Aletheia Web Layout Editor (AWL Editor / AletheiaWeb) - Integration

Christian Clausner, Christos Papadopoulos, Stefan Pletschacher

24 October 2014

PRImA Research Lab
University of Salford
UK

Contents

Introduction

This document explains how to integrate the AWE Layout Editor (formerly AletheiaWeb or WebLayoutEditor) into an existing document repository or similar framework.

Remark on the new name:

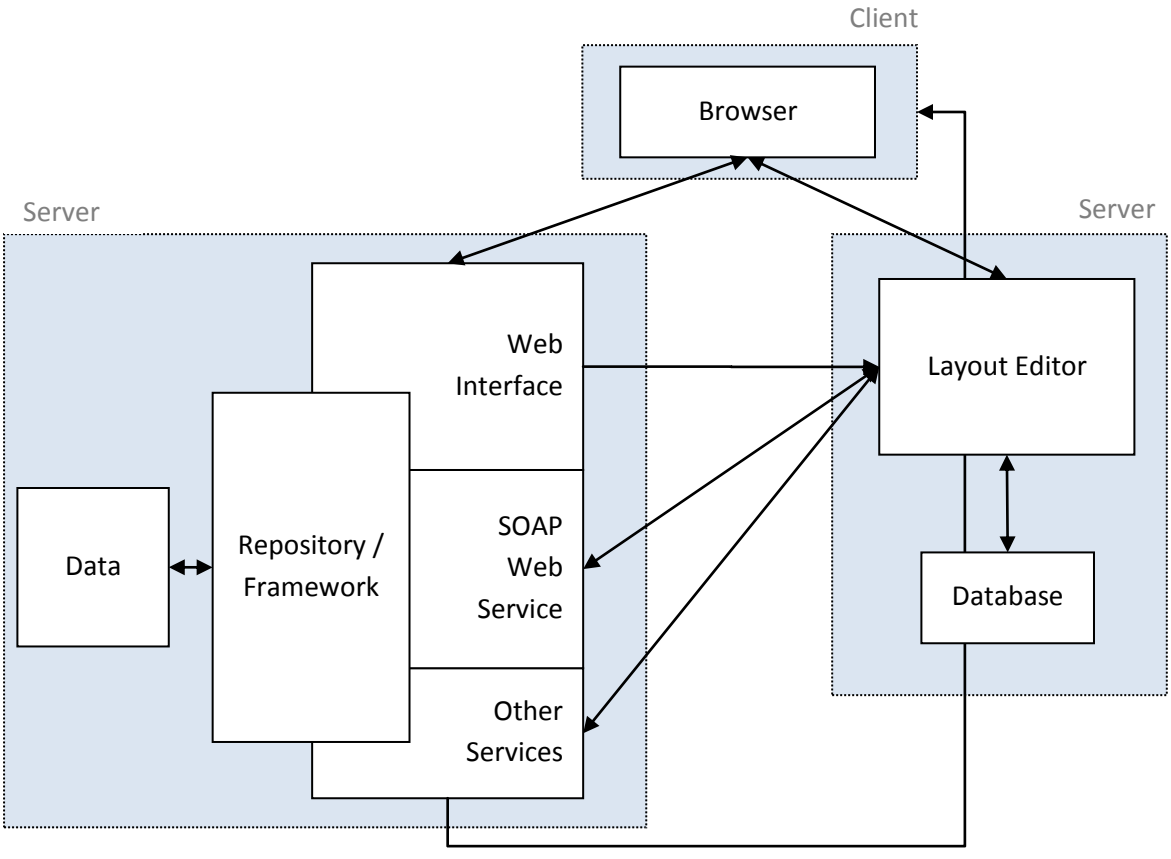
We decided to create a new branch of AletheiaWeb that is entirely dedicated to the eMOP project. The new project will share some common modules but will have considerably different look and functionality compared to AletheiaWeb.

Working title: 'Web Layout Editor'

Full name: Aletheia Web Edition Layout Editor (or short AWE Layout Editor)

Overview

Client/server architecture:



The general idea is that the Layout Editor is integrated with the document repository in a loosely coupled way. The editor has minimal knowledge of its surroundings. Initial parameters are passed via the URL. All further communication is done via web services.

