Documentation .INcontext

Contents

[About 1](#_Toc283045392)

[Features 1](#_Toc283045393)

[Installation / Embedding 2](#_Toc283045394)

[Compiled version 2](#_Toc283045395)

[HEAD 2](#_Toc283045396)

[BODY 2](#_Toc283045397)

[Source version 3](#_Toc283045398)

[HEAD 3](#_Toc283045399)

[BODY 3](#_Toc283045400)

[Configuration 4](#_Toc283045401)

[API 7](#_Toc283045402)

[Using custom colors and icons 8](#_Toc283045403)

[File structure 8](#_Toc283045404)

# About

.INcontext is a visualisation tool that allows website builders to embed and visualise RDF relations in a smooth and user friendly way.

# Features

* Light weight client side JavaScript solution using HTML5 and CSS3.
* Hide and show detailed information of the centre object
* Tested in IE6and greater, Firefox 2and greater, Safari 2&greater, Chrome 3 & greater
  + It looks better in modern browsers.
* The “More Info” button can be configured to be handled by the server, to allow specific page rendering options.
* Runs as a sandbox in memory: no conflicts with Jquery
* The history manager is allowed to be switched off to enable the existing history manager of a website to take over.

# Installation / Embedding

Basic Setup required using .INcontext.

Depending on whether you are using the compiled or the source version you need a different set of script includes.

## Compiled version

The compiled version includes all the source files in one file.

### HEAD

Put the following code within the HEAD tag.

<link type="text/css" href="/Content/visualizer.css" rel="Stylesheet" />

<!--[if IE 6]>

<script type="text/javascript" src="/Content/iepngfix\_tilebg.js"></script>

<link type="text/css" href="/Content/visualizer-ie6.css" rel="Stylesheet" />

<![endif]-->

<script type="text/javascript" src="visualize\_compiled\_min.js"></script>

<script type="text/javascript">

var app = new VisualizerApp("{DIV\_VIZ\_ID}", "{OBJECT\_ID}",

{

dataUrl: "../Scripts/mock/mockdata.js",

schemaUrl: "../Scripts/mock/mockschema.js",

titleProperty: "http://purl.org/dc/elements/1.1/title"

});

</script>

### BODY

Put the following code within the BODY tag:

<div id="{DIV\_VIZ\_ID}"></div>

Note:

DIV\_VIZ\_ID: ID of the DIV where the visualization will be rendered

OBJECT\_ID: First object to load

## Source version

Individual source files. Used for debugging and development.

### HEAD

Put the following code within the HEAD tag.

<link type="text/css" href="Content/visualizer.css" media="screen" rel="Stylesheet" />

<link type="text/css" href="Content/visualizer-skin.css" media="screen" rel="Stylesheet" />

<!--[if IE 6]>

<script type="text/javascript" src="Content/iepngfix\_tilebg.js"></script>

<link type="text/css" href="Content/visualizer-ie6.css" rel="Stylesheet" />

<![endif]-->

<script type="text/javascript" src="Scripts/dependencies/jquery-1.4.1.js"></script>

<script type="text/javascript" src="Scripts/dependencies/jquery.address-1.3.1.min.js?tracker=track"></script>

<script type="text/javascript" src="Scripts/dependencies/jquery.effects.core.js"></script>

<script type="text/javascript" src="Scripts/visualizer.js"></script>

<script type="text/javascript" src="Scripts/utils/htmlpopup.js"></script>

<script type="text/javascript" src="Scripts/framework/core.js"></script>

<script type="text/javascript" src="Scripts/framework/sandbox.js"></script>

<script type="text/javascript" src="Scripts/modules/syncdataconnector.js"></script>

<script type="text/javascript" src="Scripts/modules/schemaconnector.js"></script>

<script type="text/javascript" src="Scripts/modules/dataservice.js"></script>

<script type="text/javascript" src="Scripts/modules/schemaservice.js"></script>

<script type="text/javascript" src="Scripts/modules/htmldrawservice.js"></script>

<script type="text/javascript" src="Scripts/modules/historymanager.js"></script>

<script type="text/javascript" src="Scripts/modules/cloudservice.js"></script>

<script type="text/javascript" src="Scripts/modules/navigationservice.js"></script>

<script type="text/javascript" src="Scripts/modules/animationservice.js"></script>

<script type="text/javascript" src="Scripts/domain/node.js"></script>

<script type="text/javascript">

var app = new VisualizerApp("{DIV\_VIZ\_ID}", "{OBJECT\_ID}",

{

dataUrl: "../Scripts/mock/mockdata.js",

schemaUrl: "../Scripts/mock/mockschema.js",

titleProperty: "http://purl.org/dc/elements/1.1/title"

});

</script>

### BODY

Put the following code within the BODY tag:

<div id="{DIV\_VIZ\_ID}"></div>

Note:

DIV\_VIZ\_ID: ID of the DIV where the visualization will be rendered

OBJECT\_ID: First object to load

# Configuration

Description of the available configuration options.

