konzept).
Relationen	Am Relationsbegriff wird das **netNavDisableInstances-**Attribut gesucht und die Relationen fragen direkt den Begriff. Hierbei ist zu beachten, dass dieses Attribut an der richtigen Relationshälfte angebracht wird. Erst nach erfolgter Prüfung wird die Richtung ermittelt, die dem NetNavigator übertragen werden soll (da dieser immer nur die Primärrichtung erhält, um die Pfeile an den Relationen korrekt darstellen zu können). Anmerkung1: Dies funktioniert momentan nur für Benutzerrelationen, nicht für System- oder Abkürzungsrelationen! Anmerkung 2: Zum nachträglichen Anlegen des entsprechenden Attributs ist eine Entwickler-Version des KB nötig, da K-Infinity 2 das Editieren der Schemaattribute von Relationen noch nicht unterstützt (Siehe KTopRelationConcept>>buildNetNavDisableAttribute)

6 Net-Navigator als Standalone Applikation

6.1 Verzeichnisstruktur

6.1.1 Net-Navigator_Resources

Der Inhalt der Resources unterscheidet sich nicht von der Installation im Knowledge-Portal. Die entsprechenden Informationen finden Sie im Kapitel 1.1 Net-Navigator_Resources

6.1.2 NetNavigator.jar

Die Datei NetNavigator.jar beinhaltet das einzubindende Java-Applet. Dieses wird in der Datei nnBody.html eingebunden.

6.1.3 nnBody.html

nnBody.html ist die HTML-Seite, die im PopUp-Fenster des Net-Navigators geladen wird. Parameter, mit denen der Net-Navigator gestartet werden soll, werden hier direkt beim Einbinden des Applets übergeben. Dazu werden die <param> Tags entsprechend editiert. Weitere Informationen finden Sie im Kapitel Startkonfiguration.

6.1.4 nn.js

Diese JavaScript-Datei beinhaltet alle Funktionen zum Laden und Steuern des Net-Navigators.

Daher muss sie auf der HTML-Seite die den Net-Navigator aufruft eingebunden werden.
Dies geschieht durch folgendes HTML-Element:
<script src="nn.js" type="text/javascript"></script>

6.1.5 callbackFunctions.js

Diese Datei enthält JavaScripts zur Kommunikation mit dem NNLoader. Hier sollten keine Modifikationen vorgenommen werden.

6.1.6 debugConsole.js

Die Datei *debugConsole.js* wird nur zum Debuggen verwendet. Sie beinhaltet die JavaScript-Funktion *log(String text)* um Aufrufe des Net-Navigator-Applets anzuzeigen.

Ist die Variable *"debug"* in der Datei *customFunctions.js* auf "true" gesetzt, öffnet sich beim Start des Net-Navigators ein zusätzliches Fenster. Alle Ausgaben die in den JavaScript-Funktionen durch die Funktion *log()* getätigt werden, erscheinen in diesem Fenster.

6.1.7 netnavLoader.html

netnavLoader.html ist ein Beispiel zum Einbinden des Net-Navigators als Standaloneapplikation. Zur Integration in Ihre Website können die hier enthaltenen JavaScript-Includes und Aufrufe in Ihre Website übernommen werden.

6.1.8 customFunctions.js

Die in *custumFunctions.js* definierten Funtionen bilden die Schnittstelle zwischen dem Net-Navigator und Ihrer Webapplikation.

Die hier definierten Funtionen werden je nach Net-Navigator-Event aufgerufen. Hier können sie eigenen Code integrieren um die Events an Ihre Webapplikation weiterzuleiten. Eine detailierte Beshreibung der Funktionen finden Sie im Abschnitt 4.4 *(JavaSkript-Methoden, die vom NetNavigator aus aufgerufen werden).*

Desweiternen können hier Größe des Popup-Fensters und Debugmodus eingestellt werden.

6.2 Integration in Webseite

6.2.1 Applet Parameter

NNRequestURL

Dieser Parameter bestimmt die Quelle von der der Net-Navigator seinen Daten bezieht. Die als Wert übergebene URL muss auf die entsprechende KMultiBridge verweisen.

