Semester Project

Authoring Tool for Digitally Augmented Paper

Andreas Diener
andreas.diener@vis.ethz.ch

May 3rd 2004

Global Information Systems Group
Swiss Federal Institute of Technology (ETHZ)

Professor:
Prof. Dr. Moira C. Norrie

Supervisor:
Beat Signer

Abstract

In the field of pervasive computing one innovation chases the other — and not even paper has been left out. Several projects investigate the augmentation of paper to bridge between conventional paper and digital devices. In case of enhancing printed media, a postprocessing phase called 'authoring' becomes inevitable. This semester project, part of the European project Paper^{++} under the Disappearing Computer Programme, presents an authoring tool for digitally augmented paper. The report addresses issues of the authoring process, architectural design of the tool as well as a quick introduction to the most relevant features.

Contents

Chapter 1

Introduction

The work presented in this paper evolved in connection with the European project Paper^{++} under the Disappearing Computer Programme (IST-200-26130) [8].

Despite immense advances in computing technologies and predictions of the paperless office, paper consumption continues to rise in the digital age. This is not merely a remnant of outdated working practice; paper is cheap and supports many ways of collaboration and interaction. Paper^{++} tries to bring together the advantages of both, the digital world as well as the physical paper.

To understand the motivation for such an authoring tool as it is described in this report, it is favourable to be aware of the way the Paper^{++} system works.

Conductive ink and special printing methods are used to create an invisible grid on the paper. Input devices translate this grid information into a pair of coordinates which can then be used to link regions of the physical paper to any digital or physical content. For the mapping between physical and digital information, an Integration Server (iServer) [1, 5, 6] based on the object oriented database system OMS [7] is used. The semantically strong data model enables the system to link together arbitrary entities such as shapes on physical paper, physical pages, physical documents, movies, texts, images and webpages. A link is considered to be an abstract concept that enables a connection between two or more resources and is extensible by new types. The model also provides functionality for layering and user management.

As stated above, areas of a document are linked to arbitrary entities. It is the purpose of an authoring tool to provide the functionality to define such areas and associations. In the Paper^{++} system the sources and anchors of links within documents may consist of regions on specific pages that can be defined by a set of simple geometric shapes or by any combination of them. The current set of basic shapes supported include rectangles, polygons, ellipses, and circles. Hence, an authoring tool has to provide the following basic functionality:

- All basic shapes can be defined
- Basic shapes can be combined to define a complex shape
- Areas can be associated with arbitrary entities
- Layers can be created and shapes can be assigned to layers
- User management has to be supported

This document describes the Paper^{++} authoring tool that evolved from a semester project under the supervision of Beat Signer and Prof. Moira C. Norrie in the Institute for Information Systems at ETH Zurich. A general overview of the authoring process can be found in Chap. 2. Chapter 3 provides a description of the overall architecture of the authoring tool. A quick tour of the authoring tool is presented in Chap. 4 and final conclusions as well as an outlook on possible future work can be found in Chap. 5.

Chapter 2

Authoring

For the term 'authoring' several slightly different definitions can be found. Sometimes it is referred to as "programming by non-programmers", sometimes it stands for 'development' and sometimes it is just a synonym for 'editing'. In this report, the term authoring describes the process of linking areas of a document to arbitrary entities. In order to exploit the power of the Paper^{++} system, a user should be able to easily define his own links and not only follow predefined links. Here the authoring tool comes into play. Additionally, it visualises existing links and provides the functionality to modify and change existing ones.

Several different types of authoring exist. They are best described in [5]:

> *The first factor to consider is whether the content is already available in the form of digital and printed materials. If so, the main authoring activity is* link authoring *and a tool is required to support creation of links between the existing content elements. The second factor is whether only the publishers can author links or also the users of the material in which case* dynamic link authoring *must be supported. ... If the content does not already exist, then the authoring activity will consist of both* content authoring *and link authoring. In this case, the content can be developed with the resulting hypermedia system in mind and tools can be used to generate both printed and digital documents along with the links between them.*

The following sections describe aspects of link authoring and content authoring with respect to the Paper^{++} authoring tool.

2.1 Link Authoring

If content is available only in the form of printed material, a tool is required to support the creation of links between the existing content elements. Areas within a document are linked to objects in the information server. The tasks of an authoring tool are therefore to provide a possibility to define new links, to define areas in documents that form source or target of these links, and to highlight existing links in a way that the user is aware of them.

A major drawback of link authoring is that most changes to document content will alter the physical position of elements. The result is, that in an approach where only location information of an area is stored, the link authoring has to be redone upon changes to the document in order to ensure the semantics of a link. Approaches to circumvent invalidation of the position of a link's source or target upon minor changes in document content can be found in [9].

The authoring tool presented in this report provides functionality for link authoring.

2.2 Content Authoring

Content authoring will be the method of choice when both printed and digital material still are to be developed. In this case, not areas on existing paper but chunks of information are linked to other objects. Such link information can be stored and later used in the publishing process to create the necessary link functionality. This process is very similar to the publishing processes in today's content management systems and can be carried out using XML and XSLT. Appropriate XSLT templates are used to transform XML content to multiple delivery channels, e.g. HTML, WML or XML as described in [10]. In this transformation process, semantic link information can be processed and PDF, PS or TEX files could be generated. This approach of linking content to other entities has the advantage of making the links resistant to changes. This means that the authoring process has to be done only once.

Chapter 3

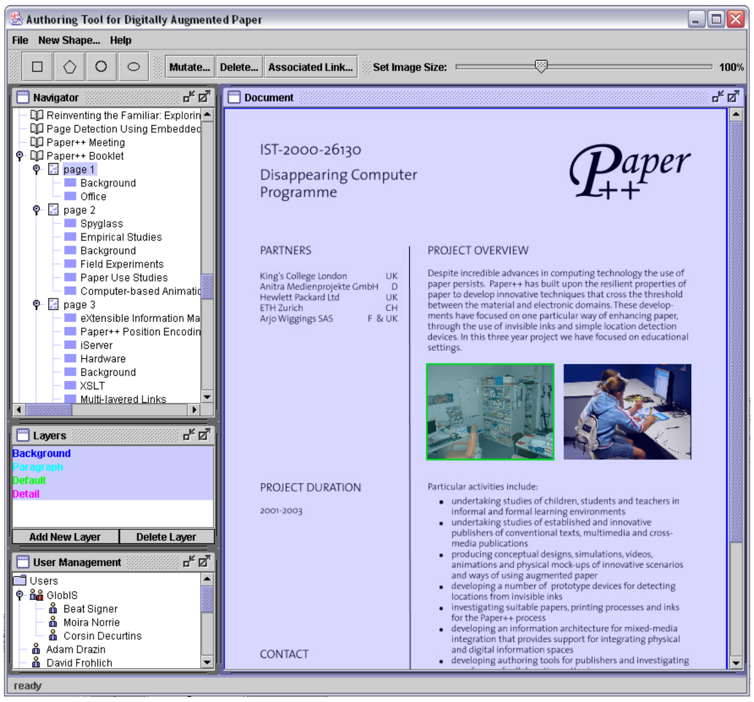
Architectural Overview

3.1 Java as Programming Language

Java was chosen as programming language for several reasons. First it integrates well into the existing Paper^{++} system which is written in Java as well. Another reason was that Java ships with an easy to use and very powerful component kit, Swing. Swing is part of the Java Foundation Classes, or JFC, a collection of APIs for developing graphical user interfaces. Apart from Swing, the JFC include the Abstract Window Toolkit (AWT), the 2D API and the Accessibility API. Swing is a set of components built on top of the AWT including a multitude of additional components that AWT lacks [3, 4, 11].

3.2 iServer and Paper^{++}

As described in [5], the iServer architecture enables mixed-media linking based on a core set of link management information concepts. The iServer architecture is designed as a platform which can be extended to support new resource types based on a plug-in mechanism. By implementing media-specific instances of the selector and resource components, any new type of media can be integrated. This framework is used within Paper^{++} for integrating physical paper and digital content. The iServer and the Paper^{++} system both offer functionality that allows the handling of link, document, shape, layer and user information. The authoring tool accesses these operations through the available APIs and encapsulates multiple method calls to logical steps in the authoring process. In presenting a graphical user interface, it can be looked at as an adapter which provides an interface between humans on one side and the iServer and Paper^{++} system on the other. A description of the iServer and the Paper^{++} system can be found in Fig. 3.2.



IST-2000-26130

Disappearing Computer Programme

Paper++

PARTNERS

King's College London	UK
Anitra Medienprojekte GmbH	D
Hewlett Packard Ltd	UK
ETH Zurich	CH
Arjo Wiggings SAS	F & UK

PROJECT OVERVIEW

Despite incredible advances in computing technology the use of paper persists. Paper++ has built upon the resilient properties of paper to develop innovative techniques that cross the threshold between the material and electronic domains. These developments have focused on one particular way of enhancing paper, through the use of invisible inks and simple location detection devices. In this three year project we have focused on educational settings.

PROJECT DURATION

2001-2003

Particular activities include:

- undertaking studies of children, students and teachers in informal and formal learning environments
- undertaking studies of established and innovative publishers of conventional texts, multimedia and cross-media publications
- producing conceptual designs, simulations, videos, animations and physical mock-ups of innovative scenarios and ways of using augmented paper
- developing a number of prototype devices for detecting locations from invisible inks
- investigating suitable papers, printing processes and inks for the Paper++ process
- developing an information architecture for mixed-media integration that provides support for integrating physical and digital information spaces
- developing authoring tools for publishers and investigating

CONTACT

Figure 3.1: Screenshot of the authoring tool

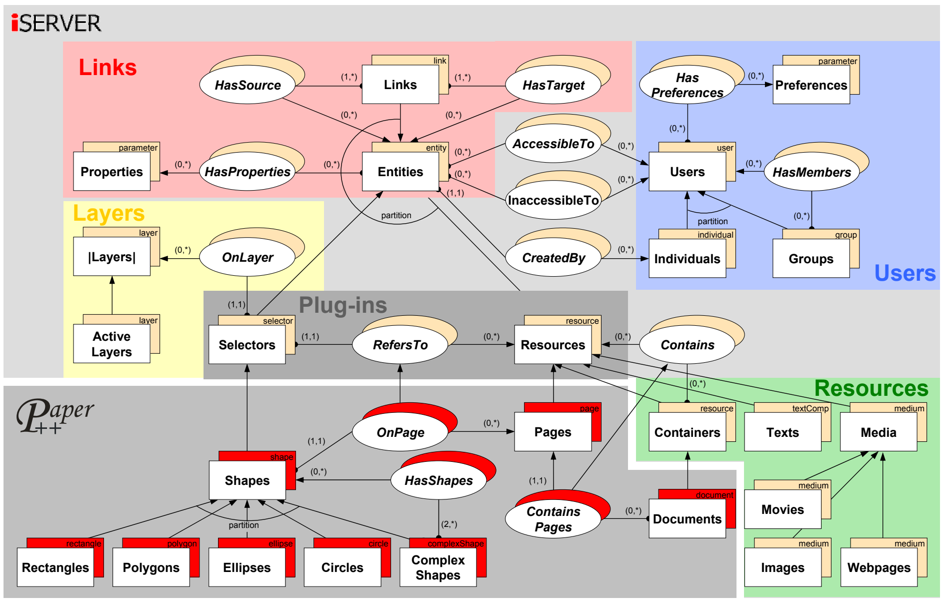


Figure 3.2: Mixed-media information server

3.3 Model View Controller

As M. Geary describes in [4], the Model View Controller (MVC) pattern is used when designing applications that need to provide multiple views of same data. MVC separates applications into three types of objects:

- *Models* are responsible for maintaining data; a `CircleModel` stores all the information of a circle as e.g. the radius and the centre of the circle. Models typically provide methods to access and modify their data. Models also fire events to registered views when a model is changed and the views respond by updating themselves based on the model change.
- *Views* are responsible for providing a visual representation of some portion of a model's data. For example, a `CircleView` provides a view of a `CircleModel` by displaying the circle as a geometric shape. At the same time multiple views of a model can exist.
- *Controllers* handle events for views.

MVC is a powerful design for a number of reasons. First, multiple views can be plugged into a single model. Second, a model's views are automatically notified when the model is changed; changing a model property in one view results in subsequent updates of the model's other views. Third, because models are not dependent upon views, models do not have to be modified to accommodate new types of views.

The MVC pattern is often utilised in the architecture of the authoring tool. Table 3.1 lists some of the model-classes and the corresponding view-classes used.

3.4 Working Areas

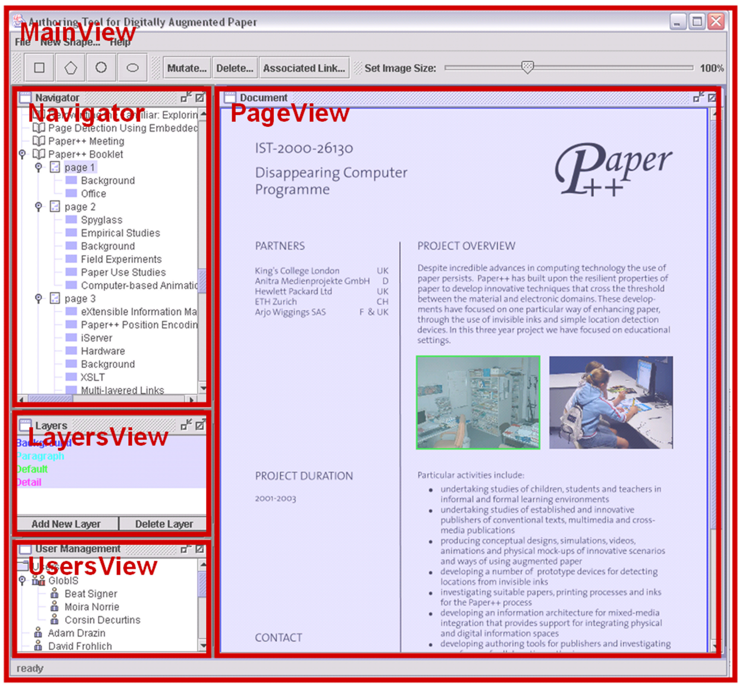
The authoring tool consists of five different working areas that group the available functionality and ensure a intuitive presentation of the already existing link information. All working areas are described below:

3.4.1 MainView

The `MainView` is the window that contains the whole authoring tool. It inherits from `JFrame` and through its menus and toolbars access to the following functionality is provided.

Model-Class	*View-Class*
`DocumentModel`	`DocumentEditView`
`PageModel`	`PageView`
	`PageEditView`
	`MultiplePageEditView`
`CircleModel`	`CircleView`
	`CircleEditView`
	`ChangingCircleView`
`EllipseModel`	`EllipseView`
	`EllipseEditView`
	`ChangingEllipseView`
`PolygonModel`	`PolygonView`
	`PolygonEditView`
	`ChangingPolygonView`
`RectangleModel`	`RectangleView`
	`RectangleEditView`
	`ChangingRectangleView`
`IndividualModel`	`IndividualEditView`
`GroupModel`	`GroupEditView`
`LayerModel`	`LayersView`
	`LayerEditView`
`MainModel`	`MainView`

Table 3.1: Layer constraints



IST-2000-26130

Disappearing Computer Programme

PARTNERS

King's College London	UK
Anitra Medienprojekte GmbH	D
Hewlett Packard Ltd	UK
ETH Zurich	CH
Arjo Wiggings SAS	F & UK

PROJECT OVERVIEW

Despite incredible advances in computing technology the use of paper persists. Paper++ has built upon the resilient properties of paper to develop innovative techniques that cross the threshold between the material and electronic domains. These developments have focused on one particular way of enhancing paper, through the use of invisible inks and simple location detection devices. In this three year project we have focused on educational settings.