Technologies

Google Web Toolkit (GWT)

The Layout Editor is implemented using GWT which has following features:

- Coding in Java and HTML/CSS
- Compilation to:
    - HTML/JavaScript on client side (native support for most browsers)
    - Java servlets on server side (e.g. on Apache Tomcat)

Simple Object Access Protocol (SOAP)

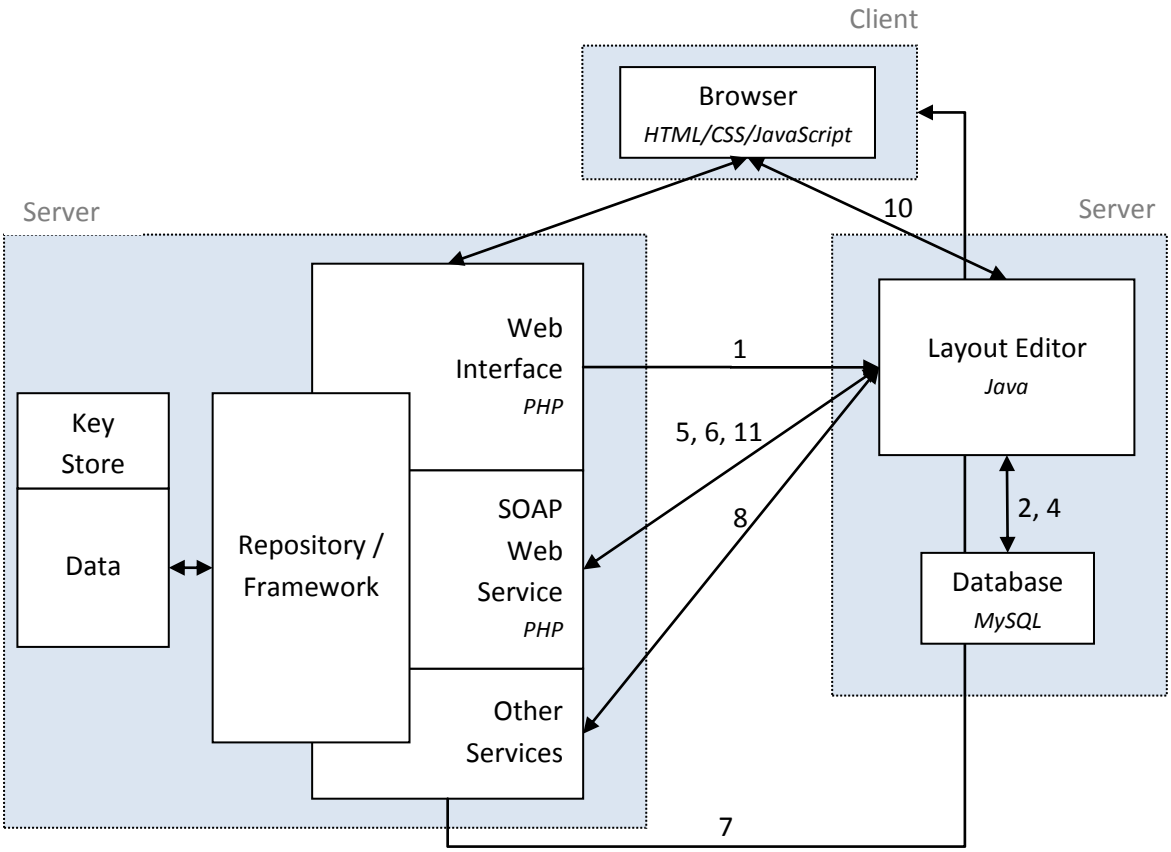
The communication between the repository and the Layout Editor is mainly done via SOAP:

- XML based protocol
- SOAP server on repository side (receives requests, sends responses)
- SOAP client on Layout Editor side (sends requests, receives responses)

PHP

The example repository that is shown here has been implemented with PHP. However, any equivalent technology can be used (basic requirement is the support of AES encryption).

Detailed Architecture and Data Flow



- Initialisation
  1. The repository website embeds/calls the Web Layout Editor (optional via secured connection) and passes:
     a. Document ID (e.g. image filename)
     b. Attachment ID (the layout is an attachment for the document)
     c. Application ID (e.g. emop-dataset)
     d. Encrypted user token (user ID, IP address, time stamp)
  2. The Layout Editor gets the decryption key from the database using the application ID (the key is unique for an application[1])
  3. The Layout Editor validates the user token using the private key
  4. The Layout Editor gets the URL of the SOAP web service from the database (the service URL can be unique for an application)
  5. The Layout Editor calls the 'getDocumentAttachmentSources' method of the SOAP web service (passing user ID and attachment ID) and gets back:
     a. URL to get document image (ImageSource)
     b. URL to get attachment (AttachmentSource)
  6. The Layout Editor requests permissions ('getDocumentAttachmentPermissions' SOAP method)

[1] The proposed architecture allows multiple applications to use the same instance of the layout editor.