**maxWidth**

Default: 700

Optional

Maximum width of the visualizer canvas in pixels.

**debug**

Default: false

Optional

When set to true, all fired events will be logged to the console.

**dataUrl**

Default: Empty

Mandatory

URL where the RDF data can be found in JSON format.

**schemaUrl**

Default: Empty

Mandatory

URL where the RDF schema data can be found in JSON format.

**idProperty**

Default: Empty

Optional

This property will be used as ID property for all external communication with the visualizer.

If it's not specified, the default RDF ID will be used.

Example: http://purl.utwente.nl/ns/escape-system.owl#id

**titleProperty**

Default: Empty

Mandatory

Defines the field that holds the object's title. This field will be displayed in the visualization.

Example: http://purl.org/dc/elements/1.1/title

**inverseTypeId**

Default: http://www.w3.org/2002/07/owl#inverseOf

Optional

Indicates the identifier of the InverseOf object in the RDF Schema. Used to find inverse relation.

**symmetricTypeId**

Default: http://www.w3.org/2002/07/owl#SymmetricProperty

Optional

Identifier for symmetric properties in the RDF Schema.

**imageTypeId**

Default: Empty

Optional

Defines the field that holds an URI that can be shown as an image. When specified, the field will not be shown as property, only as actual image.

Example: http://xmlns.com/foaf/0.1/img

**annotationTypeId**

Default: Empty

Optional

Type identifier that indicates an object is an annotation on a relation between to objects.

**objectAnnotationTypeId**

Default: Empty

Optional

Identifier for the property that holds the URI to the object of the annotation.

**subjectAnnotationTypeId**

Default: Empty

Optional

Identifier for the property that holds the URI to the subject of the annotation.

**descriptionAnnotationTypeId**

Default: Empty

Optional

Indicates the property identifier for the property on the annotation object that holds the description.

**useHistoryManager**

Default: true

Optional

If true, the browser’s history will be used to be able to navigate back and forward between objects. Deeplinking to objects will work.

If false, URL won't be changed and navigating back and forward using the browser controls won't work.

**showProperties**

Default: true

Optional

It false, the central object will not show its metadata. This saves space, so if the visualizer is intended for navigation only, set this property to false.

**linkTarget**

Default: Empty

Optional

Specifies the target of link properties.

Examples:

“\_self”, links will open in the current window

“\_blank”, links will open in a new window

“\_none”, will be rendered as link, but link won’t open (only “uri-click” event is fired)

“” (default), value will be rendered as plain text

**objectLinkTarget**

Default: \_blank

Optional

Specifies if the link to view the source of an object is rendered and the behavior of the link.

Examples:

“\_self”, object view link will open in the current window

“\_blank” (default), object view link will open in a new window

“\_none”, object view link will be rendered, but link won’t open (only “object-uri-click” event is fired)

“” (Empty), the option to view the object will not be rendered

**concatCharacters**

Default: “ | ”

Optional

Used to concat properties that have multiple values.

**baseClassTypes**

Default:

"http://www.w3.org/2002/07/owl#Thing": "thing",

"http://www.openarchives.org/ore/terms/Aggregation": "aggregation",

"http://xmlns.com/foaf/0.1/Document": "document",

"http://xmlns.com/foaf/0.1/Image": "image",

"http://xmlns.com/foaf/0.1/Group": "group",

"http://xmlns.com/foaf/0.1/Organization": "organization",

"http://xmlns.com/foaf/0.1/Person": "person",

"http://xmlns.com/foaf/0.1/Project": "project",

"http://www.w3.org/2004/02/skos/core#Concept": "concept"

Optional

Specifies the class types to use for the visualizer. This configuration has two effects:

* The aggregation page has a grouping based on base classes. It finds the deepest defined base class for a data node (out of this configuration) that is not defined as “thing”. If no base class is found, the grouping occurs based on the class of the node itself.
* In CSS it is possible to declare styles based on base class. Icons, for instance, are defined in CSS based on the first base class found (out of this configuration).