PROJECT DURATION

2001-2003

Particular activities include:

- undertaking studies of children, students and teachers in informal and formal learning environments
- undertaking studies of established and innovative publishers of conventional texts, multimedia and cross-media publications
- producing conceptual designs, simulations, videos, animations and physical mock-ups of innovative scenarios and ways of using augmented paper
- developing a number of prototype devices for detecting locations from invisible inks
- investigating suitable papers, printing processes and inks for the Paper++ process
- developing an information architecture for mixed-media integration that provides support for integrating physical and digital information spaces
- developing authoring tools for publishers and investigating

CONTACT

Figure 3.3: Working areas of the authoring tool

- Load a dump file of an existing Paper^{++} database
- Load a dml file of an existing Paper^{++} database
- Create a dump file of the currently open Paper^{++} database
- Create a dml file of the currently open Paper^{++} database
- Exit the system and close all windows
- Create a new shape (circle, ellipse, polygon or rectangle)
- Mutate the currently selected shape
- Delete the currently selected shape
- Associate the currently selected shape with any entity, thus creating new links and editing or extending existing ones
- Zoom-in or -out on the current pageview

3.4.2 PageView

The `PageView` displays one page and its shapes. The page is represented by an image generated from the document's `pdf` file. Existing shapes can be selected, moved around as well as deleted. New shapes can be drawn directly onto the page by dragging an outer-bound box with the mouse. Polygons are created by clicking onto the spots were the edges are supposed to be.

3.4.3 Navigator

The `Navigator` displays all documents, pages and shapes in hierarchical order. It reveals a structural overview of the current database and allows direct navigation between the different pages. Double clicking on a page or shape will cause the appropriate page to be loaded into the `PageView`. Upon a right click inside the `navigator`, a context menu will appear and offer a choice of the following commands:

- Add a new document to the current Paper^{++} database
- Delete the selected document
- Add a new page to the selected document
- Add multiple pages to the selected document
- Delete the selected page

3.4.4 LayersView

The `LayersView` displays all layers of the current Paper^{++} database. Every layer is displayed in a different colour and all shapes on a specific layer are displayed in the same layer-colour. All the layers of a Paper^{++} database are ordered and thus build a ranking. The authoring tool uses a `JLayeredPane` to account for the layering. Upon right clicking a layer its position can be altered by moving it upwards or downwards. New layers can be added and existing ones can be deleted. In the `PageView` window only the shapes are displayed, whose layers are selected in the `LayersView`.

3.4.5 UsersView

The `UsersView` displays all users of the current Paper^{++} database. A `User` is either an `Individual` or a `Group` that contains an arbitrary number of `Users`. This is best modelled using the *Composite*-pattern [2] as shown in Fig. 3.4. The `UsersView` offers access to following functionality:

- Add a new group to the Paper^{++} database
- Add a new group or individual to an existing group
- Delete a user (individual or group)
- Modify a user (individual or group)

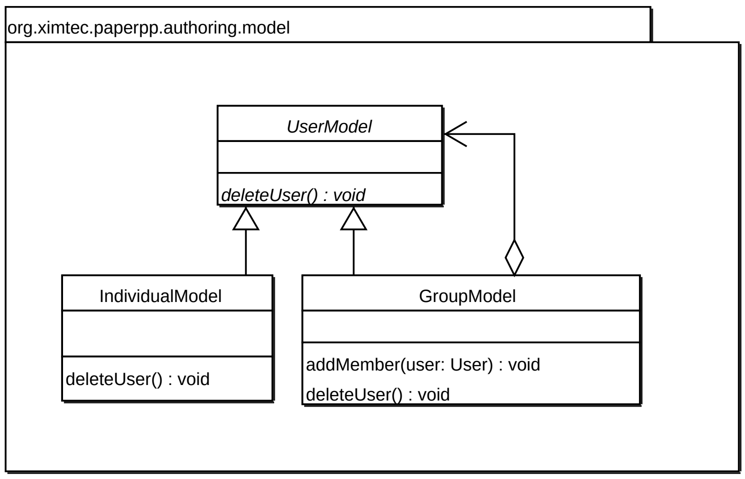


Figure 3.4: User composite pattern

Chapter 4

Quick Tour

This chapter presents a quick tutorial on how to add a new document to an existing Paper^{++} application, on how to add pages and define shapes and links.

4.1 Starting the Authoring Tool for Digitally Augmented Paper

Start the authoring tool by executing the `authoringTool.bat` file in the `org/ximtec/paperpp/authoring` subdirectory or by executing the `main` method in `AuthoringTool.java`.

4.2 Overview of the Authoring Tool

Figure 4.1 shows the overall organisation of the graphical user interface of the Paper^{++} authoring tool. Below the title bar, the menu bar and the toolbar, the workspace is divided into four main areas. The right-hand side area contains the *page view* and displays the content as well as all shapes of the currently selected page. The upper left-hand side area contains the *navigator*, where all elements forming a Paper^{++} database are displayed. The middle left-hand side area includes the *layers view* displaying all existing layers of the currently open Paper^{++} database. Finally, at the bottom left-hand side there is the *user management view* where all users consisting of individuals and groups are displayed.

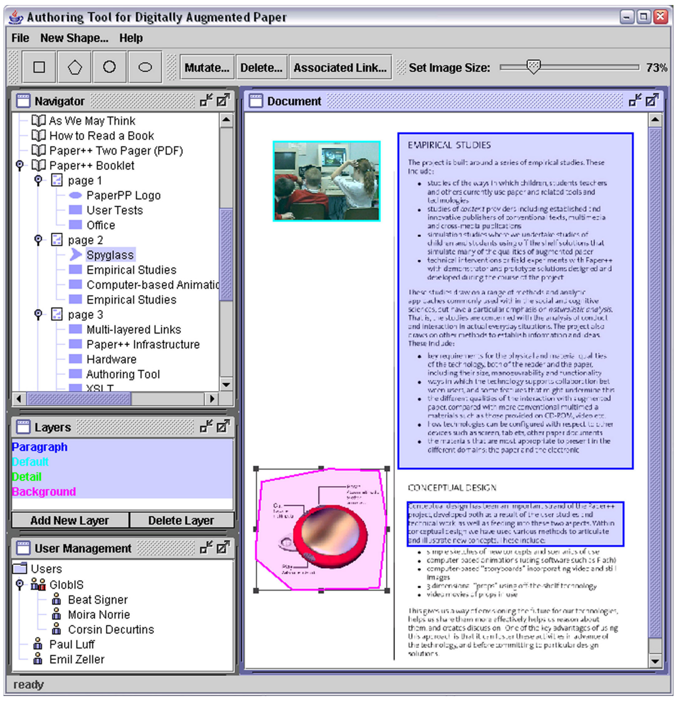


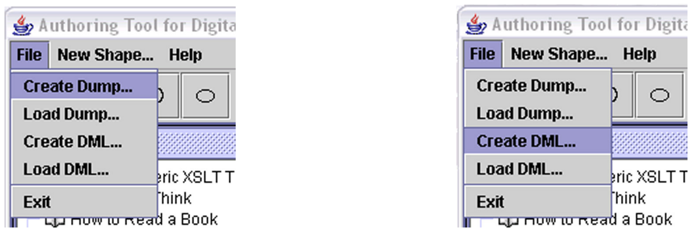
Figure 4.1: Paper^{++} authoring tool overview

4.3 Open/Save Database

After starting the authoring tool, the database of the *paperppSite* application is loaded by default as shown in Fig 4.1. This database contains documents and information about the Paper^{++} project itself and serves due to its numerous shapes and links as an example. At any time when working with the authoring tool, the state of the database can be saved in a file. Two different methods are available: `dump` or `dml` files. A `dump` file contains binary data whereas a `dml` file human readable data. This functionality is available in the **File** menu as illustrated in Fig. 4.2. By first clicking on the **File** menu button with the left mouse button, then holding the button pressed and releasing it on the **Load dump** or the **Load dml** option of the displayed menu, previously created files can be loaded to restore a state of the database.

4.4 Add New Document

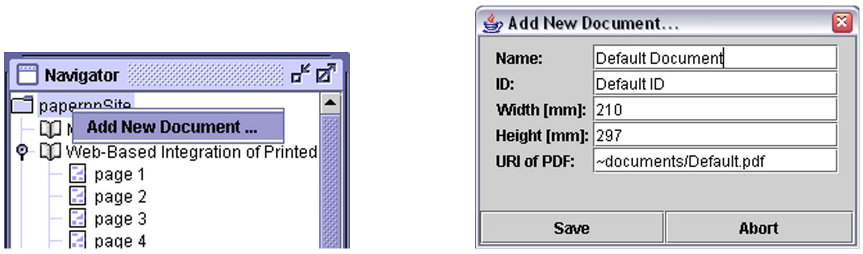
To add a new document to an existing and loaded database, right click the root node in the navigator. A popup menu appears as shown in Fig. 4.3(a).



(a) Creating a `dump` file (b) Creating a `dml` file

Figure 4.2: Save database

Upon choosing the option **Add New Document** a window as illustrated



Name:	Default Document
ID:	Default ID
Width [mm]:	210
Height [mm]:	297
URI of PDF:	~documents/Default.pdf

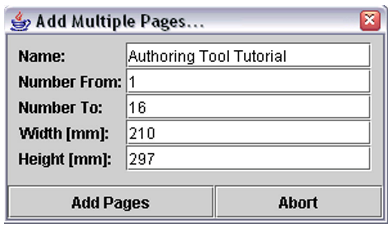
(a) Popup menu (b) Properties

Figure 4.3: Add new document

in Fig. 4.3(b) appears. Provide all necessary information and confirm by clicking the **Save** button. Make sure to provide the correct path of the appendant `pdf` file since this is required to display its content. A new document has been created.

4.5 Add New Page

Right click the newly created document and choose **Add New Page** from the popup menu. Provide the necessary information and press the **Save** button. Repeat this step until all pages of the document are created. To add subsequent pages, a faster and more comfortable way is available: choose the **Add Multiple Pages** option from the popup menu and enter values for **Number From** and **Number To** as shown in Fig. 4.4. All pages with numbers greater equal **Number from** and smaller equal **Number To** will be created automatically.



Name:	Authoring Tool Tutorial
Number From:	1
Number To:	16
Width [mm]:	210
Height [mm]:	297

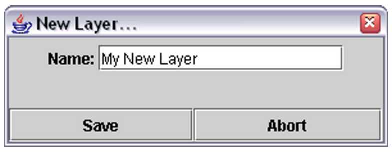
Figure 4.4: Add multiple pages

4.6 Layers View

The layers view is located on the left-hand side in the middle. Note that only the shapes of selected layers are displayed. To display all shapes, select one layer and press the keys **Ctrl** and **A** together.

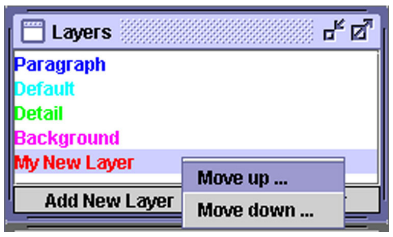
4.6.1 Add New Layer

To add a new layer, click the **Add New Layer** button, enter a name for the new layer in the **Name** field of the **New Layer** dialog shown in Fig. 4.5(a) and confirm by clicking the **Save** button. The layer appears in the layers view at the very bottom. By right clicking the newly created layer and choosing **Move up** or **Move down** in the popup menu as illustrated in Fig. 4.5(b) the layer can be moved to the desired position within the ranking of all layers.



Name: My New Layer

(a) Property



(b) Positioning

Figure 4.5: Add new layer

4.6.2 Delete Layers

Select the layers that should be deleted and click the **Delete Layer** button in the layers view. Note that the layers will be deleted immediately. There is no confirmation dialogue.

4.7 User Management

The user management view is initially located at the bottom left of the authoring tool. It hierarchically displays all groups and individuals as presented in Fig. 4.6.

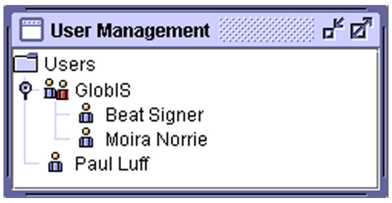
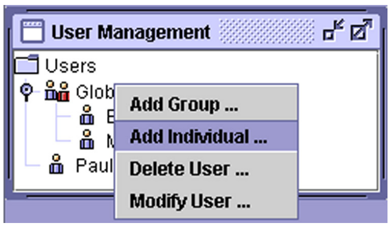


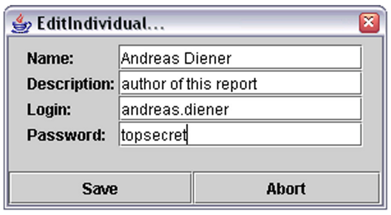
Figure 4.6: User management

4.7.1 Add Individual or Group

To add an new individual or a subgroup to an existing group, right click the group and choose the option **Add Individual** as shown in Fig. 4.7(a) or **Add Group** respectively. An edit individual or edit group dialog window will be opened and you should type name, description, login and password in the fields as shown in Fig. 4.7(b). Confirm by clicking the **Save** button. To add an individual or a group to the Paper^{++} database but to no particular existing group, right click the root node of the user management tree and proceed as described above.



(a) Popup menu



Name: Andreas Diener
Description: author of this report
Login: andreas.diener
Password: topsecret

(b) Properties

Figure 4.7: Add new individual

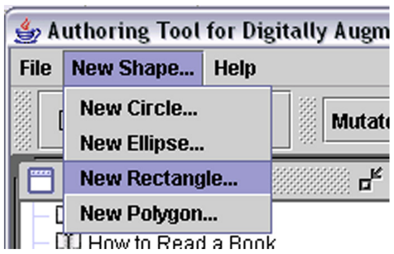
4.7.2 Delete Individual or Group

Right click the individual or group you want to deleted and click the **Delete User** option in the popup menu. Note that prior to deleting a group all its members must be deleted.

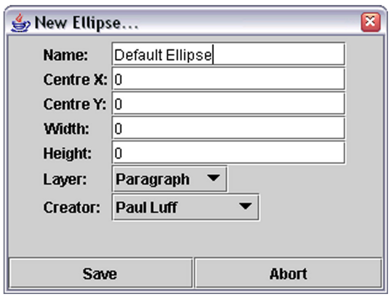
4.8 Shapes

4.8.1 Create Shape

The process of creating new shapes can be carried out in two different ways. Both require that the page the new shape belongs to is opened in the page view. One possibility is to go to the **New Shape** menu and select one of the available types of shapes as shown in Fig. 4.8(a). Four different types of shapes are available: circles, ellipses, rectangles, and polygons. In all cases,



(a) Menu



Name:	Default Ellipse
Centre X:	0
Centre Y:	0
Width:	0
Height:	0
Layer:	Paragraph
Creator:	Paul Luff

(b) Properties

Figure 4.8: Create shape

the system will display a window with all required values for the selected type of shape. To create an ellipse for example, name, x-coordinate and y-coordinate of the centre, width and height of the bounding box, layer and the creator have to be entered in the fields as shown in Fig. 4.8(b). Note that all these values can be changed later. To finally create the shape, click the **Save** button.