7. The Layout Editor requests the document image
8. The Layout Editor requests the attachment for the page layout (PAGE XML)

- Saving changes
  9. The user clicks on “Save”
  10. The client calls the remote service for saving the page layout.
  11. The Layout Editor sends a SOAP request (including the PAGE XML file)
  12. The repository handles the save (create new data entry or replace the old data entry)
  13. The repository sends a confirmation (including new attachment ID if necessary)

SOAP Requests and Responses

Requests to the SOAP web service consist of a single method with optional arguments. The response is in form of plain text, but can contain any structured data (e.g. XML).

Example:

Method: getDocumentAttachmentSources
Arguments: uid, aid

SOAP Request:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header/>
        <SOAP-ENV:Body>
                <p:getDocumentAttachmentSources xmlns:p="www.primaresearch.org">
                        <uid> user1 </uid>
                        <aid> attachment1 </aid>
                </p:getDocumentAttachmentSources>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:ns1="www.primaresearch.org"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
                ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
                ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
                <ns1:getDocumentAttachmentSourcesResponse>
                        <return xsi:type="xsd:string">
                                ...(XML data)
                        </return>
                </ns1:getDocumentAttachmentSourcesResponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Authentication

The SOAP web service is secured via .htaccess and requires authentication (BASIC).

SOAP Methods

Method	Arguments	Description	Returns
getDocumentAttachmentSources	uid	User ID	XML with sources (see next section)
	aid	Attachment ID	
getDocumentAttachmentPermissions	uid	User ID	XML with permissions (see next section)
	aid	Attachment ID	
saveDocumentAttachment	uid	User ID	XML with return code (0 for success) and new attachment ID + new attachment source or error message (see next section)
	aid	Attachment ID	
	mode	Save mode (new, replace)	
	attachment	Attachment data (e.g. XML)	

XML Format of SOAP Responses (Steps 5 and 6)

```
<DocumentRepository>
        <SaveDocumentAttachmentResult attachmentId="219877" newAttachmentId="34235" returnCode="x">
                <ReturnMessage>...</ReturnMessage>
        </SaveDocumentAttachmentResult>

        <DocumentAttachmentSources>
                <ImageSource documentId="20142" type="fullview">
                        http://www..../.../getImage.php?Did=20142&type=fullview
                </ImageSource>
                <AttachmentSource attachmentId="219877" attachmentTypeId="PAGE">
                        http://www..../.../getAttachment.php?Aid=219877
                </AttachmentSource>
        </DocumentAttachmentSources>

        <DocumentAttachmentPermissions attachmentId="219877" >
                <Permission name="…"/>
        </DocumentAttachmentPermissions>
</DocumentRepository>
```

Note: Only the portions relevant to the called SOAP method will be included in the response.

Secure Token

To prevent unauthorised access / session hijacking, sensible data is passed via an encrypted token to the Layout Editor. The token contains the user name, the client IP address and a timestamp; all encoded in JSON format and encrypted using AES (128bit). For the Layout Editor prototype the token times out after one minute. This can be an adjustable setting in the final version.

Example token (not encrypted):

```
{"ip":"146.6.65.141","ts":1362146903,"uid":"user1"}
```

(Note: The timestamp represents the *seconds* since 01/01/1970)

Encryption and decryption

A standard AES algorithm with key length 128 bit is used to encrypt/decrypt the JSON data. Private keys which have to be exchanged when setting up the system are stored on both sides.

PHP implementation of the encryption:

```
function createAuthenticationToken($username, $secretKey)
{
        $orig_json_array = json_encode(array('ip'=>$_SERVER['REMOTE_ADDR'], 'ts'=>time(),
'uid'=>$username));
        $temp = $orig_json_array;
        $secretKey = md5($secretKey);
        $initialVector = mcrypt_create_iv(mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128,
MCRYPT_MODE_CFB) ,MCRYPT_DEV_RANDOM);
        $temp = mcrypt_encrypt(MCRYPT_RIJNDAEL_128, $secretKey, $temp, MCRYPT_MODE_CFB,
$initialVector);
        $temp = $initialVector.$temp;
        $temp = base64_encode($temp);
        return urlencode($temp);
}


$authenticationToken = createAuthenticationToken($PARAMS['username'], $PARAMS['secretKey']);

$urlBase = "http://...:8080/WebLayoutEditor?";
$targeturl = $urlBase . "Did=$Did&Aid=$Aid&Appid=".$PARAMS['Appid']."&a=$authenticationToken";
```

