Creating OWASP ZAP Extensions and Add-ons

Version 1.0

Abstract

A Step by Step guideline on how to create OWASP ZAP Extensions and Add-Ons for version 2.1.0

Johanna Curiel

Johanna.curiel@owasp.org

Table of Contents

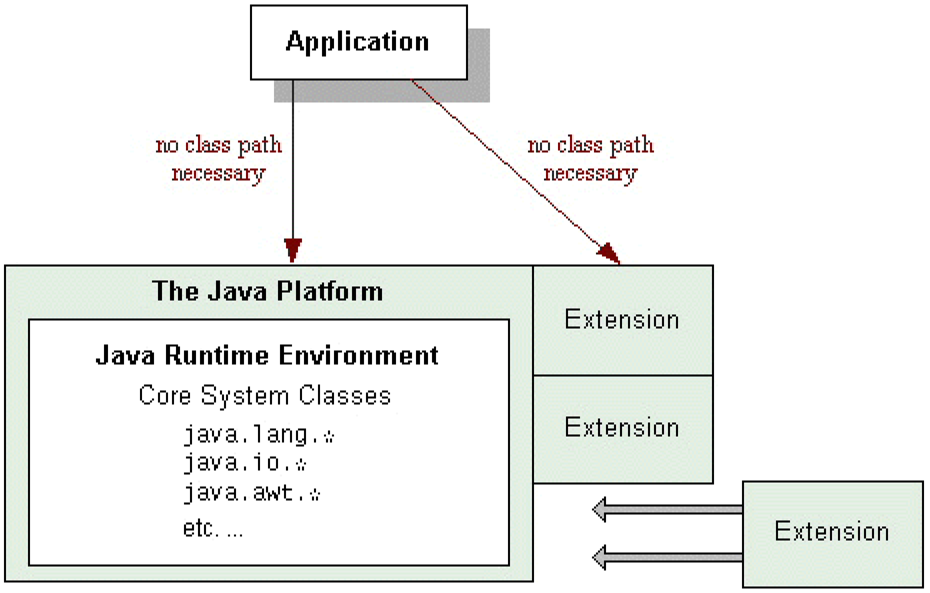
Introduction

The Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing. ZAP provides automated scanners as well as a set of tools that allow you to find security vulnerabilities manually.

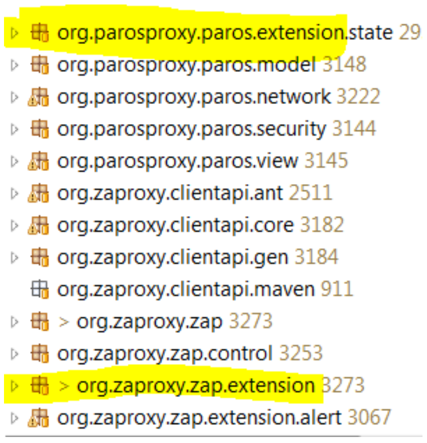
This is a quick, step by step guideline about how to create extensions and Add-Ons in OWASP ZAP.

What are zap extensions?

Zap extensions are java packages that extend the existing functionality within OWASP ZAP. This concept could be called the "Extension Mechanism" which provides a standard way to create custom features or API's to Java applications



OWASP ZAP contains major functionalities in the Parosproxy.paros packages. Paros, which is an open source Java program with multiple functionalities, it's not been updated since 2006. OWASP ZAP uses part of these functionalities and has built many new. Therefore when building a new custom extension for OWASP ZAP, you need to create a package within org.zaproxy.zap extension.


org.parosproxy.paros.extension.state 29
org.parosproxy.paros.model 3148
org.parosproxy.paros.network 3222
org.parosproxy.paros.security 3144
org.parosproxy.paros.view 3145
org.zaproxy.clientapi.ant 2511
org.zaproxy.clientapi.core 3182
org.zaproxy.clientapi.gen 3184
org.zaproxy.clientapi.maven 911
> org.zaproxy.zap 3273
org.zaproxy.zap.control 3253
> org.zaproxy.zap.extension 3273
org.zaproxy.zap.extension.alert 3067

The code that you produce as a new extension needs to go through different phases. You could consider your new package as a “Proposal” that will be feature in the “Marketplace”. This concept is proposed by Simon Bennetts (aka PSIION), the Project Leader of OWASP ZAP, to get new contributors onboard but also allow users at an early stage development of the new extension to test it and provide feedback to the new feature.

By creating features as new extensions, it makes the program manageable and modular.