Alternatively it is possible to directly paint new shapes onto the page with the mouse pointer. To do so, choose one of the four icons in the toolbar of the authoring tool. When creating a circle, an ellipse or a rectangle, click the left mouse button on the page to define one vertex of the bounding box, then holding the button pressed move to the opposite vertex of the bounding box and release the mouse button. During this process, you can see a preview of the shape that will be created in a gray dashed line as illustrated in Fig. 4.9. After releasing the left mouse button, a dialogue window appears containing all geometric information of the shape. Enter name, layer, and creator of the new shape and confirm the creation by pressing the **Save** button. When creating a polygon, click the left mouse button on the page to define a vertex of the polygon. Repeat this procedure until all vertices are defined, then finish with a double left click. Upon appearance of the dialogue window proceed as described above.

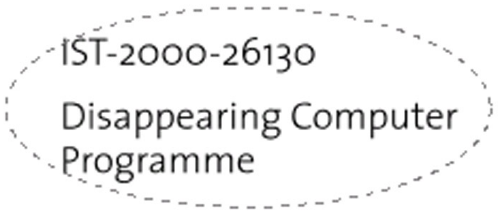
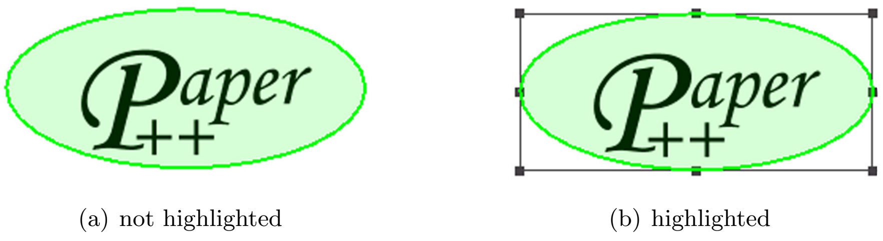


Figure 4.9: Dashed ellipse during creation

4.8.2 Edit Shape

To modify or delete an existing shape, it must be selected first. Shapes can be selected by either double clicking the shape name in the navigator or by directly clicking into the area defined by a shape on the page view. Selected shapes are highlighted as shown in in Fig. 4.10.



(a) not highlighted (b) highlighted

Figure 4.10: Selecting a shape

To move a shape, left click into an already selected shape, hold the button pressed and move the shape to the desired position. While moving, a dashed shape edge indicates the new position. Releasing the mouse button will place the shape to the new position. This process is illustrated in Fig. 4.11.

To modify or delete a selected shape, use the **Mutate** or the **Delete** button in the toolbar of the authoring tool.

4.9 Links

As mentioned in the introduction, one of the main goals of the authoring tool is the functionality of associating areas with arbitrary entities. This means that shapes can be sources of a link but also form a link target.

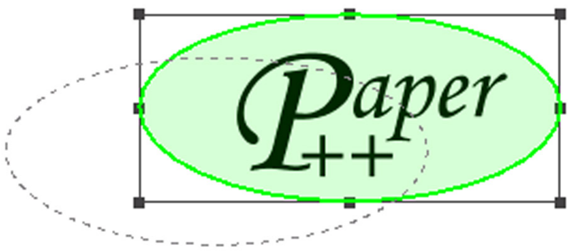
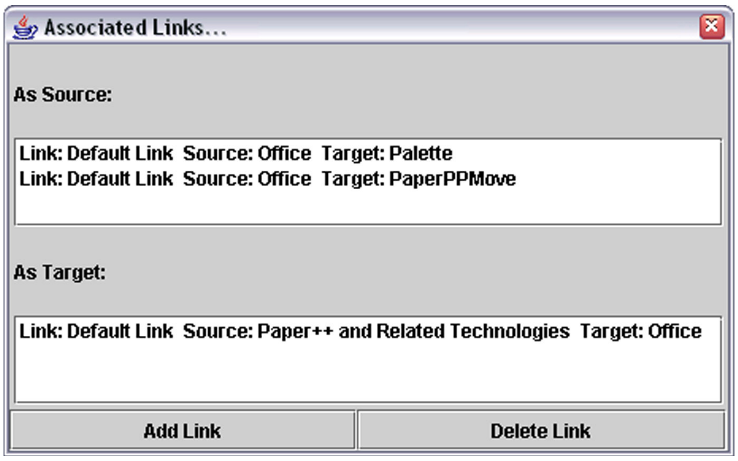


Figure 4.11: Moving a shape

4.9.1 Create Link

To create a link, first select a shape as described in 4.8.2. Then press the button **Associated Link** which is located in the toolbar. As a result the **Associated Links** window as shown in Fig. 4.12 will be displayed. Its upper section lists all links in which the shape takes part as source, the lower section all links in which the shape takes part as target. To create a new link, press the **Add Link** button also displayed in Fig. 4.12.



Link: Default Link Source: Office Target: Palette
Link: Default Link Source: Office Target: PaperPPMove

Link: Default Link Source: Paper++ and Related Technologies Target: Office

Figure 4.12: Associated links

The process of creating a new Link consists of several steps. First, the **Source or Target** dialogue appears where you have to choose whether the shape is source or target of the new link. Figure 4.13 shows the **Source or Target** window.

After choosing between source and target, the **Edit Link** window appears. It represents a link and all its properties, including name, creator, sources,

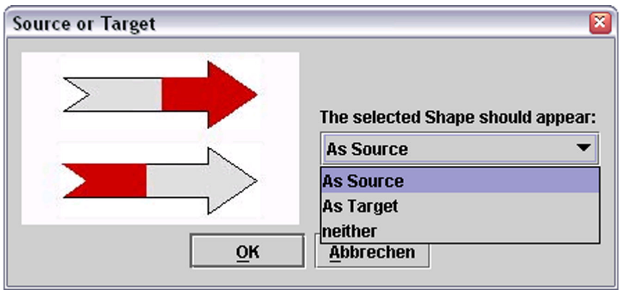


Figure 4.13: Source or target

and targets as shown in Fig 4.14. Its buttons allow to directly create new entities and to add existing ones as link target or source. The link model of the iServer is best described in [6]:

> *To achieve maximum generality, the iServer link model provides the necessary functionality to link entities, where an entity can be an entire information resource such as an image file or an element within a resource such as a word within a printed document or a time sequence within a video file. An entity can equally be either an anchor or a target of one or more links and links can themselves be entities, which allows for links over links.*

To account for such generality, the authoring tool allows to add any in the database existing entity as source or target of a link. To do so use the **Add Existing** button. The **Add New** button of the **Edit Link** window starts a wizard for a quick creation of new resources, limited to webpages, images, movies, and texts. Figure 4.15 shows the **Create Resource** window.

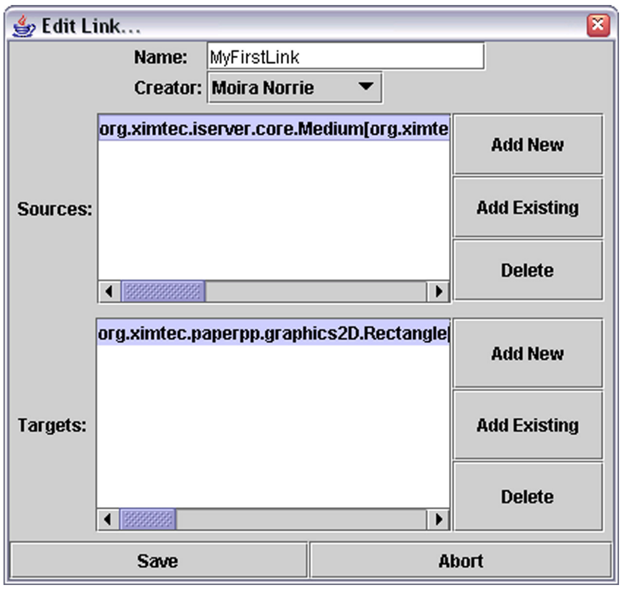


Figure 4.14: Edit link

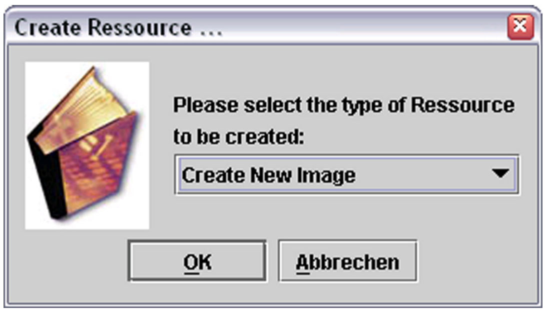


Figure 4.15: Create resource

Chapter 5

Conclusions

One main goal of this semester project was the design and implementation of a graphical user interface allowing to digitally annotate paper and visualise existing links. The current tool provides a graphical user interface for the following tasks:

- Visualisation of any pages stored in a `pdf` file (with stepless zoom functionality)
- Visualisation of all basic shapes (circles, ellipses, rectangles and polygons)
- Editing of existing and creation of new shapes
- Editing of existing and creation of new links
- Editing of existing and creation of new documents
- Editing of existing an creation of new layers
- User management (groups and individuals)

5.1 Future Work

The task to build such a link authoring tool proved to be much more time-consuming and complex than assumed in the beginning. It was not possible to implement all desired features. The following points need to or could be covered in the future:

- Implementation of `ComplexShapes` as well as all relevant functionality.

- *Refactoring* of the existing Java code, e.g. some event handlers are implemented extending `java.swing.AbstractAction`, others implementing the interface `java.util.EventListener`.
- Integration of *undo/redo* operations.
- In the current implementation an `Individual` cannot be part of multiple `Groups`. This is a limitation caused by the graphical user interface. However, the underlying iServer supports multiple group membership.
- No user tests have been carried out.
- The current version does not support *distributed and concurrent* link authoring on the same iServer database.
- Link browsing is not supported.
- It is not possible to connect a "magic" pen to the tool and do the authoring directly on the physical paper.
- It would be desirable to design the authoring tool in a way that components for future media could simply be plugged in, thus form a *generic authoring tool.*

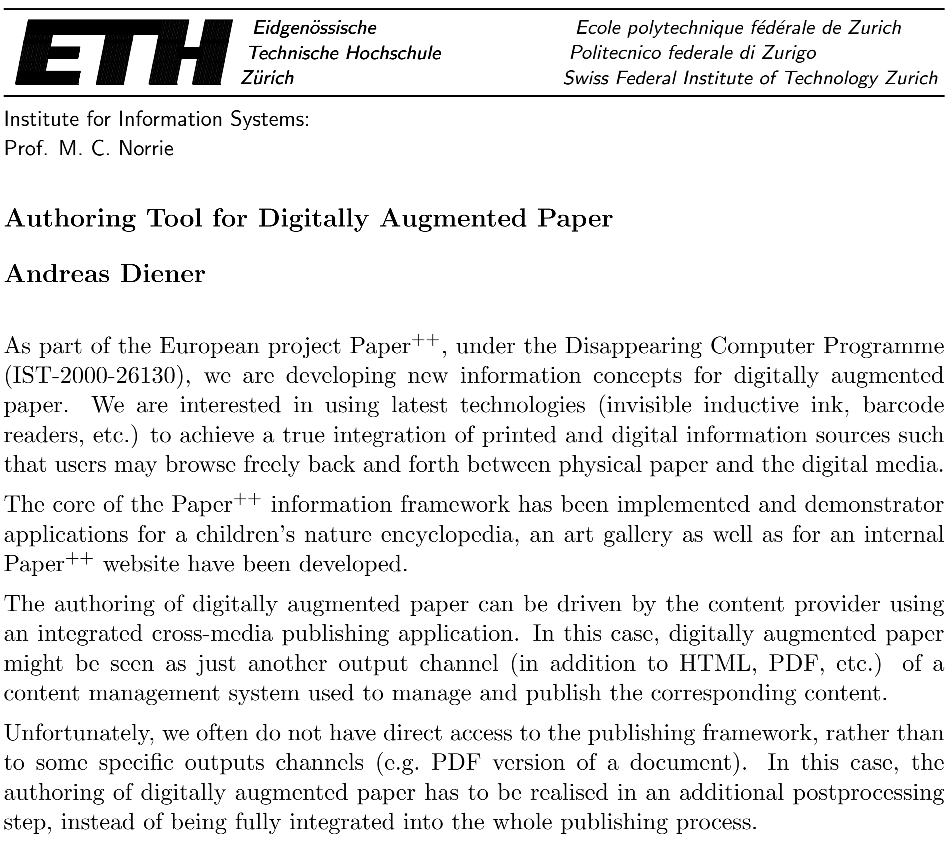
Fast development makes it hard to foresee upcoming trends and directions in this field. Acceptance, meanings and possibilities could evolve which are barely imaginable today.

5.2 Acknowledgements

I am grateful to Prof. Moira C. Norrie and Beat Signer for providing all the help, support and time needed for this semester project. Especially I thank for giving me the opportunity to take part in one of the official Paper^{++} meetings at King's College in London. Also I would like to thank Michael Grossniklaus for some onsite and sometimes immediate help with Java GUI programming.

Appendix A

Task Description



Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institute for Information Systems:
Prof. M. C. Norrie

Authoring Tool for Digitally Augmented Paper

Andreas Diener

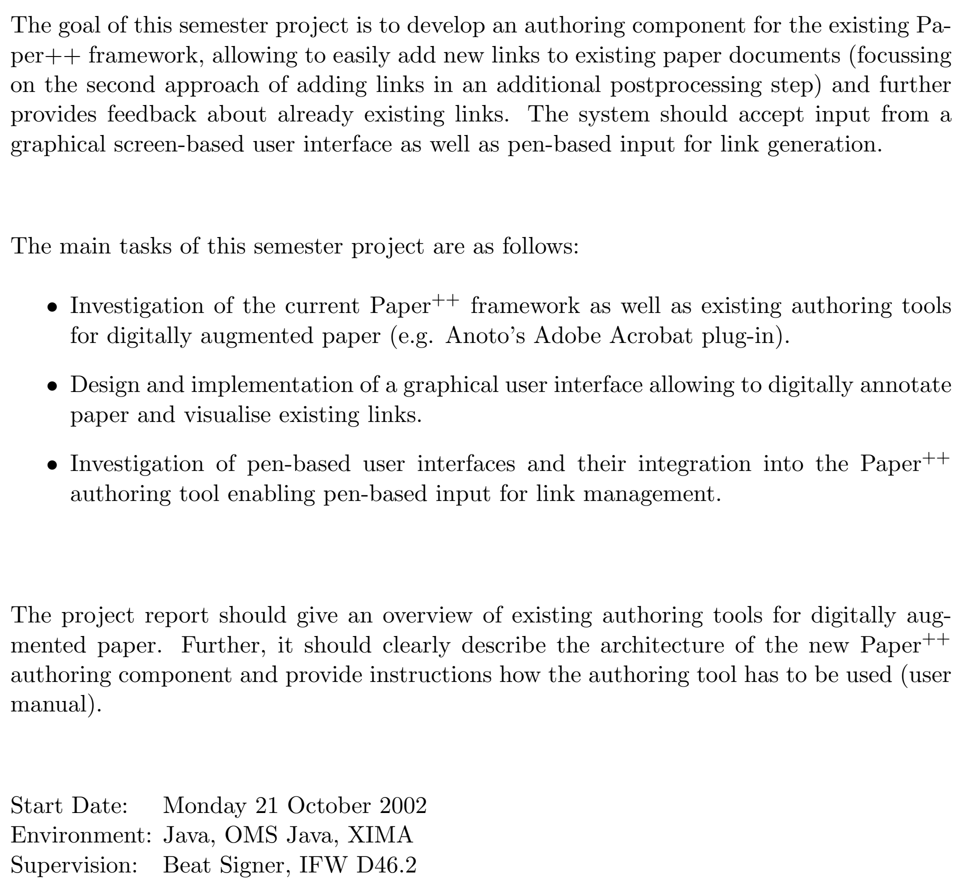
As part of the European project Paper^{++}, under the Disappearing Computer Programme (IST-2000-26130), we are developing new information concepts for digitally augmented paper. We are interested in using latest technologies (invisible inductive ink, barcode readers, etc.) to achieve a true integration of printed and digital information sources such that users may browse freely back and forth between physical paper and the digital media.