Java implementation of the decryption:

```
...
String msgBase64 = Window.Location.getParameter("a"); //This is already URL decoded
...

private String decrypt(String msgBase64, String key) {

        final String PHP_CHAR_ENCODING = "ISO-8859-1";
        final int IV_LENGTH = 16;
        Base64 base64 = new org.apache.commons.codec.binary.Base64();

        String decryptedData = null;
        try {
                byte[] msgBytes = base64.decode(msgBase64.getBytes());
                String m = new String(msgBytes, PHP_CHAR_ENCODING);

                //Split into initialisation vector and encrypted data
                String initialVectorString = m.substring(0, IV_LENGTH);
                String encryptedData = m.substring(IV_LENGTH);

                byte[] initialVectorBytes = initialVectorString.getBytes();
                byte[] encryptedDataBytes = encryptedData.getBytes(PHP_CHAR_ENCODING);

                //Decrypt
                String md5key = md5(key);
                SecretKeySpec skeySpec = new SecretKeySpec(md5key.getBytes(), "AES");
                IvParameterSpec initialVector = new IvParameterSpec(initialVectorBytes);
                Cipher cipher = Cipher.getInstance("AES/CFB8/NoPadding");
                cipher.init(Cipher.DECRYPT_MODE, skeySpec, initialVector);
```

```
                decryptedData = new String(decryptedByteArray, PHP_CHAR_ENCODING);
        } catch (Exception e) {
            //Error handling
        }
        return decryptedData;
    }

    private static String md5(String input) throws NoSuchAlgorithmException {
            MessageDigest md = MessageDigest.getInstance("MD5");
            byte[] messageDigest = md.digest(input.getBytes());
            BigInteger number = new BigInteger(1, messageDigest);
            return String.format("%032x", number);
    }
```

Character Encoding

The setting for the character encoding might have to be adjusted if PHP is not using "`ISO-8859-1`". The encoding is specified in the web.xml file of the Layout Editor:

```
<init-param>
  <description>Character encoding used for encryption/decryption</description>
  <param-name>ENCRYPTION_CHAR_ENCODING</param-name>
  <param-value>ISO-8859-1</param-value>
</init-param>
```

Required Libraries / Formats

We use "gson" for JSON decoding. It's an open library by Google:
https://sites.google.com/site/gson

Required library for Base64 encoding/decoding: Commons Codec
http://commons.apache.org/proper/commons-codec/

Java (on server side) might need an update of the cryptography library:

> *"java.security.InvalidKeyException: Illegal key size" error is a common issue which occurs when you try to invoke a secured web service in an environment where the provision for java unlimited security jurisdiction is not done.*
> *This can be avoided by installing Java Cryptography Extension (JCE) unlimited strength jurisdiction policy files.*
>
> *1. Go to http://www.oracle.com/technetwork/java/javase/downloads/index.html*
>
> *2. Go to the Additional Resources section and click on the download button next to* **"Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files X"**
>
> *3. Download* **jce_policy-X.zip** *and extract it into a directory*
>
> *4. You will find* **local_policy.jar** *and* **US_export_policy.jar** *files there in the extracted directory. Copy these two files to $JAVA_HOME/jre/lib/security directory (These files might already be there, replace them in this case)*
>
> *5. Restart the web server and invoke your secured service again. You will not encounter the "invalidkeyException" any more*

Securing the Web Application in Apache Tomcat

It is possible to secure a web application using Apache Tomcat's authentication mechanism. The users need to be set up in a configuration file. The caller can use 'BASIC' authentication to gain access to the web app.

See also:
http://www.avajava.com/tutorials/lessons/how-do-i-use-basic-authentication-with-tomcat.html

Example and Prototype

This integration example comes with a simple PHP implementation of a repository (including SOAP service files) and a prototype of the Web Layout Editor.

PHP Example Repository

Requirements:

- PHP 5.3, with the following extensions enabled
  - JSON 1.2.1 (enabled by default, might need enabling/installation in older versions of PHP)
  - mCrypt 2.5.8
  - SOAP (supporting v 1.2 calls)
  - XML writer (enabled by default)

Deployment:

- Once PHP is set up, all files for the PHP site have to be copied into a folder under the web root.

Configuration:

- All configuration parameters for the PHP prototype can be found in the config.inc file.
- Appropriate permissions have to be given for the web user to write in the targt folder for storing the PAGE files.

Web Layout Editor Web App

Requirements

- Web application server with servlet capability (servlet container), e.g. Apache Tomcat: http://tomcat.apache.org/
- Java 1.7 or higher
- Possibly an update of the Java encryption libraries (see above)

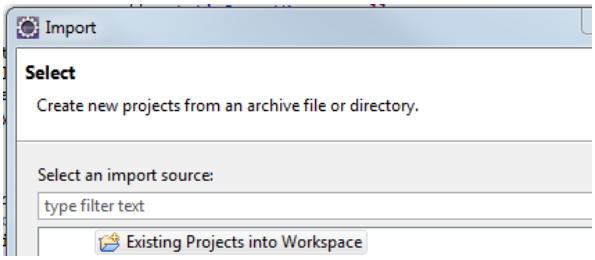
Project Setup

Requirements:

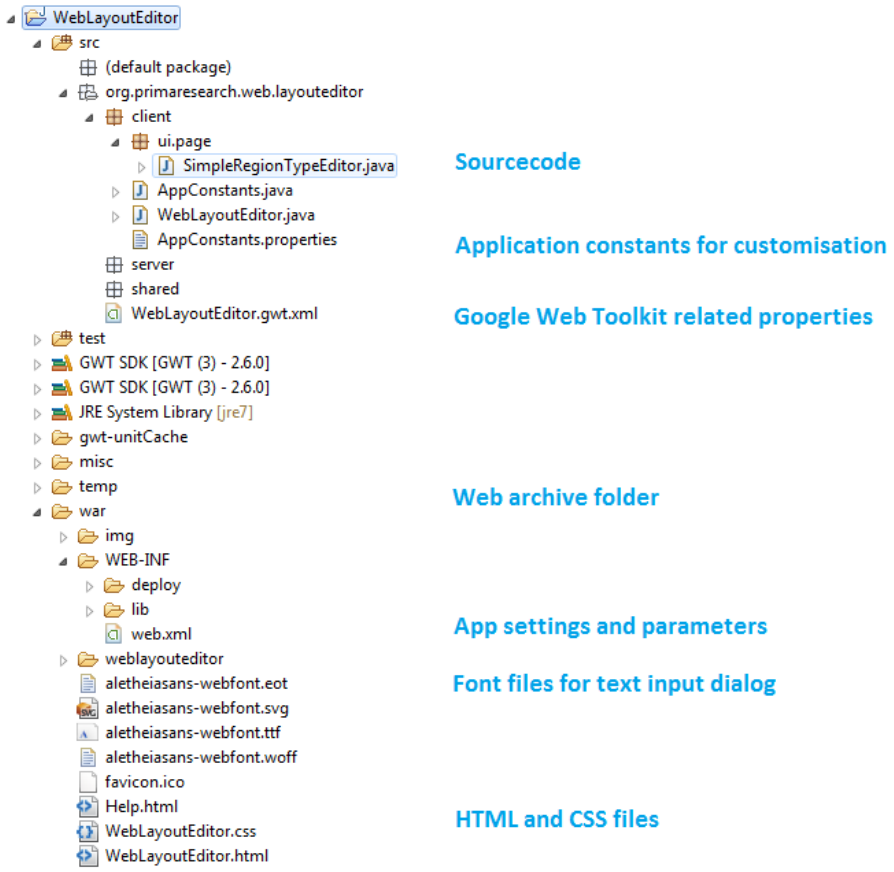
- Eclipse 3.7 or higher
- Google Web Toolkit (GWT) plugin for eclipse (version 2.6)
- Java 1.7

Getting started:

- Extract the zipped projects into the eclipse workspace (PrimaBasic, PrimaMaths, PrimaIo, PrimaDla, PrimaMaths, PrimaGwt, and WebLayoutEditor)
- Import the projects

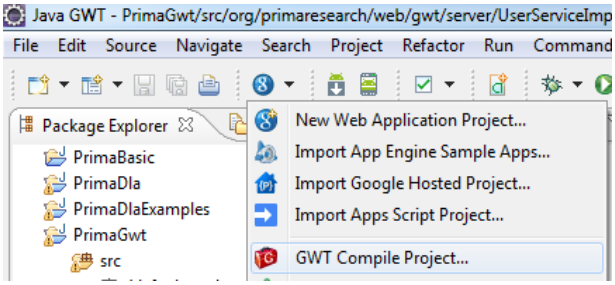


WebLayoutEditor project contents:

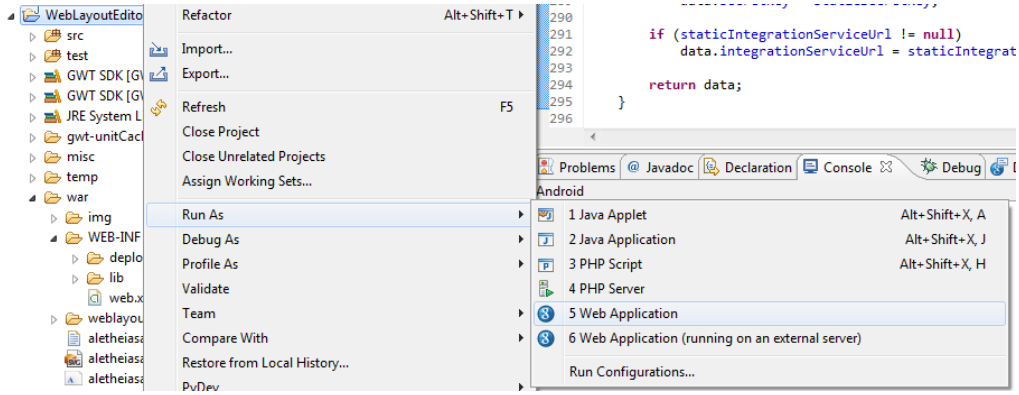


Compiling and running:

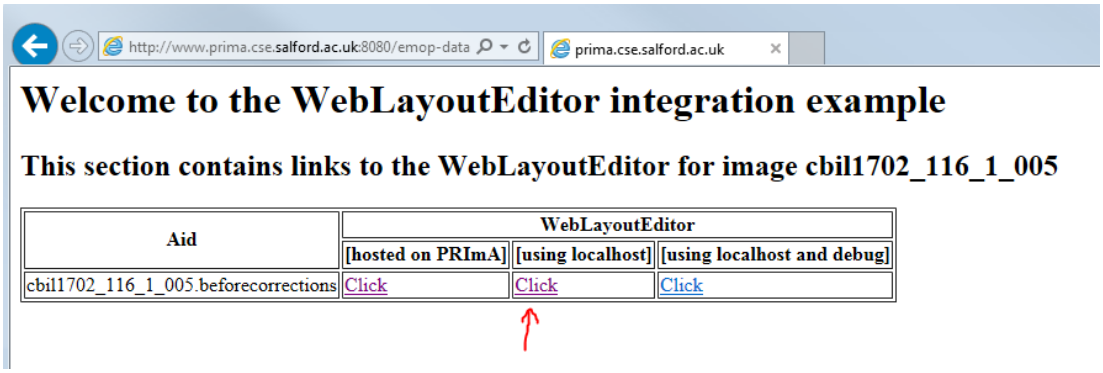
- Compile the WebLayoutEditor project using the GWT menu (the project needs to be selected in the package explorer)



- Run the project using the “Web Application” option (starts the integrated web app server)



- Open the AWL editor in your browser using the demo framework (PHP)

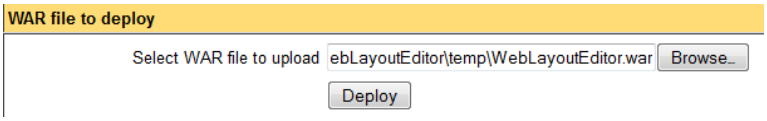


Creating a web archive (WAR) for deployment on the server

- Create Jar files for the Prima libraries, if changed (otherwise use the provided Jar files)
  - The jar files can be created using the "CreateJar.jardesc" files in the individual projects. When using for the first time, adjust the export folder of the "CreateJar.jardesc" (double click opens the editor). Otherwise right click the "CreateJar.jardesc" in the package explorer and select "Create Jar".
- Copy all required Java libraries (.jar files) into the folder "[project]\war\web-inf\lib" (if they are missing); At this point of time these libraries are:
  - commons-codec-1.8.jar
  - gson-2.2.2.jar
  - gwt-servlet.jar
  - mysql-connector-java-5.1.23-bin.jar
  - PrimaBasic.jar
  - PrimaMaths.jar
  - PrimaIo.jar
  - PrimaDla.jar
  - PrimaGwt.jar
- Put the contents of the folder "[project]\war\" in a Zip file (do not zip the whole "war" folder, only the contents)
- Rename the .zip file to a .war file

Deployment

- Simply deploy the web archive file 'WebLayoutEditor.war' to the application server



Customisation

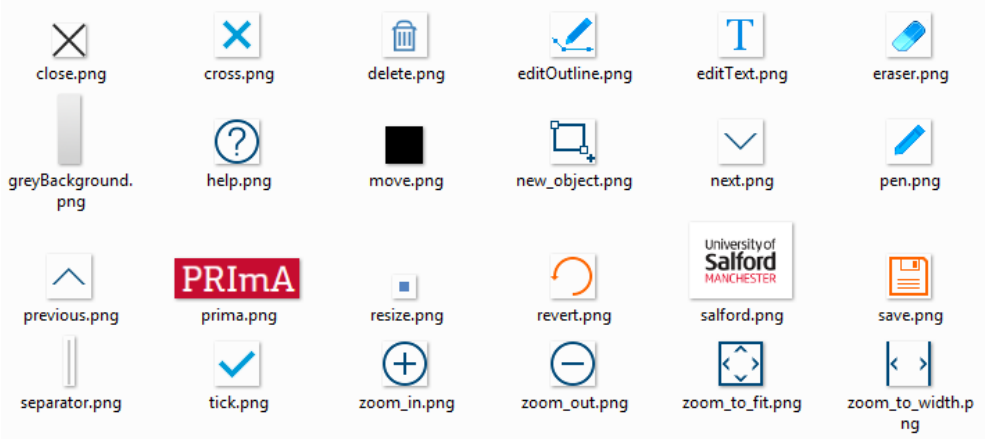
HTML Files

The web layout editor creates almost all HTML content dynamically within an 'object' element. The main HTML file is "WebLayoutEditor.html" which only contains a placeholder in case the web app

cannot be loaded. Furthermore there is “help.html” which represents the content of the help dialogue that can be opened by the user (question mark icon in toolbar).