NB. The value is used as a class identifier in CSS, it is therefore not allowed to use non-CSS class characters in it (allowed characters are: a-z, A-Z, 0-9,- and \_).

# API

**Listening to application events**

Using the subscribe method, you’re able to listen to application events. The following example shows how to listen to the load-object event and specify an action.

**Load-object**

The load-object event is fired whenever an object is loaded. This way, we can always get the current loaded object from the visualizer.

Code:

app.subscribe("\*.load-object", function (event) {

//External object id: event.data

document.getElementById('showObjectHolder').innerHTML = event.data;

});

“app” is a reference to the visualizer application. See the basic installation instructions.

**Uri-click**

The uri-click event is fired when an external link is clicked as one of the object properties. You’re able to trigger custom actions on a link click. See also the “linkTarget” configuration option.

app.subscribe("\*.uri-click", function (event) {

//Link url: event.data

document.getElementById('uriClickDiv').innerHTML = event.data;

});

**Object-uri-click**

The object-uri-click event is fired when the view link of a center object is clicked. The event data holds the Id and ExternalId of the clicked object.

app.subscribe("\*.object-uri-click", function (event) {

//Object id: event.data.id

//External id: event.data.externalId

var values = "Id: " + event.data.id + " + ExternalId: " + event.data.externalId;

});

**Manually loading an object**

When you want to force the application to show an object, the loadObject method can be used.

Example:

app.loadObject('object-id');

The object-id is the object-id of the object to load. When a custom idProperty is specified in the configuration, that property will be used to find a matching value.

# Using custom colors and icons

**Changing colors**

All font and color related styles are defined in the “visualize-skin.css” file. If needed, you can edit this file to change the font or colors used in the visualizer.

**Replacing icons/images**

Provided with the source is a folder with original photoshop (.psd) files containing all images that could need skinning. Edit them and save them to the content folder as .png to add icons/change colors etc.

# File structure

Specification of files used for the visualizer.

**Visualizer.js**

Main application starting point. This script initializes all the services required to run the application. It’s responsible for communication with the HTML page and exposes the API functions.

**Dependencies Folder**

Holds external libraries required for the application. Currently the following dependencies are included:

* Jquery-1.4.1.js
* Jquery.address-1.3.1.min.js
* Jquery.effects.core.js

**Domain Folder**

Holds the domain objects.

**Node.js**

Domain object for a node in the visualizer. Holds data from the RDF data and schema combined.

**Framework Folder**

**Core.js**

Holds the default values for the configuration options. The core is responsible for the initialization of all the services.

**Sandbox.js**

Services can’t communicate directly with the core, all communication is done through the sandbox. It exposes methods for event subscription and logging. It’s also used to get configuration values and a reference to another service.

**Modules Folder**

This folder contains the different services.

**Animationservice.js**

Responsible for animations. The beginTransition method will start the transition between the old and the new canvas.

**Cloudservice.js**

Responsible for creating a unified cloud array with all the relation objects sorted by weight. Weight of a cloud object is based on text length and number of objects in the cloud.

**Dataservice.js**

Uses the dataconnector and schemaservice. Combines the RDF data and Schema data into usable domain objects for the application.

**Historymanager.js**

Uses the jquery.address-1.3.1.min.js dependency to monitor the URL for changes. Provides functionality for the back/forward button in the browser and direct deep linking to an object.

**Htmldrawservice.js**

Uses the historymanager, dataservice and cloudservice.

Responsible to output the correct HTML to visualize all of the objects.

**Navigationservice.js**

Listens to object-click and load-object events.

Calls the dataservice to get the required data and calls the drawsService to draw this data.

**Schemaconnector.js**

Gets the JSON schema data using the configured schemaUrl.

**Schemaservice.js**

Parses the provided schema. Has functions to get property names and types for a specified uri.

**Syncdataconnector.js**

Gets the JSON data using the configured dataUrl.

**Content**

Content files, such as CSS and images, needed to run the application.

* Blank.gif
* Iepngfix.htc
* Iepngfix\_tilebg.js
* Visualizer-ie6.css
* Visualizer-skin.css
* Visualizer.css

**Images**

Holds images files

* Bg-breadcrum-item.png
* Bg-breadcrumbs.png
* Bg-objecttype-cloud.png
* External-link-icon.png
* Hdr-aggregation-icon.png
* Hover-popup-direction.png
* Ico-objecttype.png
* Type-icons.png
* Utility-icons.png