Code conventions for developing new extensions

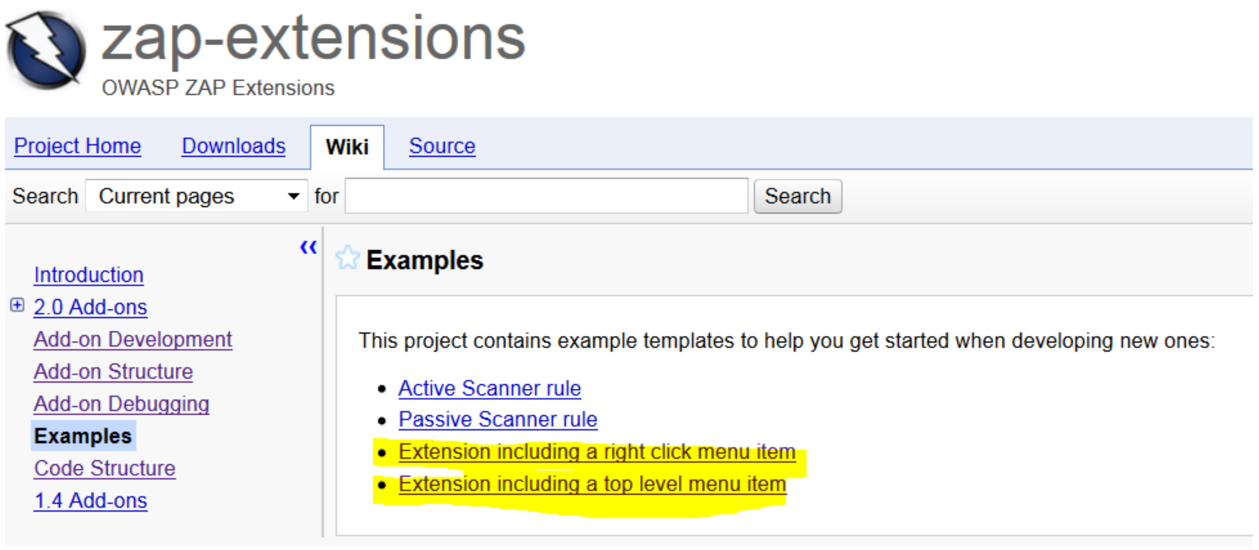
OWASP ZAP has a set of development rules and guidelines that must be followed. Please go to https://code.google.com/p/zaproxy/wiki/DevGuidelines and review these guidelines. In addition, I will highlight some regarding the creation of new packages:

- All significant new code should be under the org.zaproxy.zap package.
- All functionality should be fully documented in the help pages. However the documentation can be added after the code has been checked in as long as this is done before the next release. Note that the wiki help pages must not be changed manually - these reflect the last release, and are generated from the help pages using a script.
- Please create Issues for all significant changes and post messages to the development group so that everyone knows what’s going on

My first Extension

The easiest way to understand how to build a new extension is by following a simple example. In the wiki https://code.google.com/p/zap-extensions/wiki/Examples you can find some basic info, however this example explains step by step how to create this example templates into a new extension

In the wiki you can find 4 different examples. The first 2 are scanner rules and the last ones explain how to extent menu’s with new features.



This project contains example templates to help you get started when developing new ones:

- Active Scanner rule
- Passive Scanner rule
- ==Extension including a right click menu item==
- ==Extension including a top level menu item==

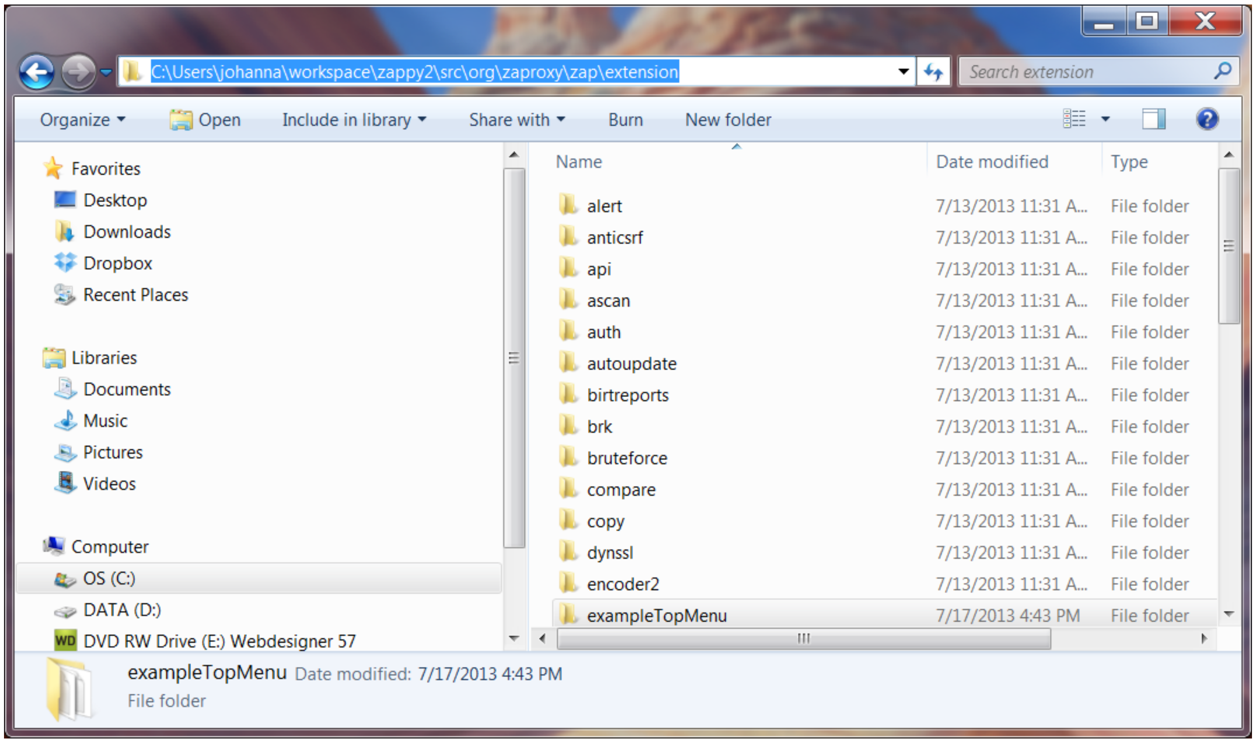
From this, we will create the TOP-LEVEL menu item