The core of the Paper^{++} information framework has been implemented and demonstrator applications for a children's nature encyclopedia, an art gallery as well as for an internal Paper^{++} website have been developed.

The authoring of digitally augmented paper can be driven by the content provider using an integrated cross-media publishing application. In this case, digitally augmented paper might be seen as just another output channel (in addition to HTML, PDF, etc.) of a content management system used to manage and publish the corresponding content.

Unfortunately, we often do not have direct access to the publishing framework, rather than to some specific outputs channels (e.g. PDF version of a document). In this case, the authoring of digitally augmented paper has to be realised in an additional postprocessing step, instead of being fully integrated into the whole publishing process.

Figure A.1: Initial task description page 1



The goal of this semester project is to develop an authoring component for the existing Paper++ framework, allowing to easily add new links to existing paper documents (focussing on the second approach of adding links in an additional postprocessing step) and further provides feedback about already existing links. The system should accept input from a graphical screen-based user interface as well as pen-based input for link generation.

The main tasks of this semester project are as follows:

- Investigation of the current Paper^{++} framework as well as existing authoring tools for digitally augmented paper (e.g. Anoto's Adobe Acrobat plug-in).
- Design and implementation of a graphical user interface allowing to digitally annotate paper and visualise existing links.
- Investigation of pen-based user interfaces and their integration into the Paper^{++} authoring tool enabling pen-based input for link management.

The project report should give an overview of existing authoring tools for digitally augmented paper. Further, it should clearly describe the architecture of the new Paper^{++} authoring component and provide instructions how the authoring tool has to be used (user manual).

Start Date:	Monday 21 October 2002
Environment:	Java, OMS Java, XIMA
Supervision:	Beat Signer, IFW D46.2

Figure A.2: Initial task description page 2

Appendix B

UML Diagrams

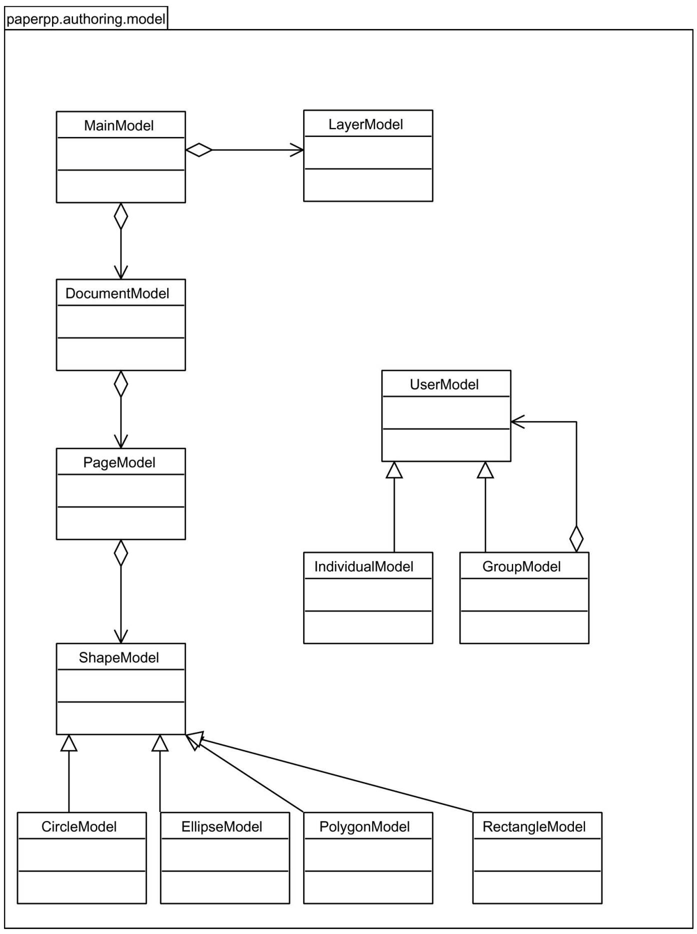


Figure B.1: Model classes

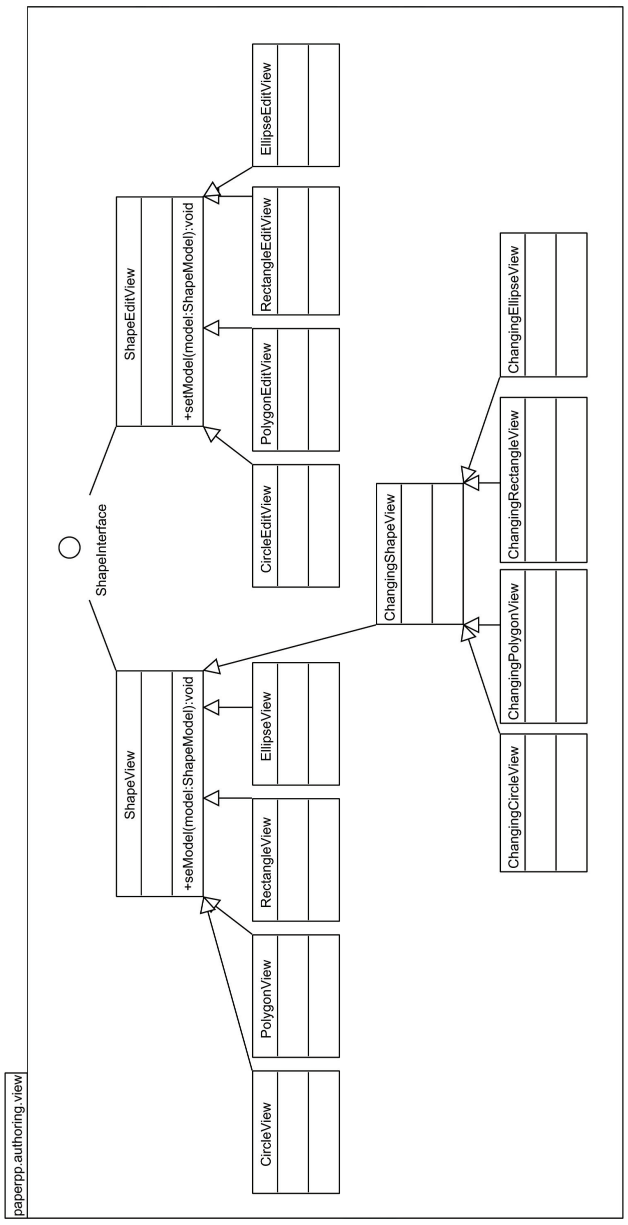


Figure B.2: View classes

Appendix C

Javadoc Documentation

C.1 Package authoring

authoring.AuthoringTool

public *AuthoringTool*
extends Object

Executable Class of the PaperPP Authoring Tool. Starts and initializes GUI. Too chooce initial DB see authoring.model.MainModel

Constructor Summary

Description
AuthoringTool()

Method Summary

Returns	Description
public static void	**main(java.lang.String[] args)** Starts and initializes the PaperPP Authoring Tool.

C.2 Package authoring.view

authoring.view.ChangingCircleView

public *ChangingCircleView*
extends ChangingShapeView

Constructor Summary

Description
ChangingCircleView() Creates a new instance of ChangingCircleView without any ShapeModel.
ChangingCircleView(authoring.model.CircleModel circleModel) Creates a new instance of ChangingCircleView with a CircleModel.
ChangingCircleView(int upperLeftX, int upperLeftY, int width, int height) Creates a new instance of ChangingCircleView with the values upperLeft, width and height.

Method Summary

Returns	Description
public void	**move(int x, int y)** Moves the circle x to the right and y down.
public void	**paint(java.awt.Graphics gr)** Paints the circle onto the screen!
public void	**setModel(authoring.model.ShapeModel model)** Sets a CircleModel
protected void	**sizePanel()** Adjusts the size of the JPanel to fit the current circle.

authoring.view.ChangingEllipseView

public *ChangingEllipseView*
extends ChangingShapeView

This Class represents a ellipse while it is beeing changed or moved around.

Constructor Summary

Description
ChangingEllipseView() Creates a new instance of ChangingEllipseView without any ShapeModel.
ChangingEllipseView(authoring.model.EllipseModel ellipseModel) Creates a new instance of ChangingEllipseView with an EllipseModel.
ChangingEllipseView(int upperLeftX, int upperLeftY, int width, int height) Creates a new instance of ChangingEllipseView with the values upperLeft, width and height.

Method Summary

Returns	Description
public void	**move(int x, int y)** Moves the ellipse x to the right and y down.
public void	**paint(java.awt.Graphics gr)** Paints the ellipse onto the screen!
public void	**setModel(authoring.model.ShapeModel model)** Sets an EllipseModel
protected void	**sizePanel()** Adjusts the size of the JPanel to fit the current ellipse.

authoring.view.ChangingPolygonPointsView

public *ChangingPolygonPointsView*
extends JPanel

This Class represents a polygon that is beeing created. This means that the polygon consists of a number of Points but is no 'closed' and finished yet.

Constructor Summary

Description
ChangingPolygonPointsView(java.util.Vector polygonXPoints, java.util.Vector polygonYPoints) Creates a new instance of ChangingPolygonPointsView with Vectors of the Points.

Method Summary

Returns	Description
public void	**paint(java.awt.Graphics gr)** Paints the rectangle onto the screen!
protected void	**sizePanel()** Adjusts the size of the JPanel to fit the current rectangle.

authoring.view.ChangingPolygonView

public *ChangingPolygonView*
extends ChangingShapeView

This Class represents a polygon while it is beeing changed or moved around.

Constructor Summary

Description
ChangingPolygonView() Creates a new instance of ChangingPolygonView without any ShapeModel.
ChangingPolygonView(authoring.model.PolygonModel polygonModel) Creates a new instance of ChangingPolygonView with a PolygonModel.

Method Summary

Returns	Description
public void	**move(int x, int y)** Moves the polygon x to the right and y down.
public void	**paint(java.awt.Graphics gr)** Paints the polygon onto the screen!
public void	**setModel(authoring.model.ShapeModel model)** Sets a PolygonModel
protected void	**sizePanel()** Adjusts the size of the JPanel to fit the current polygon.

authoring.view.ChangingRectangleView

public *ChangingRectangleView*
extends ChangingShapeView

This Class represents a rectangle while it is beeing changed or moved around.

Constructor Summary

Description
ChangingRectangleView() Creates a new instance of ChangingRectangleView without any ShapeModel.
ChangingRectangleView(authoring.model.RectangleModel rectangleModel) Creates a new instance of ChangingRectangleView with a RectangleModel.
ChangingRectangleView(int upperLeftX, int upperLeftY, int width, int height) Creates a new instance of ChangingRectangleView with the values upper-Left, width and height.

Method Summary

Returns	Description
public void	**move(int x, int y)** Moves the rectangle x to the right and y down.
public void	**paint(java.awt.Graphics gr)** Paints the rectangle onto the screen!
public void	**setModel(authoring.model.ShapeModel model)** Sets a RectangleModel
protected void	**sizePanel()** Adjusts the size of the JPanel to fit the current rectangle.

authoring.view.ChangingShapeView

public abstract *ChangingShapeView*
extends ShapeView

Extending ShapeView this class also provides an abstract skeleton to paint a Shape in the geometry of two dimensions. ChangingShapeViews are used to paint Shapes while they are changing e.g. are draged around (changing position), resized (changing size) and so on...

Constructor Summary

Description
ChangingShapeView()

Method Summary

Returns	Description
public abstract void	**move(int x, int y)**
public abstract void	**setModel(authoring.model.ShapeModel model)**
protected abstract void	**sizePanel()**

authoring.view.CircleEditView

public *CircleEditView*
extends ShapeEditView

A JPanel with Fields for all values that define a circle.

Constructor Summary

Description
CircleEditView() Creates a new instance of CircleEditView.
CircleEditView(boolean isMutate) Creates a new instance of CircleEditView with specified isMutate.

Method Summary

Returns	Description
public Integer	**getCentreXFieldValue()**

Returns	Description
public Integer	**getCentreYFieldValue()**
public Individual	**getCreatorBoxValue()**
public Layer	**getLayerBoxValue()**
public String	**getNameFieldValue()**
public Page	**getPage()**
public Integer	**getRadiusFieldValue()**
public void	**setCentreXFieldValue(int centreXFieldValue)**
public void	**setCentreYFieldValue(int centreYFieldValue)**
public void	**setModel(authoring.model.ShapeModel model)** Sets a CircleModel and fills all fields according to this existing circle.
public void	**setRadiusFieldValue(int radius)**

authoring.view.CircleView

public *CircleView*
extends ShapeView

This Class represents an 'actual' circle one can see on the PaperPP pages.

Constructor Summary

Description
CircleView() Creates a new instance of CircleView without any ShapeModel.
CircleView(authoring.model.CircleModel circle) Creates a new instance of CircleView with a CircleModel.

Method Summary

Returns	Description
public void	**paint(java.awt.Graphics gr)** Paints the Circle onto the Screen!
protected void	**sizePanel()** Whenever the size of the Shape itself has changed, this method resizes the size of the JPanel.

authoring.view.ComplexShapeEditView

public *ComplexShapeEditView*
extends ShapeEditView

Functionality for Complex Shapes is not provided yet.

Constructor Summary

Description
ComplexShapeEditView()

authoring.view.ComplexShapeView

public *ComplexShapeView*
extends ShapeView

Functionality for Complex Shapes is not provided yet.

Constructor Summary

Description
ComplexShapeView() Creates a new instance of Circle
ComplexShapeView(authoring.model.ComplexShapeModel complexShapeModel)

Method Summary

Returns	Description
protected void	**sizePanel()**

authoring.view.DocumentEditView

public *DocumentEditView*
extends JPanel

A JPanel with Fields for all values that define a document.

Constructor Summary

Description
DocumentEditView(authoring.model.MainModel mainModel) Creates a new instance of DocumentEditView and all Swing Elements of that view.

Method Summary

Returns	Description
public String	**getContentFieldValue()**
public int	**getHeightFieldValue()**
public String	**getIdFieldValue()**
public MainModel	**getMainModel()**
public String	**getNameFieldValue()**
public int	**getWidthFieldValue()**

authoring.view.EllipseEditView

public *EllipseEditView*
extends ShapeEditView

A JPanel with Fields for all values that define an ellipse.

Constructor Summary

Description
EllipseEditView() Creates a new instance of EllipseEditView.
EllipseEditView(boolean isMutate) Creates a new instance of RectanlgeEditView with specified isMutate.

Method Summary

Returns	Description
public Integer	**getCentreXFieldValue()**
public Integer	**getCentreYFieldValue()**
public Individual	**getCreatorBoxValue()**
public Integer	**getHeightFieldValue()**
public Layer	**getLayerBoxValue()**
public String	**getNameFieldValue()**
public Page	**getPage()**
public Integer	**getWidthFieldValue()**
public void	**setCentreXFieldValue(int centreXFieldValue)**
public void	**setCentreYFieldValue(int centreYFieldValue)**
public void	**setHeightFieldValue(int height)**

Returns	Description
public void	**setModel(authoring.model.ShapeModel model)** Sets a EllipseModel and fills all fields according to this existing ellipse.
public void	**setWidthtFieldValue(int width)**

authoring.view.EllipseView

public *EllipseView*
extends ShapeView

This Class represents an 'actual' ellipse one can see on the PaperPP pages.