In der Regel lautet diese URL *"http://localhost:3030/nn".* Der Parameter würde dann wie folgt aussehen:

<param name="NNRequestURL" value="http://localhost:3030/nn" />

Volume

Beim Starten des Net-Navigators muss der Name des Wissensnetzes übergeben werden. Die geschieht durch den Parameter *"Volume"*. Er bestimmt auch, welche Kategorien-Konfiguration geladen wird. (siehe Kapitel 1.1 Net-Navigator_Resources).

Beispiel:
<param name="Volume" value="myNetName" />

Alternativ kann das Volume auch in der NetNavigator.dat eingetragen werden (Schlüssel "Volume"). Der Eintrag in der Konfigurationsdatei **überschreibt** den hier übergebenen Parameter!

language

Sprache in der der Net-Navigator gestartet wird. Dieser Parameter bestimmt auch welche *language.dat, Volume_Kategorien.dat* und Bilder eingelesen werden. Entsprechende Dateien, die sich im Verzeichnis *Net-Navigator_Resources/{Kürzel der Sprache}* befinden, werden bevorzugt behandelt. Existieren diese nicht, wird auf die Dateien im Verzeichnis *Net-Navigator_Resources/* zurückgegriffen.

Beispiel:
<param name="language" value="ger"/>

StartNodes

Dieser optionale Parameter bestimmt welche Wissensobjekte initial dargestellt werden sollen. Als Wert werden dabei die DMIDs aller Startknoten übergeben. Dieser werden durch Semikolon getrennt übergeben.

Beispiel:
<param name="StartNodes" value="DMID1;DMID2;DMID3;..." />

6.2.2 Startparameter

Alle Parameter, die sie optional zum Starten des Netnavigators übergeben können, werden in die customFunctions.js eingetragen. Dies gewährleistet ein Kapselung und Übersicht Ihrer Modifikationen.

Die hier erläuterten Startparameter sind JavaScript-Variablen, die, falls gesetzt, beim Öffnen des Net-Navigators berücksichtigt werden.

Größe des PopUp-Fensters

Die Größe des Net-Navigator-Fensters kann über die Variablen *nnWidth* und *nnHeight* gesetzt werden. Editieren sie dazu die Datei *customFunctions.js* und fügen Sie folgende Zeilen ein:

```
var nnWidth=600;
var nnHeight=700;
```

Die Werte mussen dabei Ihren Anforderungen entsprechen.

Debug-Modus

Zum Experimentieren können Sie in der Datei *custumFunctions.js* den Debug-Modus aktivieren. Dieser veranlasst, dass ein weiteres PopUp-Fenster geöffnet wird, in dem alle Aufrufe des NetNavigators protokolliert werden.

Das Loggen der Events wird durch den Aufruf *log(text)* in den jeweiligen Funktionsgerüsten in der Datei *customFunctions.js* erreicht. Wenn Sie diese Funktionen anpassen und diese Zeile auskommentieren bzw. entfernen, wird der Event nicht mehr in die Debug-Konsole geloggt!

Die Steuerung des Debug-Modus wird durch das Setzen der Variable *debug* auf *true* bzw. *false* erreicht (*customFunctions.js*):
var debug=true;

6.2.3 Aufruf des Net-Navigators

Der Aufruf des Net-Navigators erfolgt über den NNLoader (in *nn.js* definiert). Dazu wird ein neues Objekt vom Typ NNLoader instanziert. Infolge dessen wird ein weiteres Objekt **nnApplet** erzeugt. Dieses repräsentiert das Applet im DOM und wird zur Kommunikation mit dem Applet verwendet.

Beim Starten des Net-Navigators bestehen zwei Möglichkeiten

Normaler Start

Der Net-Navigator wird normal gestartet. Die initiale Anzeige entspricht, wenn definiert, den Angaben aus der Konfigurationsdatei (z.B. Angezeigte Knoten). Werden in der Konfiguration keine Angaben gemacht, wird der Wurzelbegriff mit seinen Nachbarn angezeigt.