Step 1: Download source code and Build ZAP

There is a very detailed guideline on how to get the latest OWASP ZAP source code and configure properly your Eclipse environment. Raul Siles of Taddong is updating these guidelines which explain in very detail how to Build ZAP. Keep in mind that this info can change depending on the version of ZAP. You can find the latest guides here http://code.google.com/p/zaproxy/wiki/Building

Step 2: Create new extension

Create a new folder exampleTopMenu, under the project OWASP ZAP in your workspace location. Located the folder "..\src\org\zaproxy\zap\extension" and place 2 files into the folder. In the wiki you will find the code of the extension class and the Message.properties file.

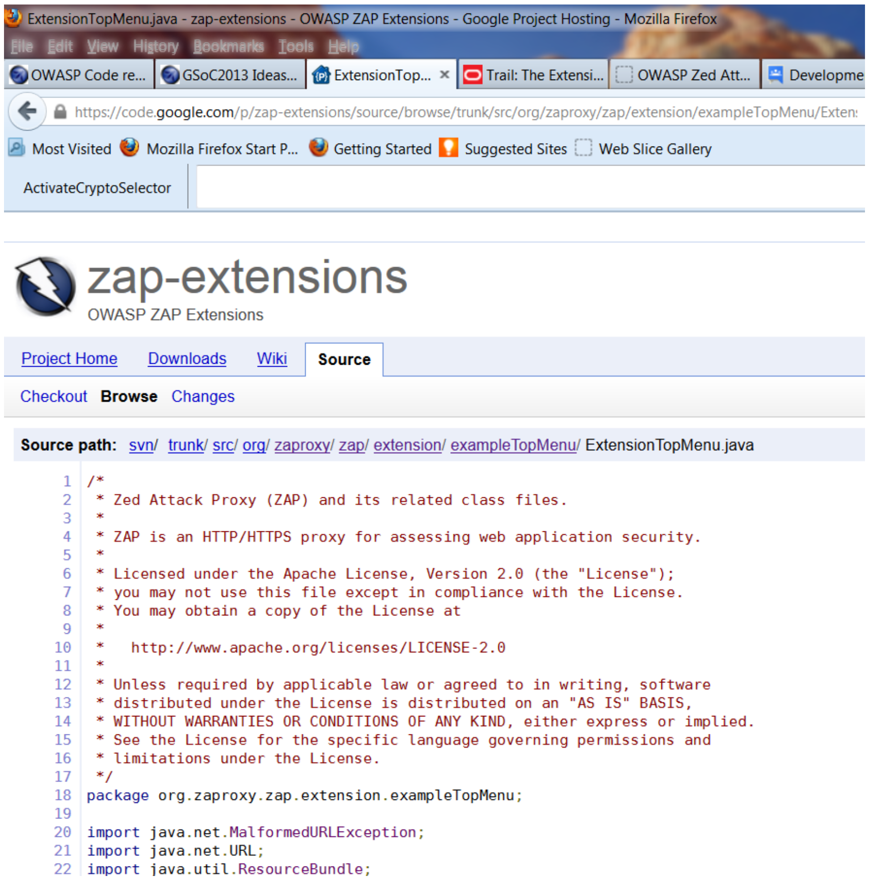


C:\Users\johanna\workspace\zappy2\src\org\zaproxy\zap\extension

Name	Date modified	Type
alert	7/13/2013 11:31 A...	File folder
anticsrf	7/13/2013 11:31 A...	File folder
api	7/13/2013 11:31 A...	File folder
ascan	7/13/2013 11:31 A...	File folder
auth	7/13/2013 11:31 A...	File folder
autoupdate	7/13/2013 11:31 A...	File folder
birtreports	7/13/2013 11:31 A...	File folder
brk	7/13/2013 11:31 A...	File folder
bruteforce	7/13/2013 11:31 A...	File folder
compare	7/13/2013 11:31 A...	File folder
copy	7/13/2013 11:31 A...	File folder
dynssl	7/13/2013 11:31 A...	File folder
encoder2	7/13/2013 11:31 A...	File folder
exampleTopMenu	7/17/2013 4:43 PM	File folder

exampleTopMenu Date modified: 7/17/2013 4:43 PM
File folder

https://code.google.com/p/zap-extensions/source/browse/trunk/src/org/zaproxy/zap/extension/exampleTopMenu/ExtensionTopMenu



ExtensionTopMenu.java - zap-extensions - OWASP ZAP Extensions - Google Project Hosting - Mozilla Firefox

https://code.google.com/p/zap-extensions/source/browse/trunk/src/org/zaproxy/zap/extension/exampleTopMenu/Exten

zap-extensions
OWASP ZAP Extensions

Source path: svn/ trunk/ src/ org/ zaproxy/ zap/ extension/ exampleTopMenu/ ExtensionTopMenu.java

```
/*
 * Zed Attack Proxy (ZAP) and its related class files.
 *
 * ZAP is an HTTP/HTTPS proxy for assessing web application security.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.zaproxy.zap.extension.exampleTopMenu;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.ResourceBundle;
```

You can also download the example folder. Go to the following location:

https://code.google.com/p/zap-extensions/wiki/Examples and click on Extension including a right click menu item

In the Example package create a class called ExtensionTopMenu.java

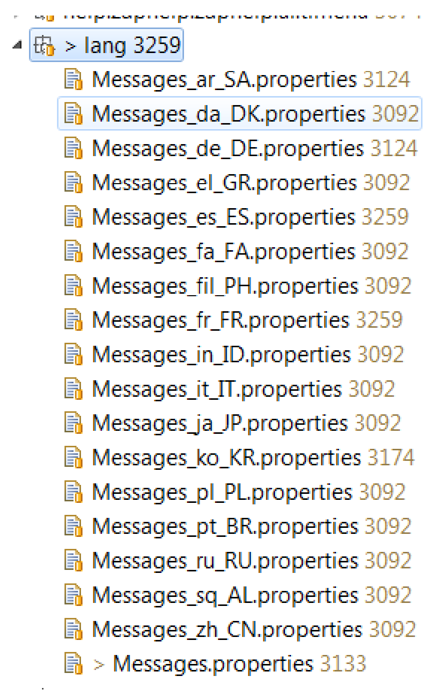
```
/*
 * Zed Attack Proxy (ZAP) and its related class files.
 *
 * ZAP is an HTTP/HTTPS proxy for assessing web application security.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.zaproxy.zap.extension.exampleTopMenu;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.ResourceBundle;