Constructor Summary

Description
EllipseView() Creates a new instance of EllipseView without any ShapeModel.
EllipseView(authoring.model.EllipseModel ellipseModel) Creates a new instance of EllipseView with an EllipseModel.

Method Summary

Returns	Description
public void	**paint(java.awt.Graphics graphics)** Paints the Ellipse onto the Screen!
protected void	**sizePanel()** Whenever the size of the Shape itself has changed, this method resizes the size of the JPanel.

authoring.view.GroupEditView

public *GroupEditView*
extends JPanel

A JPanel with Fields for all values that define a Group.

Constructor Summary

Description
GroupEditView(boolean isMutate, java.lang.Object userModel, authoring.view.UsersView usersView) Creates a new instance of GroupEditView and all Swing Elements of the View.

Method Summary

Returns	Description
public String	**getDescription()**
public GroupModel	**getGroupModel()**
public String	**getName()**
public UsersView	**getUsersView()**

authoring.view.IndividualEditView

public *IndividualEditView*
extends JPanel

A JPanel with Fields for all values that define an Individual.

Constructor Summary

Description
IndividualEditView(boolean isMutate, java.lang.Object userModel, authoring.view.UsersView usersView) Creates a new instance of IndividualEditView and all Swing Elements of the View.

Method Summary

Returns	Description
public String	**getDescription()**
public IndividualModel	**getIndividualModel()**
public String	**getLogin()**
public String	**getName()**
public String	**getPasswd()**
public UsersView	**getUsersView()**

authoring.view.LayerEditView

public *LayerEditView*
extends JPanel

A JPanel with Fields for all values that define a Layer.

Constructor Summary

Description
LayerEditView(authoring.view.LayersView layersView, boolean isMutate) Creates a new instance of LayerEditView and all Swing Elements of the View.

Method Summary

Returns	Description
public LayersView	**getLayersView()**

Returns	Description
public String	**getName()**
public void	**saveModifiedLayer()**
public void	**setLayerModel(authoring.model.LayerModel layerModel)**
public void	**setName(java.lang.String name)**

authoring.view.LayersView

public *LayersView*
extends JPanel
implements Observer

A JPanel that shows all the Layers of the current PaperPP application. It has two Buttons: "Add new Layer" and "Delete Layer". They do what they say they do!!!

Constructor Summary

Description
LayersView(authoring.view.MainView mainView) Creates a new instance of LayersView.

Method Summary

Returns	Description
public void	**deleteSelectedLayer()** Deletes the selected Layers.
public MainView	**getMainView()**
public void	**update(java.util.Observable o, java.lang.Object arg)** Whenever an observed Object changes, this update method is called!

authoring.view.LinksView

public *LinksView*
extends JPanel

A JPanel that shows all Links the selected Shape participates either as Source or as Target.

Constructor Summary

Description
LinksView(authoring.view.PageView pageView, authoring.model.ShapeModel shapeModel) Creates a new instance of LinksView

Method Summary

Returns	Description
public void	**deleteSelectedLink()** Deletes the selected Link.
public void	**fillList()** Fills the two lists (sourceList and targetList) with the links the selected Shape is part of.
public ShapeModel	**getShapeModel()**

authoring.view.LinkView

public *LinkView*
extends JPanel

A JPanel with Fields for all values that define a Link: The Sources, the Targets, possibilities to add and create new Targetes and Sources as well as Save and Abort Buttons.

Constructor Summary

Description
LinkView(authoring.view.PageView pageView, boolean isMutate, authoring.view.LinksView linksView) Creates a new instance of LinkView an all Swing Elements of the View.

Method Summary

Returns	Description
public void	**addNewRessource(org.ximtec.iserver.core.Entity entity, java.lang.String addTo)** Adds a new Ressource (Source or Target) to the current Link.
public void	**addSource(org.ximtec.iserver.core.Entity entity)** Add a Source to the current link.
public void	**addTarget(org.ximtec.iserver.core.Entity entity)** Add a Traget to the current Link.
public void	**createLink()** Creates a new Link with the values entered in the Fields of this LinkView.
public void	**deleteSelectedSource()** Deletes the selected Source in the Sources list.
public void	**deleteSelectedTarget()** Deletes the selected Target in the Targets list.
public boolean	**linkIsOk()** Checks whether at least one Source and one Target are specified.
public void	**saveLink()** Saves all values of the fields of this LinkView.
public void	**setLink(org.ximtec.iserver.core.Link link)** Fills all Fields with the values of a link.

authoring.view.MainView

public *MainView*
extends JFrame

The main JFrame that keeps everyting togehter. Consists of the Navigator, the PageView, the LayersView, the UsersView and all the Menues.

Field Summary

Type	Description
protected DefaultBound-edRangeModel	**sizeModel** Model for the Slider to adjust presentation size of the page that is displayed.

Constructor Summary

Description
MainView(java.lang.String string, authoring.model.MainModel mainModel) Creates a new instance of MainView.

Method Summary

Returns	Description
public JDesktopPane	**getDesktop()**
public Frame	**getFrame()**
public MainModel	**getMainModel()**
public PageView	**getPageView()**
public DefaultBound-edRangeModel	**getSizeModel()**
public void	**loadMainView()** Creates all Views that are part of the MainView and initial-izes them.

authoring.view.MultiplePageEditView

public *MultiplePageEditView*
extends JPanel

A JPanel with Fields for all values needed to add multiple pages to a document.

Constructor Summary

Description
MultiplePageEditView(authoring.model.DocumentModel documentModel, authoring.view.Navigator navigator) Creates a new instance of MultiplePageEditView and all Swing Elements of the View.

Method Summary

Returns	Description
public DocumentModel	**getDocumentModel()**
public int	**getHeightFieldValue()**
public String	**getNameFieldValue()**
public int	**getNumberFromFieldValue()**
public int	**getNumberToFieldValue()**
public int	**getWidthFieldValue()**

authoring.view.Navigator

public *Navigator*
extends JPanel
implements Observer

A JPanel that shows all documents, pages and shapes of the loaded PaperPP application. It allows to browse the application and to add and delete documents and pages.

Constructor Summary

Description
Navigator(authoring.view.MainView mainView, authoring.model.MainModel mainModel) Creates a new instance of Navigator and initialzes the construction of the document/page/shape tree.

Method Summary

Returns	Description
public PageModel	**getLoadedPageModel()**
public MainModel	**getMainModel()**
public MainView	**getMainView()**
public JTree	**getTree()**
public void	**setLoadedPageModel(authoring.model.PageModel loadedPageModel)**
public void	**update(java.util.Observable o, java.lang.Object arg)** This method is called whenever the observed object is changed.

authoring.view.PageEditView

public *PageEditView*
extends JPanel

A JPanel with Fields for all values needed to create a new Page.

Constructor Summary

Description
PageEditView(authoring.model.DocumentModel documentModel, authoring.view.Navigator navigator) Creates a new instance of PageEditView and all Swing Elements of the View.

Method Summary

Returns	Description
public DocumentModel	**getDocumentModel()**
public int	**getHeightFieldValue()**
public String	**getNameFieldValue()**
public int	**getNumberFieldValue()**
public int	**getWidthFieldValue()**

authoring.view.PageView

public *PageView*
extends JLayeredPane
implements Observer

A JLayeredPane that shows the loaded Page and all its Shapes. The JLayeredPane proviedes comfortable functionality for the presentation of the Layers.

Constructor Summary

Description
PageView(authoring.view.MainView mainView, javax.swing.BoundedRangeModel m) Creates a new instance of PageView

Method Summary

Returns	Description
public void	**displayShapeModel(authoring.model.ShapeModel shapeModelToCreateView)** Displays a Shape on the loaded Page.
public void	**emptyDragLayer()** Removes all elements from the Drag Layer.
public int	**getDragMode()** Returns current dragMode.
public MainView	**getMainView()**
public PageModel	**getPageModel()**
public ShapeView	**getShapeInFocus()**
public void	**imageSizeChanged(double multiplier)** Repaints the whole view when the size has been changed.
public void	**openPage(authoring.model.PageModel pageModel)** Loads a new Page into the PageView.
public void	**removeShape(authoring.model.ShapeModel shapeModel)**
public static float	**scaleToFitDB(float x)** Converts [px] measure of PageView to [mm] measure of DB
public static float	**scaleToFitPage(float x)** Converts [mm] measure of DB to [px] measure of PageView.
public void	**setDraggingShapeView(authoring.view.ShapeView movingShapeView, int xMoveScaled, int yMoveScaled)** Sets a Shape during Dragging into the JLayeredPane.DRAG_LAYER at the right position.
public void	**setDragMode(int dragMode)** Sets pageView.drapMode and adjusts the Cursor accordingly.
public void	**setFocusOnShape(authoring.model.ShapeModel shapeModel)** Sets the focus on the Shape

Returns	Description
public void	**setNewDragCircle(int upperLeftX, int upperLeftY, int xMoveScaled, int yMoveScaled)** Creates a new ChangingShapeView of a Shape that is beeing created at the Moement.
public void	**setNewDragEllipse(int upperLeftX, int upperLeftY, int xMoveScaled, int yMoveScaled)** See setNewDragCircle
public void	**setNewDragPolygon(java.util.Vector polygonXPoints, java.util.Vector polygonYPoints)** setNewDragCircle
public void	**setNewDragRectangle(int upperLeftX, int upperLeftY, int xMoveScaled, int yMoveScaled)** setNewDragCircle
public void	**update(java.util.Observable o, java.lang.Object arg)** This method is called whenever the observed object is changed.

authoring.view.PointEditView

public *PointEditView*
extends JPanel

A JPanel with Fields for all values (= x & y) that define a point

Constructor Summary

Description
PointEditView(authoring.view.PolygonEditView polygonEditView, boolean isMutate) Creates a new instance of PointEditView

Method Summary

Returns	Description
public Point	**getPoint()**
public Poly-gonEditView	**getPolygonEditView()**

Returns	Description
public Integer	**getXFieldValue()**
public Integer	**getYFieldValue()**
public void	**setPoint(org.sigtec.om.graphics2D.Point point)**

authoring.view.PolygonEditView

public *PolygonEditView*
extends ShapeEditView

A JPanel with Fields for all values that define a polygon.

Constructor Summary

Description
PolygonEditView() reates a new instance of PolygonEditView.
PolygonEditView(boolean isMutate) Creates a new instance of PolygonEditView with specified isMutate.

Method Summary

Returns	Description
public void	**addNewPoint()** Opens a dialog to enter the data of a new Point.
public void	**addNewPoint(java.lang.Integer x, java.lang.Integer y)** Adds a new Point to the list.
public void	**buildMe(boolean isMutate)** Creates all Swing Elements of the View.
public void	**deleteSelectedPoint()**
public Individual	**getCreatorBoxValue()**
public Layer	**getLayerBoxValue()**

Returns	Description
public String	**getNameFieldValue()**
public OMCollection	**getNewCollectionOfPoints()** Creates a new OMCollection and adds the Points in the list of the current view.
public OMCollection	**getNewPoints()** Returns an OMCollection filled with the Points in the list of the current view.
public Page	**getPage()**
public void	**mutatePoint(org.sigtec.om.graphics2D.Point point, java.lang.Integer x, java.lang.Integer y)** Modifies an existing Point.
public void	**setModel(authoring.model.ShapeModel model)** Sets a PolygonModel and fills all fields according to this existing polygon.
public void	**setPoints(java.util.Vector polygonXPoints, java.util.Vector polygonYPoints)**

authoring.view.PolygonView

public *PolygonView*
extends ShapeView

This Class represents an 'actual' polygon one can see on the PaperPP pages.

Constructor Summary

Description
PolygonView() Creates a new instance of PolygonView without any ShapeModel.
PolygonView(authoring.model.PolygonModel polygonModel) Creates a new instance of PolygonView with a PolygonModel.

Method Summary

Returns	Description
public void	**paint(java.awt.Graphics graphics)** Paints the Polygon onto the Screen!

Returns	Description
protected void	**sizePanel()** Whenever the size of the Shape itself has changed, this method resizes the size of the JPanel.

authoring.view.RectangleEditView

public *RectangleEditView*
extends ShapeEditView

A JPanel with Fields for all values that define a rectangle.

Constructor Summary

Description
RectangleEditView() Creates a new instance of RectangleEditView.
RectangleEditView(boolean isMutate) Creates a new instance of RectanlgeEditView with specified isMutate.

Method Summary

Returns	Description
public void	**buildMe(boolean isMutate)** Creates all Swing Elements of the View.
public Individual	**getCreatorBoxValue()**
public Integer	**getHeightFieldValue()**
public Layer	**getLayerBoxValue()**
public String	**getNameFieldValue()**
public Page	**getPage()**
public Integer	**getUpperLeftXFieldValue()**
public Integer	**getUpperLeftYFieldValue()**

Returns	Description
public Integer	**getWidthFieldValue()**
public void	**setHeightFieldValue(int height)**
public void	**setModel(authoring.model.ShapeModel model)** Sets a RectangleModel and fills all fields according to this existing rectangle.
public void	**setUpperLeftXFieldValue(int upperLeftXFieldValue)**
public void	**setUpperLeftYFieldValue(int upperLeftYFieldValue)**
public void	**setWidthtFieldValue(int width)**

authoring.view.RectangleView

public *RectangleView*
extends ShapeView

This Class represents an 'actual' rectangle one can see on the PaperPP pages.

Constructor Summary

Description
RectangleView() Creates a new instance of RectangleView without any ShapeModel.
RectangleView(authoring.model.RectangleModel rectangleModel) Creates a new instance of RectangleView with a RectangleModel.

Method Summary

Returns	Description
public void	**paint(java.awt.Graphics graphics)** Paints the Rectangle onto the Screen!

Returns	Description
protected void	**sizePanel()** Whenever the size of the Shape itself has changed, this method resizes the size of the JPanel.

authoring.view.RessourceEditView

public *RessourceEditView*
extends JPanel

A JPanel with Fields for all values that define a Ressource (Text, Webpage, Image or Movie).

Constructor Summary

Description
RessourceEditView(java.lang.String s, authoring.view.PageView pageView, authoring.view.LinkView linkView, java.lang.String addTo) Creates a new instance of RessourceEditView and creates the Swing Elements depending on what kind of Ressource is to be created.

Method Summary

Returns	Description
public String	**getContentField()**
public String	**getDescriptionFieldValue()**
public String	**getNameFieldValue()**
public String	**getUrlFieldValue()**

authoring.view.ShapeEditView

public abstract *ShapeEditView*
extends JPanel
implements ShapeInterface

This class provides a skeletal implementation of the ShapeInterface, to minimize the effort required to implement this interface. See ShapeView and ChangingShapeView as well. ShapeEditViews provide masks to change all values a Shape consists of.

Field Summary

Type	Description
protected ShapeModel	**model** The model of the Shape this view is providing a GUI for.
protected PageView	**pageView** The View of the Page the Shape lies on...

Constructor Summary

Description
ShapeEditView()

Method Summary

Returns	Description
public ShapeModel	**getModel()**
public PageView	**getPageView()**
public void	**setModel(authoring.model.ShapeModel model)**
public void	**setPageView(authoring.view.PageView pageView)**

authoring.view.ShapeInterface

public interface *ShapeInterface*

The root interface in the *shape view hierarchy.* Only states, that a ShapeView is linked to a ShapeModel.