Der Start erfolgt per JavaScript mit folgendem Code:

```
var nn = new NNLoader('nnBody.html');
nn.load();
```

Der Parameter von NNLoader entspricht dabei der relativen URL zum HTML-Dokument des PopUp-Fensters.

Start mit Callbackfunktion

Dieser Start ermöglicht das Ausführen von JavaScript-Code **nach** dem Start des Net-Navigators. Dies wird benötigt wenn direkt nach dem Start auf die API des Net-Navigators zugegriffen werden soll.

Dazu wird nach dem Initialisieren des *NNLoader* der Methode *callback()* eine Funktion übergeben. Diese wird nach dem Start des NetNavigator ausgeführt. Als Parameter wird dabei immer das Applet-Objekt übergeben.

WICHTIG: Wird der Net-Navigator mit der Methode *callback()* gestartet, darf die Methode *load()* nicht mehr aufgerufen werden. Die *callback()*-Methode beinhaltet den Aufruf von *load()*

Das folgende Beispiel zeigt den Aufruf des Net-Navigators per *callback()*. Dabei wird nach dem Start über die JavaScript-API das initial selektierte Knoten mit *alert()* ausgegeben.

```
var nn = new NNLoader('nnBody.html');
nn.callback(function(nnApplet) {
alert("StartNode: "+nnApplet.getSelectedNodeName());
});
```

6.2.4 Integration eigener Funktionen

Abfangen von Events

Die Integration eigener Funktionen findet in der Datei *customFunctions.js* statt. Die hier vordefinierten Funktions-Prototypen werden bei den entsprechenden Events im Net-Navigator aufgerufen. Sie können entweder den Code direkt in diese Funktionen integrieren, oder eine eigene Funktion aufrufen.

Nähere Informationen zu Event und Parameter der einzelnen Funktionen finden Sie in der Java-Script-API in Kapitel 4.4.

Steuern des Net-Navigators

Die Steuerung des Net-Navigators findet über das Objekt **nnApplet** statt. Dieses wird beim Start durch den NNLoader erzeugt. nnApplet unterstützt verschiedenen Methoden zur Interaktion mit dem Net-Navigator. Die zugehörige API finden Sie im Kapitel 4 (*Java Skript API Dokumentation)*.

6.3 Net-Navigator Konfiguration

6.3.1 Allgemeine Konfiguration

Die Konfiguration erfolgt beim Standalone-Betrieb wie beim Betrieb im Knowledge-Portal. Lesen Sie dazu Kapitel 2 *(Konfiguration)*.

6.3.2 Spezielle Parameter für Standalone Applikation

Diese Kapitel weist auf spezielle Konfigurationsparameter hin, die für den Standalone-Betrieb besonders relevant sein können. Die folgenden Abschnitte sollen nur auf die Parameter aufmersam machen und kurz deren Funktion erläutern.

Wie diese Parameter zu wählen sind entnehmen Sie bitte dem Kapitel *Konfiguration*.

6.3.2.1 DownloadGraphAtOnce

Diese Option veranlasst das Applet alle Daten aus dem Wissensnetz beim Initialisieren zu lesen. Dazu werden jedoch vordefinierte Informationen über Kategorien und Relationen im Verzeichnis *Net-Navigator_Resources/data/* erwartet.

6.3.2.2 DoubleClickEnabled

Dieser Parameter sollte beim Standalone-Betrieb auf *true* gesetzt werden, da meist eine Interaktion über Events gewünscht ist. Ist dieser Wert auf *true* gesetzt, wird bei einem Doppelklick der Event *NodeOpened* ausgelöst. Der einfache Klick dient hingegen zum Navigieren im Graphen.

Der Default-Wert ist *false*!