import javax.swing.JMenuItem;

import org.parosproxy.paros.Constant;
import org.parosproxy.paros.extension.ExtensionAdaptor;
import org.parosproxy.paros.extension.ExtensionHook;
import org.parosproxy.paros.view.View;

/*
 * An example ZAP extension which adds a top level menu item.
 *
 * This class is defines the extension.
 */
public class ExtensionTopMenu extends ExtensionAdaptor {

    private JMenuItem menuExample = null;
    private ResourceBundle messages = null;

        /**
     *
     */
    public ExtensionTopMenu() {
        super();
                initialize();
    }

    /**
     * @param name
     */
    public ExtensionTopMenu(String name) {
        super(name);
    }

        /**
         * This method initializes this
         *
         */
        private void initialize() {
        this.setName("ExtensionTopMenu");
        // Load extension specific language files - these are held in the
extension jar
        messages = ResourceBundle.getBundle(
                        this.getClass().getPackage().getName() +
".Messages", Constant.getLocale());
        }

        @Override
        public void hook(ExtensionHook extensionHook) {
            super.hook(extensionHook);

            if (getView() != null) {
                // Register our top menu item, as long as we're not running
as a daemon
                // Use one of the other methods to add to a different menu
list
                extensionHook.getHookMenu().addToolsMenuItem(getMenuExample(
));
            }

        }

        private JMenuItem getMenuExample() {
        if (menuExample == null) {
                menuExample = new JMenuItem();
                menuExample.setText(getMessageString("ext.topmenu.tools.exam
ple"));

                menuExample.addActionListener(new
java.awt.event.ActionListener() {
                @Override
                public void actionPerformed(java.awt.event.ActionEvent e) {
                        // This is where you do what you want to do.
                        // In this case we'll just show a popup message.
                        View.getSingleton().showMessageDialog(getMessageStri
ng("ext.topmenu.msg.example"));
                }
            });
        }
        return menuExample;
    }

        public String getMessageString (String key) {
                return messages.getString(key);
        }
        @Override
        public String getAuthor() {
                return Constant.ZAP_TEAM;
        }

        @Override
        public String getDescription() {
                return messages.getString("ext.topmenu.desc");
        }

        @Override
        public URL getURL() {
                try {
                        return new URL(Constant.ZAP_EXTENSIONS_PAGE);
                } catch (MalformedURLException e) {
                        return null;
                }
        }
}
```