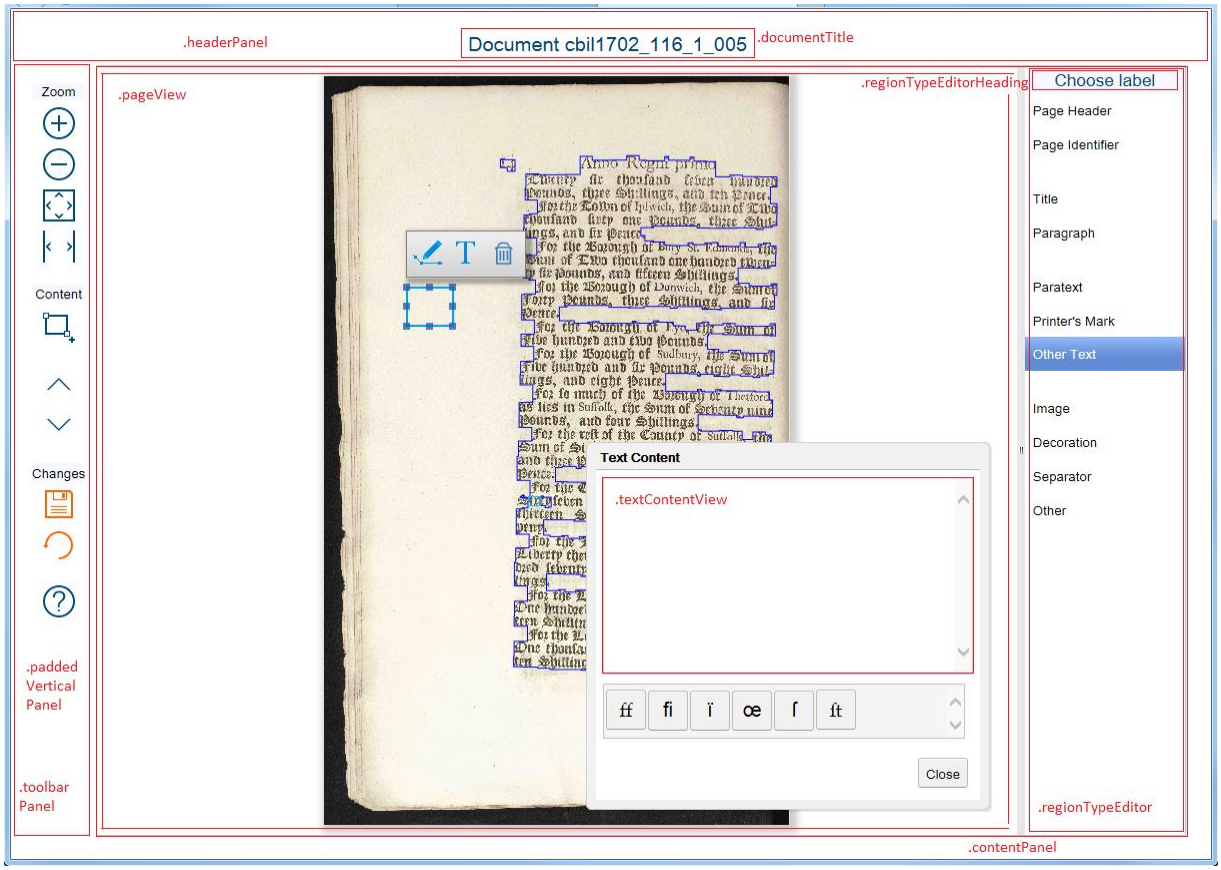
Images and Icons

All images and icons used in the web app are located in: [war]\img\



CSS

These are the main CSS classes used for the editor:



Constants and Strings

Several application constants and strings have been externalised and can be changed without recompiling the web app. The list of constants/strings is in following file:

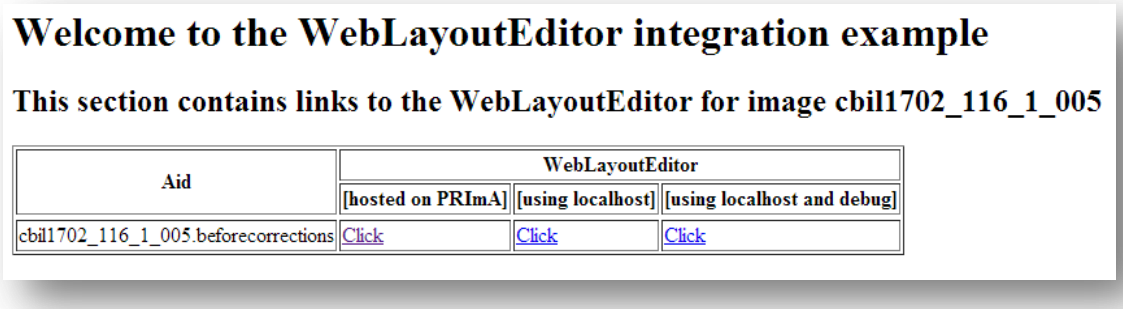
```
[war]\WEB-INF\classes\org\primaresearch\web\layouteditor\client\AppConstants.properties
```

This includes several settings for the user interface that cannot be changed via CSS.

Error Handling

As long as the client-to-server communication works all errors and warnings should be logged on server side (accessible through the log files of the application server, e.g. Apache Tomcat). Critical errors on client side are also presented to the user as popup window.

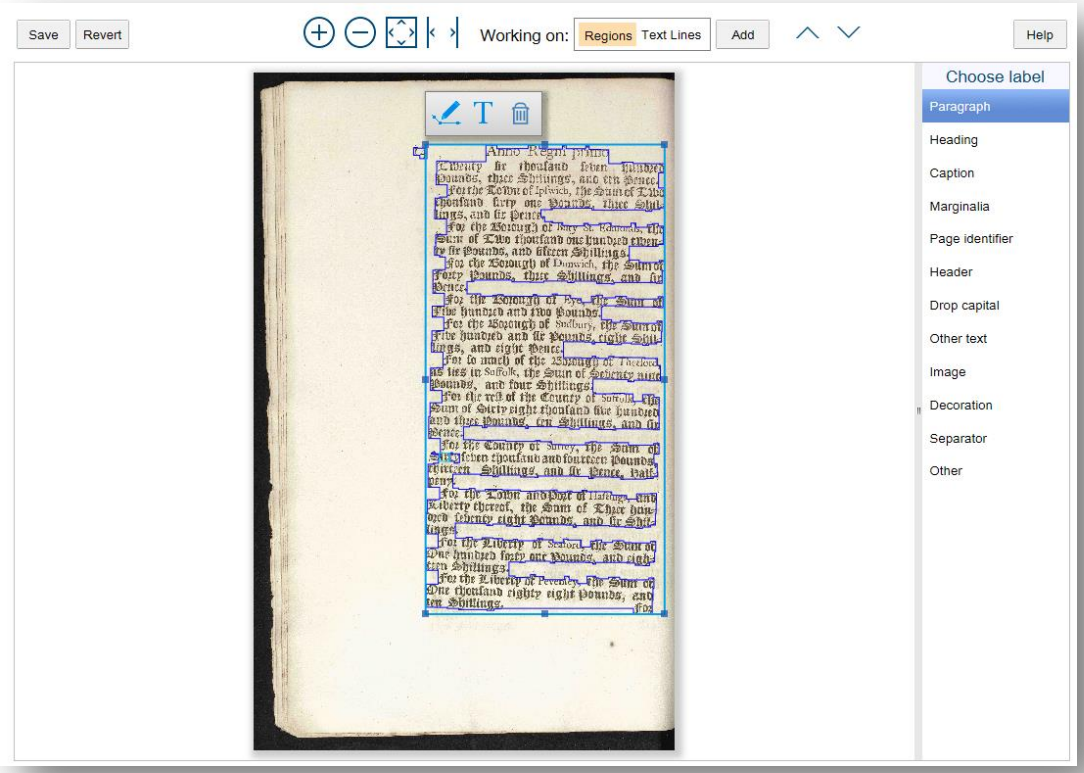
Example



Welcome to the WebLayoutEditor integration example

This section contains links to the WebLayoutEditor for image cbil1702_116_1_005

Aid	WebLayoutEditor		
	[hosted on PRImA]	[using localhost]	[using localhost and debug]
cbil1702_116_1_005.beforecorrections	Click	Click	Click



Implemented features (prototype):

- Pan + zoom
- Show regions (or text lines – disabled for current version)
- Select layout objects
- Edit polygonal outlines of layout objects
- Change type (label, tag) of region
- Show / modify text content
- Create layout objects
- Delete layout objects
- Save changes
- Discard changes