Method Summary

Returns	Description
public ShapeModel	**getModel()**
public void	**setModel(authoring.model.ShapeModel model)**

authoring.view.ShapeView

public abstract *ShapeView*
extends JPanel
implements ShapeInterface

This class provides a skeletal implementation of the ShapeInterface, to minimize the effort required to implement this interface. See ShapeEditView and ChangingShapeView as well. ShapeViews paint a Shape in the geometry of two dimensions.

Field Summary

Type	Description
protected boolean	**focus** whether a current focus is set onto this view.
protected ShapeModel	**model** The model of the Shape this view is providing a GUI for.
protected Color	**myColor** The current color of the Shape.

Constructor Summary

Description
ShapeView()

Method Summary

Returns	Description
public ShapeModel	**getModel()**
public void	**setFocus(boolean focus)**
public void	**setModel(authoring.model.ShapeModel model)**
protected abstract void	**sizePanel()** Whenever the size of the Shape itself has changed, this method resizes the size of the JPanel.

authoring.view.UsersView

public *UsersView*
extends JPanel

A JPanel that shows all Users (Individuals and Groups) of the loaded PaperPP application. It uses a JTree for clear presentation.

Constructor Summary

Description
UsersView(authoring.view.MainView mainView, authoring.model.MainModel mainModel) Creates a new instance of UsersView an creates all Swing elements.

Method Summary

Returns	Description
public MainView	**getMainView()**
public void	**update()** Updates the view upon change to any Users

C.3 Package authoring.model

authoring.model.CircleModel

public *CircleModel*
extends ShapeModel

Model for a circle. Contains data and functionality for a circle.

Constructor Summary

Description
CircleModel() Creates a new instance of CircleModel
CircleModel(graphics2D.Shape shape) Creates a new instance of CircleModel.

Method Summary

Returns	Description
public void	**deleteShape()**
public Integer	**getCentreX()**
public Integer	**getCentreY()**
public ImageIcon	**getIcon16()**
public Shape	**getModel()**
public Integer	**getRadius()**
public String	**getShapeName()**
public void	**move(int x, int y)** Move the Shape x to the right and y down.
public void	**setCentre(java.lang.Integer x, java.lang.Integer y)**
public void	**setModel(graphics2D.Shape model)**

Returns	Description
public void	**setRadius(java.lang.Integer radius)**
public void	**setShapeName(java.lang.String name)**
public String	**toString()**

authoring.model.ComplexShapeModel

public *ComplexShapeModel*
extends ShapeModel

Functionality for Complex Shapes is not provided yet.

Constructor Summary

Description
ComplexShapeModel() Creates a new instance of ComplexShapeModel
ComplexShapeModel(graphics2D.Shape shape)

Method Summary

Returns	Description
public void	**deleteShape()**
public ImageIcon	**getIcon16()**
public Shape	**getModel()**
public String	**getShapeName()**
public void	**move(int x, int y)**
public void	**setModel(graphics2D.Shape model)**
public void	**setShapeName(java.lang.String name)**

Returns	Description
public String	**toString()**

authoring.model.DocumentModel

public *DocumentModel*
extends Observable
implements Observer,Decorator

DocumentModel contains all data of a Document of a PaperPP application.

Constructor Summary

Description
DocumentModel(org.ximtec.paperpp.core.Document document, authoring.model.MainModel mainModel) Creates a new instance of DocumentModel

Method Summary

Returns	Description
public void	**addNewPage(java.lang.String name, int number, int width, int height, authoring.view.Navigator navigator)** Add new page to the document.
public void	**deleteAllPagesOfDocument()** Delete all the Pages of a Document, all Shapes on these pages and all corresponding ShapeModels.
public void	**deleteDocument()** Deletes a Document only from the underlaying OMSJava.
public Document	**getDocument()**
public String	**getDocumentName()**
public ImageIcon	**getIcon16()**
public MainModel	**getMainModel()**

Returns	Description
public String	**getName()**
public Vector	**getPages()**
public void	**removePageModel(authoring.model.PageModel pageModel)** Removes a Page, the PageModel, all Shapes on that page and all corresponding ShapeModels.
public String	**toString()**
public void	**update(java.util.Observable o, java.lang.Object arg)** This method is called whenever the observed object is changed.

authoring.model.EllipseModel

public *EllipseModel*
extends ShapeModel

Model for an ellipse. Contains data and functionality for an ellipse.

Constructor Summary

Description
EllipseModel() Creates a new instance of EllipseModel
EllipseModel(graphics2D.Shape shape) Creates a new instance of EllipseModel.

Method Summary

Returns	Description
public void	**deleteShape()**
public Integer	**getCentreX()**
public Integer	**getCentreY()**
public Integer	**getHeight()**

Returns	Description
public ImageIcon	**getIcon16()**
public Shape	**getModel()**
public String	**getShapeName()**
public Integer	**getWidth()**
public void	**move(int x, int y)** Move the Shape x to the right and y down.
public void	**setCentre(java.lang.Integer x, java.lang.Integer y)**
public void	**setHeight(java.lang.Integer height)**
public void	**setModel(graphics2D.Shape model)**
public void	**setShapeName(java.lang.String name)**
public void	**setWidth(java.lang.Integer width)**
public String	**toString()**

authoring.model.GroupModel

public *GroupModel*
extends UserModel

Model for a Group containing all its data and functionality. Subclass of UserModel. See UserModel and IndividualModel.

Constructor Summary

Description
GroupModel(org.ximtec.iserver.core.Group group) Creates a new instance of GroupModel

Method Summary

Returns	Description
public void	**addMember(org.ximtec.iserver.core.User user)** Adds a new user (group or individual) to the Group.
public void	**deleteUser(authoring.view.UsersView usersView)** Deletes this Group and the GroupModel.
public String	**getDescription()**
public ImageIcon	**getIcon16()**
public String	**getName()**
public void	**setDescription(java.lang.String description)**
public void	**setName(java.lang.String name)**
public String	**toString()**

authoring.model.IndividualModel

public *IndividualModel*
extends UserModel

Model for an Individual containing all its data and functionality. Subclass of UserModel. See UserModel and GroupModel.

Constructor Summary

Description
IndividualModel(org.ximtec.iserver.core.Individual individual) Creates a new instance of IndividualModel

Method Summary

Returns	Description
public void	**deleteUser(authoring.view.UsersView usersView)** Deletes this Group and the GroupModel.

Returns	Description
public String	**getDescription()**
public ImageIcon	**getIcon16()**
public String	**getLogin()**
public String	**getName()**
public String	**getPasswd()**
public void	**setDescription(java.lang.String description)**
public void	**setLogin(java.lang.String login)**
public void	**setName(java.lang.String name)**
public void	**setPasswd(java.lang.String passwd)**
public String	**toString()**

authoring.model.LayerModel

public *LayerModel*
extends Observable

LayerModel contains all data of a Layer of a PaperPP application.

Constructor Summary

Description
LayerModel(org.ximtec.iserver.core.Layer layer, authoring.model.MainModel mainModel) Creates a new instance of LayerModel.

Method Summary

Returns	Description
public void	**deleteLayer()**
public Layer	**getLayer()**
public MainModel	**getMainModel()**
public String	**getName()**
public Vector	**getShapeModels()** Retruns a Vector of all ShapeModels on this Layer.
public boolean	**isSelected()** Retruns bool whether the Layer is currently selected.
public void	**moveDown()**
public void	**moveUp()** Moves the Layer one position up.
public void	**setName(java.lang.String name)**
public void	**setSelected(boolean b)** Sets the Layer selected if bool=true, not_selected if bool=false.

authoring.model.MainModel

public *MainModel*
extends Observable
implements Observer

The Main Model models a whole PaperPP Application. It provides all data including documents and layers for MainView.

Constructor Summary

Description
MainModel() Constructs a MainModel initialized with DEFAULT_NAME and DEFAULT_ROOT.

Description
MainModel(java.lang.String applicationName, java.lang.String applicationRoot) Constructs a MainModel that is initialized with applicationName and applicationRoot.

Method Summary

Returns	Description
public void	**addNewDocument(java.lang.String name, java.lang.String id, int width, int height, java.lang.String content)** Adds a new Document to the PaperPP application.
public void	**addNewLayer(java.lang.String name)** Adds a new Layer to the PaperPP application.
public void	**deleteLayer(authoring.model.LayerModel layerModel)** Removes a Layer from the PaperPP application.
public Vector	**getDocuments()** Returns the documents Vector.
public static int	**getLayerIndexByLayer(org.ximtec.iserver.core.Layer layer)** Returns the index of a certain Layer
public static LayerModel	**getLayerModelByLayer(org.ximtec.iserver.core.Layer layer)** Returns the LayerModel of the Layer.
public List	**getLayers()** Returns layers List.
public static int	**getNumberOfLayers()** Returns the number of Layers contained in the PaperPP application.
public String	**getSiteName()**
public OMWorkspace-	**getWorkspace()**
public void	**loadDB()** Loads an OMS Java DB of a PaperPP application into the MainModel.

Returns	Description
public void	**loadDB(java.lang.String applicationName, java.lang.String applicationRoot)** Loads am OMS Java DB of a PaperPP application into the MainModel.
public void	**loadSite()** Loads documents, layers, shapes, ... into the MainModel.
public void	**moveLayerDown(org.ximtec.iserver.core.Layer layer)** Moves a Layer one position down.
public void	**moveLayerUp(org.ximtec.iserver.core.Layer layer)** Moves a Layer one position up.
public void	**removeDocumentModel(authoring.model.DocumentModel documentModel)** Removes a Document and all contained Pages and Shapes.
public void	**update(java.util.Observable o, java.lang.Object arg)** This method is called whenever the observed object is changed.

authoring.model.PageModel

public *PageModel*
extends Observable
implements Observer,Comparable,Decorator

PageModel contains all data of a Page of a PaperPP application. It provides all data for PageView.

Constructor Summary

Description
PageModel(org.ximtec.paperpp.core.Page page, authoring.model.DocumentModel documentModel) Constructs a PageModel and all ShapeModels contained in this Page.

Method Summary

Returns	Description
public void	**addNewShapeModel(authoring.model.ShapeModel shapeModel)** Whenever a new Shape is created on this Page, this method adds the corresponding ShapeModel to the PageModel.

Returns	Description
public int	**compareTo(java.lang.Object o)**
public void	**deleteAllShapesOnPage()** Deletes all the Shapes on this Page.
public void	**deletePage()** Deletes the Page of the PageModel from the PaperPP DB.
public DocumentModel	**getDocumentModel()**
public Integer	**getHeight()**
public ImageIcon	**getIcon16()**
public BufferedImage	**getImage()**
public Page	**getPage()**
public Vector	**getShapes()**
public void	**removeShapeModel(authoring.model.ShapeModel shapeModel)** Removes a ShapeModel from the PageModel.
public String	**toString()**
public void	**update(java.util.Observable o, java.lang.Object arg)** This method is called whenever the observed object is changed.

authoring.model.PolygonModel

public *PolygonModel*
extends ShapeModel

Model for a polygon. Contains data and functionality for a polygon.

Constructor Summary

Description
PolygonModel() Creates a new instance of PolygonModel
PolygonModel(graphics2D.Shape shape) Creates a new instance of RectangleModel.

Method Summary

Returns	Description
public void	**deleteShape()**
public ImageIcon	**getIcon16()**
public Shape	**getModel()**
public OMCollection	**getPoints()**
public String	**getShapeName()**
public void	**move(int x, int y)** Moves the polygon (= all points of the polygon) x to the right and y down.
public void	**setModel(graphics2D.Shape model)**
public void	**setPoints(org.xdatabase.om.OMCollection omColl)**
public void	**setShapeName(java.lang.String name)**
public String	**toString()**

authoring.model.RectangleModel

public *RectangleModel*
extends ShapeModel

Model for a rectangle. Contains data and functionality for a rectangle.

Constructor Summary

Description
RectangleModel() Creates a new instance of RectangleModel
RectangleModel(graphics2D.Shape shape) Creates a new instance of RectangleModel.

Method Summary

Returns	Description
public void	**deleteShape()**
public Integer	**getHeight()**
public ImageIcon	**getIcon16()**
public Shape	**getModel()**
public String	**getShapeName()**
public Integer	**getUpperLeftX()** X-axis starts at the upperleft an goes to the right.
public Integer	**getUpperLeftY()** Y-axis starts th the upperleft an goes down.
public Integer	**getWidth()**
public void	**move(int x, int y)** Move the Shape x to the right and y down.
public void	**setHeight(java.lang.Integer h)**
public void	**setModel(graphics2D.Shape model)**
public void	**setShapeName(java.lang.String name)**
public void	**setUpperLeft(java.lang.Integer x, java.lang.Integer y)**
public void	**setWidth(java.lang.Integer w)**

Returns	Description
public String	**toString()**

authoring.model.ShapeModel

public abstract *ShapeModel*
extends Observable
implements Observer,Cloneable,Decorator

Abstract Class modelling a general Shape and providing certain general functionality including the handling of Layers and an update method for the MVC pattern. Also specifies few abstract methods that must be implemented by any Subclass of ShapeModel.

Constructor Summary

Description
ShapeModel()

Method Summary

Returns	Description
public abstract void	**deleteShape()**
public Color	**getLayerColor()** Returns the current color of the Layer that Shape lies on.
public int	**getLayerIndex()** Returns the index of the Layer that Shape lies on.
public LayerModel	**getLayerModel()** Returns the Model of the Layer that Shape lies on.
public abstract Shape	**getModel()**
public abstract String	**getShapeName()**

Returns	Description
public abstract void	**move(int x, int y)**
public abstract void	**setModel(graphics2D.Shape model)**
public abstract void	**setShapeName(java.lang.String name)**
public void	**update(java.util.Observable o, java.lang.Object arg)** MVC pattern.

authoring.model.UserModel

public abstract *UserModel*
extends Object
implements Decorator

Abstract Model for a user that can either be a Individual or a Group. The composite design pattern is applied. A group can contain an arbitrary number of users.

Constructor Summary

Description
UserModel()

Method Summary

Returns	Description
public abstract void	**deleteUser(authoring.view.UsersView usersView)** To delete this user.

C.4 Package authoring.event

authoring.event.CircleEditViewButtonListener

public *CircleEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of CircleEditView when modifying an existing Circle.

Constructor Summary

Description
CircleEditViewButtonListener(authoring.view.CircleEditView circleEditView) Creates a new instance of CircleEditViewButtonListener and sets CircleEditView.
CircleEditViewButtonListener() Creates a new instance of CircleEditViewButtonListener

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the CircleEditView.
public void	**setCircleEditView(authoring.view.CircleEditView circleEditView)** Sets CircleEditView.

authoring.event.CreateDMLAction

public *CreateDMLAction*
extends AbstractAction

Action to create a DML-File of the current PaperPP application.

Constructor Summary

Description
CreateDMLAction(authoring.view.PageView pageView) Creates a new instance of CreateDMLAction

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Action creates a DML-File of the current PaperPP application.

authoring.event.CreateDumpAction

public *CreateDumpAction*
extends AbstractAction

ction to create a Dump-File of the current PaperPP application.

Constructor Summary

Description
CreateDumpAction(authoring.view.PageView pageView) Creates a new instance of CreateDumpAction

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Action creates a Dump-File of the current PaperPP application.

authoring.event.DeleteShapeAction

public *DeleteShapeAction*
extends AbstractAction

Action that deletes the Shape that is currently in focus.