Step 3 : Define Message.properties file(s)

It is important to keep in mind that every extension you build with its own display text (or text that can be translated) must contain its own Message.properties file

```
# An example ZAP extension which adds a top level menu item.
#
# This file defines the default (English) variants of all of the
internationalised messages

ext.topmenu.desc=Example extension which provides a top level menu
ext.topmenu.tools.example=topmenu: an example menu
ext.topmenu.msg.example=topmenu: An example message
```

This text is used in the menu application. In fact, once you get acquaintance with the OWASP ZAP code, you will find this file and multiple translation text files under ‘lang’ package, still, remember that your extension must have its own Message.properties file.


```
> lang 3259
    Messages_ar_SA.properties 3124
    Messages_da_DK.properties 3092
    Messages_de_DE.properties 3124
    Messages_el_GR.properties 3092
    Messages_es_ES.properties 3259
    Messages_fa_FA.properties 3092
    Messages_fil_PH.properties 3092
    Messages_fr_FR.properties 3259
    Messages_in_ID.properties 3092
    Messages_it_IT.properties 3092
    Messages_ja_JP.properties 3092
    Messages_ko_KR.properties 3174
    Messages_pl_PL.properties 3092
    Messages_pt_BR.properties 3092
    Messages_ru_RU.properties 3092
    Messages_sq_AL.properties 3092
    Messages_zh_CN.properties 3092
    > Messages.properties 3133
```

Message.properties text translation

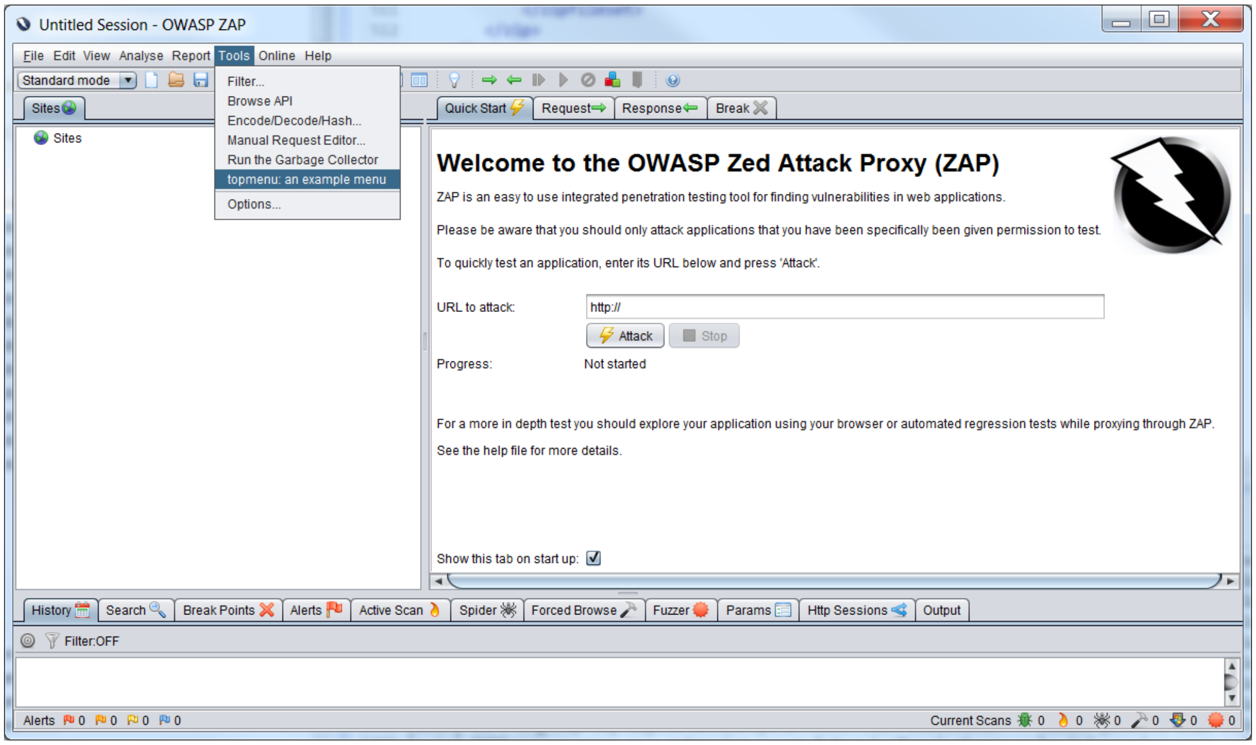
If you wish to have your text translated into multiple languages, then you need to add additional properties files such as the example above. For the proper translation of your extension text, you must submit your translation files to http://crowdin.net/project/owasp-zap/activity_stream