Ein feinere Konfiguration ist mit den folgenden Parametern zu erreichen

6.3.2.3 DoubleClick{1}{2} und SingleClick{1}{2}

Diese dienen zur Definition des Klickverhaltens im Graphen. Diese Parameternamen setzen sich aus verschiedenen Bestandteilen zusammen. die Wertzuweisung erfolgt mit *true* oder *false*. Der Parametername hat dabei zwei variable Anteile. Der erste {1} definiert den Ort, für den das Klickverhalten definiert wird, Parameter zwei {2} die Aktion. Mit dem Wert *true* wird definiert, dass diese Aktion an diesem Ort ausgefürt werden soll, dem Wert *false* die Aktion unterdrückt.

ACHTUNG: Dadurch besteht auch die Möglichkeit bei einem Klick sowohl die Aktion Navigation als auch einen Event auszulösen. Wählen Sie die Einstellungen mit Bedacht.

{1} Der erste Teil definiert den Teil des Knotens für den der Klick registriert wurde. Dabei wird zwischen drei verschiedenen Hit-Zonen pro Node unterschieden.

- **Text**
  Einstellung für Klicks auf den Text des Knotens
- **Symbol**
  Einstellung für Klicks auf das Bild bzw. die Box, die den Knoten darstellt.
- **PlusBox**
  Einstellung für die kleine Box am linken unteren Rand des Icons

Um beim Benutzer unnötige Verwirrung zu vermeiden, wird empfohlen die Einstellungen für alle Hit-Zonen gleich einzustellen.

{2} Bei der Aktion handelt sich immer um *Navigation*, also das Öffnen aller Nachbarknoten im Graphen, oder *Link* für das Auslösen eines Events. Eines dieser zwei Schlüsselworte bildet den zweiten Teil des variablen Parameternamens

Zusätzlich wird immer noch zwischen Single- und Doubleclick unterschieden. Somit ergibt sich folgender Satz an Parametern mit dem das Klickverhalten des Netnavigators konfiguriert werden kann. Die hier eingetragenen Werte repräsentieren die Defaultwerte.

#DoubleClickTextNavigation false
#DoubleClickSymbolNavigation false
#DoubleClickPlusBoxNavigation true
#DoubleClickTextLink false
#DoubleClickSymbolLink false
#DoubleClickPlusBoxLink false

#SingleClickTextNavigation true
#SingleClickSymbolNavigation true
#SingleClickPlusBoxNavigation true
#SingleClickTextLink true
#SingleClickSymbolLink true
#SingleClickPlusBoxLink false

Einstellungen bei Benutzung der JavaScript-API Events

Wenn die Events von Ihrer Applikation abgefangen und ausgewertet werden sollen, empfehlen wir die Einstellung, dass beim einfachem Klick navigiert, und bei doppeltem Klick die Node geöffnet , also der Event *eventNodeOpened()* ausgelöst wird. Die Konfiguration würde also wie folgt aussehen.

DoubleClickTextNavigation false
DoubleClickSymbolNavigation false
DoubleClickPlusBoxNavigation false
DoubleClickTextLink true
DoubleClickSymbolLink true
DoubleClickPlusBoxLink false
#Doppelklick auf PlusBox ist komplett deaktiviert

SingleClickTextNavigation true
SingleClickSymbolNavigation true
SingleClickPlusBoxNavigation true
SingleClickTextLink false
SingleClickSymbolLink false
SingleClickPlusBoxLink false

6.3.2.4 FrameEmbedded

Dieser Konfigurationsparameter gibt an ob der Net-Navigator in einem neuen Fenster geöffnet werden, oder in die aufrufende Seite eingebettet werden soll.eine bestehendes Frame eingebettet werden soll.

Der Wert diese Parameters kann *true* oder *false* sein *(defult: false).*

6.3.2.5 TargetFrame

Dieser Parameter bestimmt, in welchem Frame externe Ressourcen geöffnet werden. Dies betrifft jedoch nur Dokumente und externe URLs, die aus dem Net-Navigator heraus geöffnet werden.

Hinweis: Die Java-Script-API ist von diesem Parameter nicht betroffen. Hier muss die aufgerufene JavaScript-Funktion über das Zielframe entscheiden.