Constructor Summary

Description
DeleteShapeAction(authoring.view.PageView pageView) Creates a new instance of DeleteShapeAction and sehts PageView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Action deletes the Shape that is currently in focus.

authoring.event.EllipseEditViewButtonListener

public *EllipseEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of EllipseEditView when modifying an existing Ellipse.

Constructor Summary

Description
EllipseEditViewButtonListener(authoring.view.EllipseEditView ellipseEditView) Creates a new instance of EllipseEditViewButtonListener and sets EllipseEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the EllipseEditView.

authoring.event.GroupEditViewButtonListener

public *GroupEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of GroupEditView when modifying an existing Group.

Constructor Summary

Description
GroupEditViewButtonListener(authoring.view.GroupEditView groupEditView) Creates a new instance of GroupEditViewButtonListener and sets GroupEditView.

Method Summary

Returns	Description
public void	**mousePressed(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the GroupEditView.

authoring.event.IndividualEditViewButtonListener

public *IndividualEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of IndividualEditView when modifying an existing Individual.

Constructor Summary

Description
IndividualEditViewButtonListener(authoring.view.IndividualEditView individualEditView) Creates a new instance of IndividualEditViewButtonListener and sets IndividualEditView.

Method Summary

Returns	Description
public void	**mousePressed(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the IndividualEditView.

authoring.event.LayerEditViewButtonListener

public *LayerEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of LayerEditView when modifying an existing Layer.

Constructor Summary

Description
LayerEditViewButtonListener(authoring.view.LayerEditView layerEditView) Creates a new instance of LayerEditViewButtonLisener and sets LayerEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the LayerEditView.

authoring.event.LayersListSelectionListener

public *LayersListSelectionListener*
extends Object
implements ListSelectionListener

ListSelectionListener.valueChanged() is called whenever the Selection of ListItems change. This listener is for LayersView and makes sure that the selected layers are visible and the unselected are not.

Constructor Summary

Description
LayersListSelectionListener()

Method Summary

Returns	Description
public void	**valueChanged(javax.swing.event.ListSelectionEvent e)** Sets the selected Layers to selected and for this reason to visible. (..and otherwise)

authoring.event.LayersViewButtonListener

public *LayersViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of LayersView.

Constructor Summary

Description
LayersViewButtonListener(authoring.view.LayersView layersView, authoring.view.MainView mainView) Creates a new instance of LayersViewButtonListener and sets LayersView and MainView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the LayersView.

authoring.event.LayersViewPopupMenuListener

public *LayersViewPopupMenuListener*
extends Object
implements ActionListener

ActionListener that handles all Events of JPopupMenu that is constructed upon right-click on a layer entry in LayersView.

Constructor Summary

Description
LayersViewPopupMenuListener(authoring.model.LayerModel layerModel, authoring.view.LayersView layersView) Creates a new instance of LayersViewPopupMenuListener and sets LayerModel and LayersView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when e.g. a mouse button has been clicked on the JPoupupMenu.

authoring.event.LinkAction

public *LinkAction*
extends AbstractAction

Action to open LinksView for the selected Shape.

Constructor Summary

Description
LinkAction(authoring.view.PageView pageView) Creates a new instance of LinkAction and sets PageView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Action opens LinksView for the selected Shape.

authoring.event.LinkEditViewButtonListener

public *LinkEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of LinkEditView when modifying an existing Link.

Constructor Summary

Description
LinkEditViewButtonListener(authoring.view.LinkView linkView, authoring.view.PageView pageView) Creates a new instance of LinkEditViewButtonListener and sets LinkView and PageView.

Method Summary

Returns	Description
public LinkView	**getLinkView()**
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the LinkEditView.

authoring.event.LinksViewButtonListener

public *LinksViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of LinksView.

Constructor Summary

Description
LinksViewButtonListener(authoring.view.LinksView linksView, authoring.view.PageView pageView) Creates a new instance of LinksViewButtonListener and sets LinksView and PageView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the LinksView.

authoring.event.LoadDMLAction

public *LoadDMLAction*
extends AbstractAction

Action to load a DML-File of a PaperPP application into the AuthoringTool.

Constructor Summary

Description
LoadDMLAction(authoring.view.PageView pageView) Creates a new instance of LoadDMLAction and sets PageView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Opens a JFileChooser and tries to load the selected DML-File.

authoring.event.LoadDumpAction

public *LoadDumpAction*
extends AbstractAction

Action to load a Dump-File of a PaperPP application into the Authoring-Tool.

Constructor Summary

Description
LoadDumpAction(authoring.view.PageView pageView) Creates a new instance of LoadDumpAction and sets PageView.

Method Summary

Returns	**Description**
public void	**actionPerformed(java.awt.event.ActionEvent e)** Opens a JFileChooser and tries to load the selected Dump-File.

authoring.event.MyTreeMouseAdapter

public *MyTreeMouseAdapter*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of Navigator.

Constructor Summary

Description
MyTreeMouseAdapter(authoring.view.Navigator navigator) Creates a new instance of MyTreeMouseAdapter and stes Navigator.

Method Summary

Returns	**Description**
public void	**mousePressed(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the Navigator.

authoring.event.NavigatorViewPopupMenuListener

public *NavigatorViewPopupMenuListener*
extends Object
implements ActionListener

ActionListener that handles all Events of JPopupMenu that is constructed upon right-click on an entry in Navigator.

Constructor Summary

Description
NavigatorViewPopupMenuListener(java.lang.Object userObject, authoring.view.Navigator navigator) Creates a new instance of NavigatorViewPopupMenuListener and sets the clicked Object and Navigator.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when e.g. a mouse button has been clicked on the JPoupupMenu.

authoring.event.NewCircleAction

public *NewCircleAction*
extends AbstractAction

Action to create a new Cirlce by entering the necessary values into a CirlceEditView.

Constructor Summary

Description
NewCircleAction(authoring.view.PageView pageView) Creates a new instance of NewCircleAction and sets PageView

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewCircleEditViewButtonListener

public *NewCircleEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of CircleEditView when creating a new Circle.

Constructor Summary

Description
NewCircleEditViewButtonListener(authoring.view.CircleEditView circleEditView) Creates a new instance of NewCircleEditViewButtonListener and sets CircleEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the CircleEditView.

authoring.event.NewDocumentEditViewButtonListener

public *NewDocumentEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of DocumentEditView when creating a new Document.

Constructor Summary

Description
NewDocumentEditViewButtonListener(authoring.view.DocumentEditView documentEditView) Creates a new instance of NewDocumentEditViewButtonListener and sets DocumentEditView.

Method Summary

Returns	**Description**
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the DocumentEditView.

authoring.event.NewDragCircleAction

public *NewDragCircleAction*
extends AbstractAction

Action to start creation process of a new Cirlce by mouse-dragging on the PageView.

Constructor Summary

Description
NewDragCircleAction(authoring.view.PageView pageView) Creates a new instance of NewDragCircleAction and sets PageView.

Method Summary

Returns	**Description**
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewDragEllipseAction

public *NewDragEllipseAction*
extends AbstractAction

Action to start creation process of a new Ellipse by mouse-dragging on the PageView.

Constructor Summary

Description
NewDragEllipseAction(authoring.view.PageView pageView) Creates a new instance of NewDragEllipseAction and sets PageView.

Method Summary

Returns	**Description**
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewDragPolygonAction

public *NewDragPolygonAction*
extends AbstractAction

Action to start creation process of a new Poygon by click-creating the edges on the PageView.

Constructor Summary

Description
NewDragPolygonAction(authoring.view.PageView pageView) Creates a new instance of NewDragPolygonAction and sets PageView.

Method Summary

Returns	**Description**
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewDragRectangleAction

public *NewDragRectangleAction*
extends AbstractAction

Action to start creation process of a new Rectangle by mouse-dragging on the PageView.

Constructor Summary

Description
NewDragRectangleAction(authoring.view.PageView pageView) Creates a new instance of NewDragRectangleAction and sets PageView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewEllipseAction

public *NewEllipseAction*
extends AbstractAction

Action to create a new Ellipse by entering the necessary values into a EllipseEditView.

Constructor Summary

Description
NewEllipseAction(authoring.view.PageView pageView) Creates a new instance of NewEllipseAction and sets PageView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewEllipseEditViewButtonListener

public *NewEllipseEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of EllipseEditView when creating a new Ellipse.

Constructor Summary

Description
NewEllipseEditViewButtonListener(authoring.view.EllipseEditView ellipseEditView) Creates a new instance of NewEllipseEditViewButtoListener and sets EllipseEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the EllipseEditView.

authoring.event.NewGroupEditViewButtonListener

public *NewGroupEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of GroupEditView when creating a new Group.

Constructor Summary

Description
NewGroupEditViewButtonListener(authoring.view.GroupEditView groupEditView, java.lang.Object groupModel) Creates a new instance of NewGroupEditViewButtonListener and sets GroupEditView and GroupModel.

Method Summary

Returns	Description
public void	**mousePressed(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the GroupEditView.

authoring.event.NewIndividualEditViewButtonListener

public *NewIndividualEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of IndividualEditView when creating a new Individual.

Constructor Summary

Description
NewIndividualEditViewButtonListener(authoring.view.IndividualEditView individualEditView, java.lang.Object groupModel) Creates a new instance of NewIndividualEditViewButtonListener and sets IndividualEditView and GroupModel.

Method Summary

Returns	Description
public void	**mousePressed(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the IndividualEditView.

authoring.event.NewLayerEditViewButtonListener

public *NewLayerEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of LayerEditView when creating a new Layer.

Constructor Summary

Description
NewLayerEditViewButtonListener(authoring.view.LayerEditView layerEditView) Creates a new instance of NewLayerEditViewButtonLisener and sets LayerEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the LayerEditView.

authoring.event.NewLinkEditViewButtonListener

public *NewLinkEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of LinkView when creating a new Link.

Constructor Summary

Description
NewLinkEditViewButtonListener(authoring.view.LinkView linkView, authoring.view.PageView pageView) Creates a new instance of NewLinkEditViewButtonListener and sets LinkView and PageView.

Method Summary

Returns	Description
public LinkView	**getLinkView()**

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the LinkView.

authoring.event.NewMultiplePageEditViewButtonListener

public *NewMultiplePageEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of MultiplePageEditView when creating a new multiple pages.

Constructor Summary

Description
NewMultiplePageEditViewButtonListener(authoring.view.MultiplePageEditView multiplePageEditView, authoring.view.Navigator navigator) Creates a new instance of NewMultiplePageEditViewButtonListener and sets MultiplePageEditView and Navigator.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the MultiplePageEditView.

authoring.event.NewPageEditViewButtonListener

public *NewPageEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of PageEditView when creating a new page.

Constructor Summary

Description
NewPageEditViewButtonListener(authoring.view.PageEditView pageEditView, authoring.view.Navigator navigator) Creates a new instance of NewPageEditViewButtonListener and sets PageEditView and Navigator.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the PageEditView.

authoring.event.NewPointEditViewButtonListener

public *NewPointEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of PointEditView when creating a new point.

Constructor Summary

Description
NewPointEditViewButtonListener(authoring.view.PointEditView pointEditView) Creates a new instance of NewPointEditViewButtonListener and sets PointEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the PointEditView.

authoring.event.NewPolygonAction

public *NewPolygonAction*
extends AbstractAction

Action to create a new Polygon by entering the necessary values into a PolygonEditView.

Constructor Summary

Description
NewPolygonAction(authoring.view.PageView pageView) Creates a new instance of NewPolygonAction and sets PageView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewPolygonEditViewButtonListener

public *NewPolygonEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of PolygonEditView when creating a new Polygon.

Constructor Summary

Description
NewPolygonEditViewButtonListener(authoring.view.PolygonEditView polygonEditView) Creates a new instance of NewPolygonEditViewButtonListener and sets PolygonEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the PolygonEditView.

authoring.event.NewRectangleAction

public *NewRectangleAction*
extends AbstractAction

Action to create a new Rectangle by entering the necessary values into a RectangleEditView.

Constructor Summary

Description
NewRectangleAction(authoring.view.PageView pageView) Creates a new instance of NewRectangleAction and sets PageView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when an action occurs.

authoring.event.NewRectangleEditViewButtonListener

public *NewRectangleEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of RectangleEditView when creating a new Rectangle.

Constructor Summary

Description
NewRectangleEditViewButtonListener(authoring.view.RectangleEditView rectangleEditView) Creates a new instance of NewRectangleEditViewButtonListener and sets RectangleEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the RectangleEditView.

authoring.event.PageViewButtonListener

public *PageViewButtonListener*
extends MouseAdapter

MouseAdapter that handles MouseEvents of ToolBars of the PageView. TODO replace with Action.

Constructor Summary

Description
PageViewButtonListener(authoring.view.PageView pageView) Creates a new instance of PageViewButtonListener and sets PageView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on a ToolBar of the PageView.

authoring.event.PageViewMouseInputAdapter

public *PageViewMouseInputAdapter*
extends MouseInputAdapter

MouseInputAdapter that handles all MouseInput of PageView. This includes drag-creation of Shapes, Moving of Shapes as well as simple selection of Shapes.

Constructor Summary

Description
PageViewMouseInputAdapter(authoring.view.PageView pageView) Creates a new instance of PageViewMouseAdapter and sets PageView.

Method Summary

Returns	**Description**
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when the mouse button has been clicked (pressed and released) on PageView.
public void	**mouseDragged(java.awt.event.MouseEvent e)** Invoked when a mouse button is pressed on PageView and then dragged.
public void	**mouseEntered(java.awt.event.MouseEvent e)** Invoked when the mouse enters PageView.
public void	**mouseExited(java.awt.event.MouseEvent e)** Invoked when the mouse exits PageView.
public void	**mouseMoved(java.awt.event.MouseEvent e)** Invoked when the mouse cursor has been moved onto PageView but no buttons have been pushed.
public void	**mousePressed(java.awt.event.MouseEvent e)** Invoked when a mouse button has been pressed on Pageview.
public void	**mouseReleased(java.awt.event.MouseEvent e)** Invoked when a mouse button has been released on PageView.

authoring.event.PointEditViewButtonListener

public *PointEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of PointEditView when modifying an existing Point.

Constructor Summary

Description
PointEditViewButtonListener(authoring.view.PointEditView pointEditView) Creates a new instance of NewPointEditViewButtonListener and sets PointEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the PointEditView.

authoring.event.PolygonEditViewButtonListener

public *PolygonEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of PolygonEditView when modifying an existing Polygon.

Constructor Summary

Description
PolygonEditViewButtonListener(authoring.view.PolygonEditView polygonEditView) Creates a new instance of PolygonEditViewButtonListener and sets PolygonEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the PolygonEditView.

authoring.event.RectangleEditViewButtonListener

public *RectangleEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of RectangleEditView when modifying an existing Rectangle.

Constructor Summary

Description
RectangleEditViewButtonListener(authoring.view.RectangleEditView rectangleEditView) Creates a new instance of RectangleEditViewButtonListener and sets RectangleEditView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the RectangleEditView.

authoring.event.RessourceEditViewButtonListener

public *RessourceEditViewButtonListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of RessourceEditView when creating a new Ressource.

Constructor Summary

Description
RessourceEditViewButtonListener(authoring.view.RessourceEditView ressourceEditView, java.lang.String s, authoring.view.LinkView linkView, java.lang.String addTo) Creates a new instance of RessourceEditViewButtonListener

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the RessourceEditView.

authoring.event.SizeModelListener

public *SizeModelListener*
extends Object
implements ChangeListener

ChangeListener for the SizeModel that adjusts the presentation size of the PageView and is altered by the JSlider.