Step 4: Refresh and Run

Go back to eclipse environment, Refresh the project by right-click the "src" folder under the project and select 'Refresh' from the drop down menu.

Right click the 'src' again and select Run

When the ZAP application launches, right click in the left pane or the bottom pane of the ZAP window. You will see a new menu item as follows:


Untitled Session - OWASP ZAP
File Edit View Analyse Report Tools Online Help
Standard mode
Filter...
Browse API
Encode/Decode/Hash...
Manual Request Editor...
Run the Garbage Collector
topmenu: an example menu
Options...
Sites
Quick Start
Request
Response
Break
History Search Break Points Alerts Active Scan Spider Forced Browse Fuzzer Params Http Sessions Output
Filter:OFF
Alerts 0 0 0 0
Current Scans 0 0 0 0 0 0

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://

Attack Stop

Progress: Not started

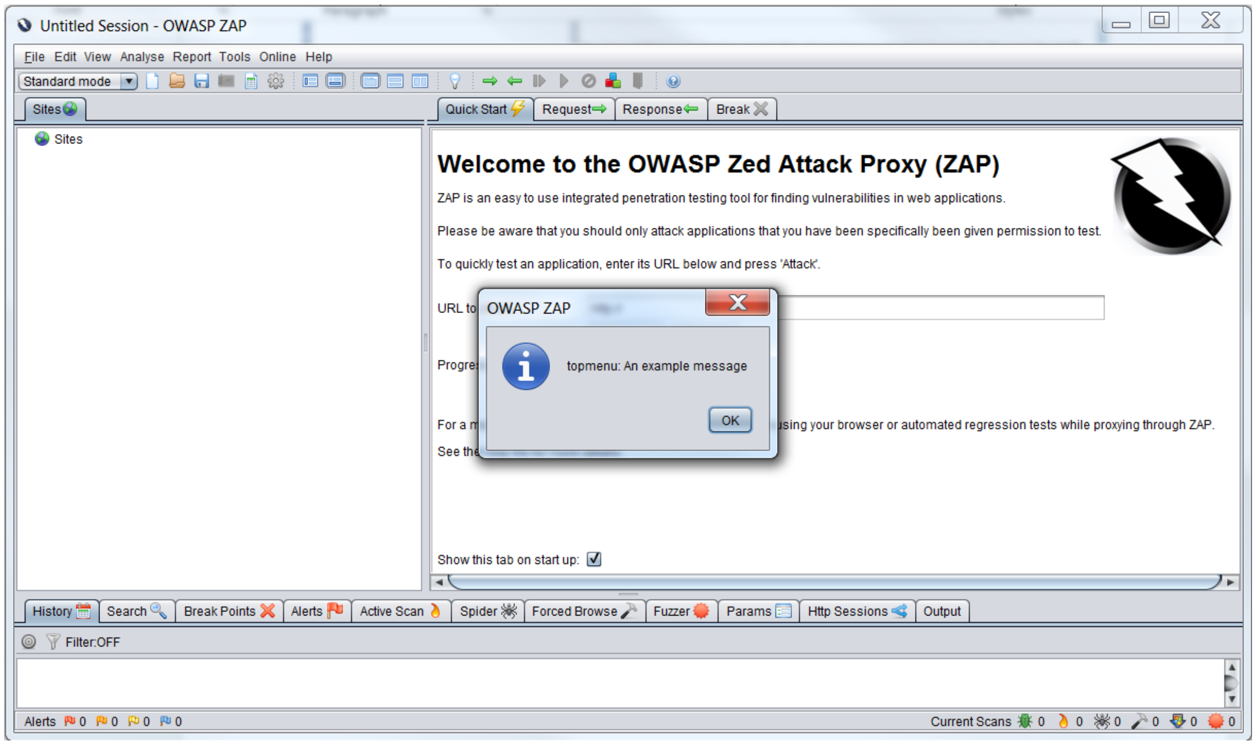
For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.

See the help file for more details.

Show this tab on start up: ☑

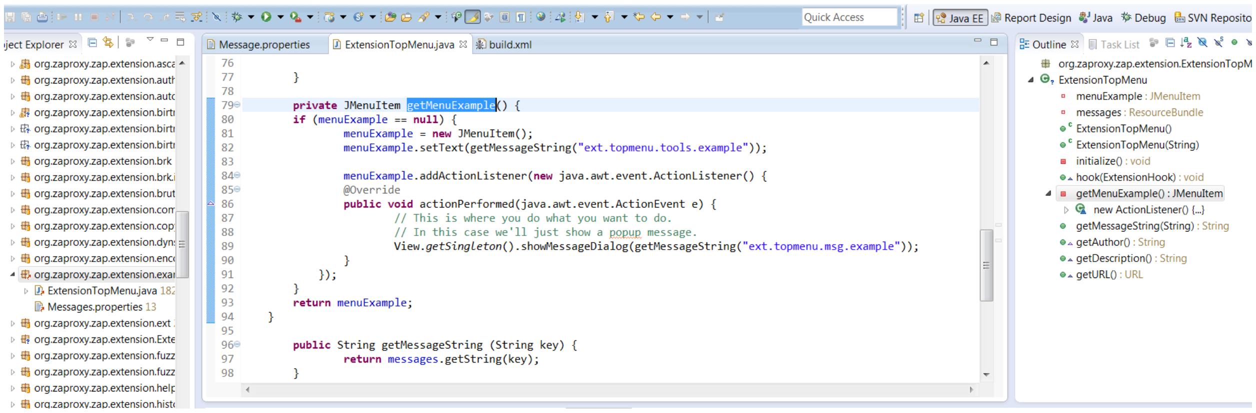
Testing Extension code

Now we have a basic example, all the code does is display a small window


Untitled Session - OWASP ZAP
File Edit View Analyse Report Tools Online Help
Standard mode
Sites
Quick Start
Request
Response
Break
OWASP ZAP
OK
History Search Break Points Alerts Active Scan Spider Forced Browse Fuzzer Params Http Sessions Output
Filter:OFF
Alerts 0 0 0 0
Current Scans 0 0 0 0 0 0

topmenu: An example message

You can use this example to build your own code. Open the Java class ExtensionTopMenu.java in the package and look for the following code:



```java
        }

        private JMenuItem getMenuExample() {
        if (menuExample == null) {
                menuExample = new JMenuItem();
                menuExample.setText(getMessageString("ext.topmenu.tools.example"));

                menuExample.addActionListener(new java.awt.event.ActionListener() {
                @Override
                public void actionPerformed(java.awt.event.ActionEvent e) {
                        // This is where you do what you want to do.
                        // In this case we'll just show a popup message.
                        View.getSingleton().showMessageDialog(getMessageString("ext.topmenu.msg.example"));
                }
            });
        }
        return menuExample;
    }

        public String getMessageString (String key) {
                return messages.getString(key);
        }
```

In the commented code, you can define the methods that you want to call from here

Step 5 (optional): Add New Libraries

Your new code might require the use of new jars of libraries. For development purposes you can place these JAR files in the lib folder

Copy the Jar files into the lib folder in your workspace "..\workspace\zaproxy\lib" then select the Jar files and add then to the Build Path however, for deployment of this extension then we need to convert it as an add-on.

Creating an add-on

Now that we have the example extension ready, you can proceed to make this extension and add on.

Any new add on can be considered in the first development stage 'Alpha' . There are indeed 3 development stages

- Alpha
- Beta
- Final (trunk)

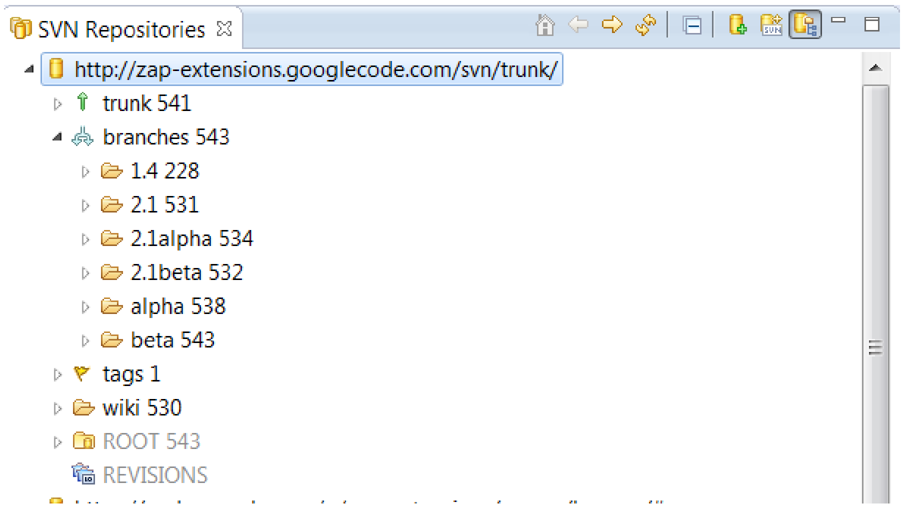
For each one there is a different branch based on the ZAP version.
On the wiki you can find some basic information about building add-on and their relevant code source
http://code.google.com/p/zaproxy/wiki/ZapAddOns