Constructor Summary

Description
SizeModelListener(authoring.view.PageView pageView) Creates a new instance of SizeModelListener and sets PageView.

Method Summary

Returns	Description
public void	**setPageView(authoring.view.PageView pageView)**
public void	**stateChanged(javax.swing.event.ChangeEvent e)** Invoked when the SizeModel changed its state.

authoring.event.SliderReadOutSynchronizer

public *SliderReadOutSynchronizer*
extends Object
implements ChangeListener

ChangeListener that adjusts the written value of the slider state.

Constructor Summary

Description
SliderReadOutSynchronizer(javax.swing.JLabel sliderReadOut, javax.swing.DefaultBoundedRangeModel sizeModel) Creates a new instance of SliderReadOutSynchronizer

Method Summary

Returns	Description
public void	**stateChanged(javax.swing.event.ChangeEvent e)** Invoked when the sizeModel changed its state.

authoring.event.UsersViewListener

public *UsersViewListener*
extends MouseAdapter

MouseAdapter that handles all MouseEvents of the UsersView. Generates PopupMenus when right-clicking on a user.

Constructor Summary

Description
UsersViewListener(authoring.view.UsersView usersView) Creates a new instance of UsersViewListener an sets UsersView.

Method Summary

Returns	Description
public void	**mouseClicked(java.awt.event.MouseEvent e)** Invoked when a mouse button has been clicked (pressed and released) on the UsersView.

authoring.event.UsersViewPopupMenuListener

public *UsersViewPopupMenuListener*
extends Object
implements ActionListener

ActionListener that handles all Events of JPopupMenu that is constructed upon right-click on a user in UsersView.

Constructor Summary

Description
UsersViewPopupMenuListener(java.lang.Object modelClicked, authoring.view.UsersView usersView) Creates a new instance of UsersViewPopupMenuListener and sets the clicked UserModel and the UsersView.

Method Summary

Returns	Description
public void	**actionPerformed(java.awt.event.ActionEvent e)** Invoked when e.g. a mouse button has been clicked on the JPoupupMenu.

C.5 Package authoring.util

authoring.util.Constant

public *Constant*
extends Object

Constants used in authoring.*

Field Summary

Type	Description
public static final String	**APPLICATION_WINDOW_NAME**
public static final String	**CIRCLE**
public static final String	**COLLECTION_NAME_CIRCLES**
public static final String	**COLLECTION_NAME_DOCUMENTS**
public static final String	**COLLECTION_NAME_ELLIPSES**
public static final String	**COLLECTION_NAME_PAGES**
public static final String	**COLLECTION_NAME_POLYGONS**
public static final String	**COLLECTION_NAME_RECTANGLES**
public static final Color[]	**COLOR_LAYER**
public static final String	**DEFAULT_FILE_PATH**
public static final String	**DOCUMENT**
public static final int	**DOUBLE_CLICK**
public static final int	**DRAG_CIRCLE**

Type	**Description**
public static final int	**DRAG_DISABLED**
public static final int	**DRAG_ELLIPSE**
public static final int	**DRAG_POLYGON**
public static final int	**DRAG_RECTANGLE**
public static final String	**ELLIPSE**
public static final String	**GROUP_TREE_LABEL**
public static final int	**HANDLE_BORDER_THICKNESS**
public static final String	**INDIV_TREE_LABEL**
public static final String	**INITIAL_PERCENTAGE**
public static final String	**JTREE_LINE_STYLE**
public static final String	**JTREE_LINE_STYLE_PROPERTY**
public static final String	**LOADING_DOCS**
public static final int	**MIN_MOVE_DIST**
public static final String	**NEW_IMAGE**

Type	Description
public static final String	**NEW_MOVIE**
public static final String	**NEW_TEXT**
public static final String	**NEW_WEBPAGE**
public static final String	**PERCENT**
public static final String	**POLYGON**
public static final String	**PROPERTIES_LOADED**
public static final String	**PROPERTIES_PATH**
public static final String	**RECTANGLE**
public static final int	**SINGLE_CLICK**
public static final String	**SPLASH_IMAGE_LOCATION**
public static final String	**WELCOME_IMAGE_LOCATION**

Constructor Summary

Description
Constructant()

authoring.util.Decorator

public interface *Decorator*

Method Summary

Returns	Description
public ImageIcon	**getIcon16()**

authoring.util.DecoratorMutableTreeNode

public *DecoratorMutableTreeNode*
extends DefaultMutableTreeNode

Constructor Summary

Description
DecoratorMutableTreeNode(java.lang.Object userObject, authoring.util.Decorator decorator)

Method Summary

Returns	Description
public Decorator	**getDecorator()**

authoring.util.Dictionary

public *Dictionary*
extends Object

A HashMap that contains (Shape, ShapeModel) pairs. It allows to find an existing authoring.model.ShapeModel by a graphics2D.Shape. See get-ShapeModelByShape(Shape shape)

Constructor Summary

Description
Dictionary()

Method Summary

Returns	Description
public static ShapeModel	**getShapeModelByShape(graphics2D.Shape shape)** Lookup in the Dictionary: return the ShapeModel of a Shape.
public static void	**putShape(graphics2D.Shape shape, authoring.model.ShapeModel shapeModel)** Puts a (Shape, ShapeModel) pair into the Dictionary
public static void	**removeShape(graphics2D.Shape shape)** Removes a (Shape, ShapeModel) pair from the Dictionary.

authoring.util.EntityContainer

public *EntityContainer*
extends Object

Wrapper Class for an Entity of the PaperPP application. The purpose is to provide a customized toString() method to generate List entries.

Constructor Summary

Description
EntityContainer(org.ximtec.iserver.core.Entity entity) Creates a new instance of EntityContainer

Method Summary

Returns	Description
public Entity	**getEntity()**

Returns	Description
public String	**toString()** Customized toString() method.

authoring.util.GUITool

public *GUITool*
extends Object

Helper functionality for the GUI.

Constructor Summary

Description
GUITool()

Method Summary

Returns	Description
public static JComboBox	**addCreatorComboBox(java.lang.String labelName, javax.swing.JPanel mainPanel, java.awt.GridBagLayout gridbag, java.awt.GridBagConstraints c)** Adds a JLabel and a JComboBox containing all Individuals of the PaperPP application to a JPanel with a GridBagLayout.
public static JTextField	**addLabeledTextField(java.lang.String labelName, java.lang.String defaultValue, javax.swing.JPanel mainPanel, java.awt.GridBagLayout gridbag, java.awt.GridBagConstraints c)** Adds a JLabel and a JTextField to a JPanel with a Grid-BagLayout.
public static JComboBox	**addLayerComboBox(java.lang.String labelName, javax.swing.JPanel mainPanel, java.awt.GridBagLayout gridbag, java.awt.GridBagConstraints c)** Adds a JLabel and a JComboBox containing all Layers of the PaperPP application to a JPanel with a GridBagLayout.
public static void	**disposeDialog(javax.swing.JPanel view)** Closes and disposes the JDialog which contains the view.

Returns	Description
public static Color	**getLayerColorByLayerIndex(int index)** Finds the Color of a Layer by its LayerIndex.
public static Component	**getParentJDialog(java.awt.Component component)** Returns a JDialog that is parent of the given Component.
public static Component	**getParentJInternalFrame(java.awt.Component component)** Returns a JInternalFrame that is parent of the given Component.
public static void	**setFocusInPageView(authoring.view.PageView pageView)** Sets the Focus into the PageView.
public static void	**showEditView(java.lang.String iFrameName, javax.swing.JPanel view, authoring.view.MainView mainView)** Opens a JDialog containing the JPanel view and displays it.
public static void	**showEditView(java.lang.String iFrameName, javax.swing.JPanel view, authoring.view.MainView mainView, int height)** Opens a JDialog containing the JPanel view and displays it.
public static void	**showEditView(java.lang.String iFrameName, javax.swing.JPanel view, authoring.view.MainView mainView, int x, int y, int width, int height)** Opens a JDialog containing the JPanel view and displays it.

authoring.util.HandleBorder

public *HandleBorder*
extends AbstractBorder

Nice border with 8 handles that is painted whenever a shape is in focus.

Constructor Summary

Description
HandleBorder() Creates a new instance of HandleBorder of gray color and standard thickness.
HandleBorder(java.awt.Color lineColor, int thickness) Creates a new instance of HandleBorder, customized.

Method Summary

Returns	Description
public void	**paintBorder(java.awt.Component component, java.awt.Graphics graphics, int x, int y, int width, int height)** Paints the Border.

authoring.util.IconTreeCellRenderer

public *IconTreeCellRenderer*
extends DefaultTreeCellRenderer

Constructor Summary

Description
IconTreeCellRenderer()

Method Summary

Returns	Description
public Component	**getTreeCellRendererComponent(javax.swing.JTree tree, java.lang.Object value, boolean selected, boolean expanded, boolean leaf, int row, boolean hasFocus)**

authoring.util.IndividualComboBoxListCellRenderer

public *IndividualComboBoxListCellRenderer*
extends Object
implements ListCellRenderer

This class implements the ListCellRenderer interface and defines how an entry for an Inidivual looks in a JComboBox.

Constructor Summary

Description
IndividualComboBoxListCellRenderer()

Method Summary

Returns	Description
public Component	**getListCellRendererComponent(javax.swing.JList list, java.lang.Object value, int index, boolean isSelected, boolean cellHasFocus)**

authoring.util.LayerComboBoxListCellRenderer

public *LayerComboBoxListCellRenderer*
extends Object
implements ListCellRenderer

This class implements the ListCellRenderer interface and defines how an entry for a Layer looks in a JComboBox.

Constructor Summary

Description
LayerComboBoxListCellRenderer()

Method Summary

Returns	Description
public Component	**getListCellRendererComponent(javax.swing.JList list, java.lang.Object value, int index, boolean isSelected, boolean cellHasFocus)**

authoring.util.LayersListCellRenderer

public *LayersListCellRenderer*
extends Object
implements ListCellRenderer

This class implements the ListCellRenderer interface and defines how an entry for a Layer looks in the LayersView.

Constructor Summary

Description
LayersListCellRenderer()

Method Summary

Returns	Description
public Component	**getListCellRendererComponent(javax.swing.JList list, java.lang.Object value, int index, boolean isSelected, boolean cellHasFocus)**

authoring.util.LinksListCellRenderer

public *LinksListCellRenderer*
extends Object
implements ListCellRenderer

This class implements the ListCellRenderer interface and defines how an entry for a Link looks in the LinksView.

Constructor Summary

Description
LinksListCellRenderer()

Method Summary

Returns	Description
public Component	**getListCellRendererComponent(javax.swing.JList list, java.lang.Object value, int index, boolean isSelected, boolean cellHasFocus)**

authoring.util.MenuTool

public *MenuTool*
extends Object

This is a Helper Class that generates JMenus and JMenuItems for the MainView of the PaperPP Authoring Tool.

Constructor Summary

Description
MenuTool()

Method Summary

Returns	Description
public static JMenuBar	**createMenuBar(authoring.event.PageViewButtonListener pageViewButtonListener, authoring.view.PageView pageView)** Creates the MenuBar for the MainView.
public static JToolBar	**createSliderToolbar(javax.swing.DefaultBoundedRangeModel sizeModel, javax.swing.JLabel sliderReadOut)** Creates the SliderToolbar for the MainView.
public static JLabel	**createStatusBar()** Creates the Status Bar for the MainView.

Returns	Description
public static JPanel	**createToolBarPanel(authoring.view.PageView pageView, authoring.event.PageViewButtonListener pageViewButtonListener, javax.swing.DefaultBoundedRangeModel sizeModel, javax.swing.JLabel sliderReadOut)** Creates the ToolBarPanel for the MainView.

authoring.util.ModelCreator

public *ModelCreator*
extends Object

Helper Class to generate various ShapeModels from the Shapes themselves.

Constructor Summary

Description
ModelCreator()

Method Summary

Returns	Description
public static ShapeModel	**createModel(graphics2D.Shape shape)** Generates the ShapeModel for a given Shape.

authoring.util.PolygonPointsListCellRenderer

public *PolygonPointsListCellRenderer*
extends JLabel
implements ListCellRenderer

This class implements the ListCellRenderer interface and defines how an entry for a Point looks in a List. Used in PolygonEditview.

Constructor Summary

Description
PolygonPointsListCellRenderer()

Method Summary

Returns	Description
public Component	**getListCellRendererComponent(javax.swing.JList list, java.lang.Object value, int index, boolean isSelected, boolean cellHasFocus)**

authoring.util.ViewCreator

public *ViewCreator*
extends Object

Helper Class to generate various Views for given ShapeModels. To keep this very general and extendable, the Name of the View Classes that are generated are kept in a Property File. That Way, the chosen Views can easily be altered by changing the entries in the Porperty File.

Constructor Summary

Description
ViewCreator() Creates a new instance of ViewCreator an loads the properties.

Method Summary

Returns	Description
public ShapeInterface	**createChangingShapeView(authoring.model.ShapeModel shapeModel)** Create ChangingShapeViews for given ShapeModels.
public ShapeInterface	**createShapeEditView(authoring.model.ShapeModel shapeModel)** Create ShapeEditViews for given ShapeModels.

Returns	Description
public ShapeInterface	**createShapeView(authoring.model.ShapeModel shapeModel)** Create ShapeViews for given ShapeModels.

Bibliography

[1] C. Decurtins, M. C. Norrie, and B. Signer. Putting the Gloss on Paper: A Framework for Cross-Media Annotation. *NRHM 2003, New Review in Hypermedia and Multimedia*, 2003.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns.* Addison-Wesley, San Francisco, USA, 1995.

[3] D. M. Geary. *Graphic JAVA 1.2, Mastering the JFC Volume 1 AWT, 3rd Edition.* Sun Microsystems Press, Palo Alto, USA, 1999.

[4] D. M. Geary. *Graphic JAVA 2, Mastering the JFC Volume 2 SWING, 3rd Edition.* Sun Microsystems Press, Palo Alto, USA, 1999.

[5] M. C. Norrie and B. Signer. Information Server for Highly-Connected Cross-Media Publishing. *Information Systems Journal, Special Issue: The 12th International Conference on Advanced Information Systems Engineering (CAiSE'03)*, 2003.

[6] M. C. Norrie and B. Signer. Web-Based Integration of Printed and Digital Information. *Lecture Notes in Computer Science*, 2590:200–219, 2003.

[7] M. C. Norrie, A. Würgler, A. Palinginis, K. Gunten, and M. Grossniklaus. OMS Pro 2.0, Introductory Tutorial. Tutorial, Institute for Information Systems, ETH Zurich, Zurich, Switzerland, 2003.

[8] Paper^{++} Project, IST-2000-26130, Disappearing Computer Initiative, http://www.paperplusplus.net.

[9] S. Schulé. XHTML Plug-in for the iServer Architecture. Master's thesis, Institute for Information Systems, ETH Zurich, Zurich, Switzerland, 2004.

[10] B. Signer, M. Grossniklaus, and M. C. Norrie. Java Framework for Database-Centric Web Engineering. In *Proceedings of WebE'2001, 4th Workshop on Web Engineering (in conjunction with WWW10, 10th International World Wide Web Conference)*, Hong Kong, May 2001.

[11] K. Walrath, M. Campione, A. Huml, and S. Zakhour. *The JFC Swing Tutorial, A Guide to Constructing GUIs*. Addison-Wesley, San Francisco, USA, 1999.