The extension that we have created needs an additional file, a "ZapAddOn.xml" .The ZAP add-on file is a standard jar file, but ideally should include a ZapAddOn.xml file at the top level - this contains information about the contents of the add-on and allows it to be loaded and unloaded dynamically.

```
<zapaddon>
      <name>Short text name (no HTML)</name>
      <version></version>                <!-- Integer than increments with each (released) change -->
      <description>Longer test description (no HTML), but not too long!</description>
```

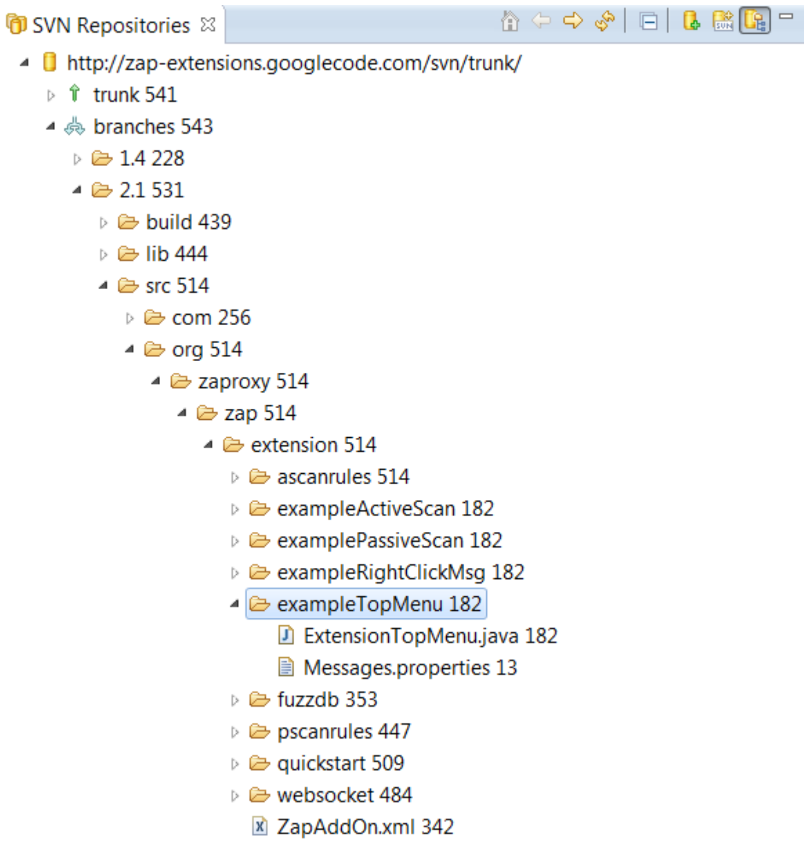
```
    <author>Author (no HTML)</author>
    <url/>                          <!-- Optional URL to more info, eg on the zap-extensions wiki -->
    <changes/>                      <!-- Optional, text summary of changes since the last version -->
    <dependson>                     <!-- Optional, to be used if this addon depends on other addons -->
        <zapaddonid/>               <!-- id (and optionally version) of other addon this one depends on - can be multiple of these -->
    </dependson>
    <extensions>                    <!-- Optional, to be used if this addon includes extensions -->
        <extension/>                <!-- Full class name of the extension - can be multiple of these -->
    </extensions>
    <ascanrules>                    <!-- Optional, to be used if this addon includes active scan rules -->
        <ascanrule/>                <!-- Full class name of the active scan rule - can be multiple of these -->
    </ascanrules>
    <pscanrules>                    <!-- Optional, to be used if this addon includes passive scan rules -->
        <pscanrule/>                <!-- Full class name of the passive scan rule - can be multiple of these -->
    </pscanrules>
    <filters>                       <!-- Optional, to be used if this addon includes filters -->
        <filter/>                   <!-- Full class name of the filter - can be multiple of these -->
    </filters>
    <files>                         <!-- Optional, to be used if this addon includes raw files -->
        <file/>                     <!-- Relative file name (to this package, and the users directory) - can be multiple of these -->
    </files>
    <not-before-version/>           <!-- Optional, to be used if this addon can only be run from the specified version e.g. 2.1.0 -->
    <not-from-version/>             <!-- Optional, to be used if this addon can not run from the specified version e.g. 2.4.0 -->
</zapaddon>
```

For building an add-on, you need actually another build file found in "zap-extensions" project. These zap-extensions repositories are found under http://zap-extensions.googlecode.com/svn



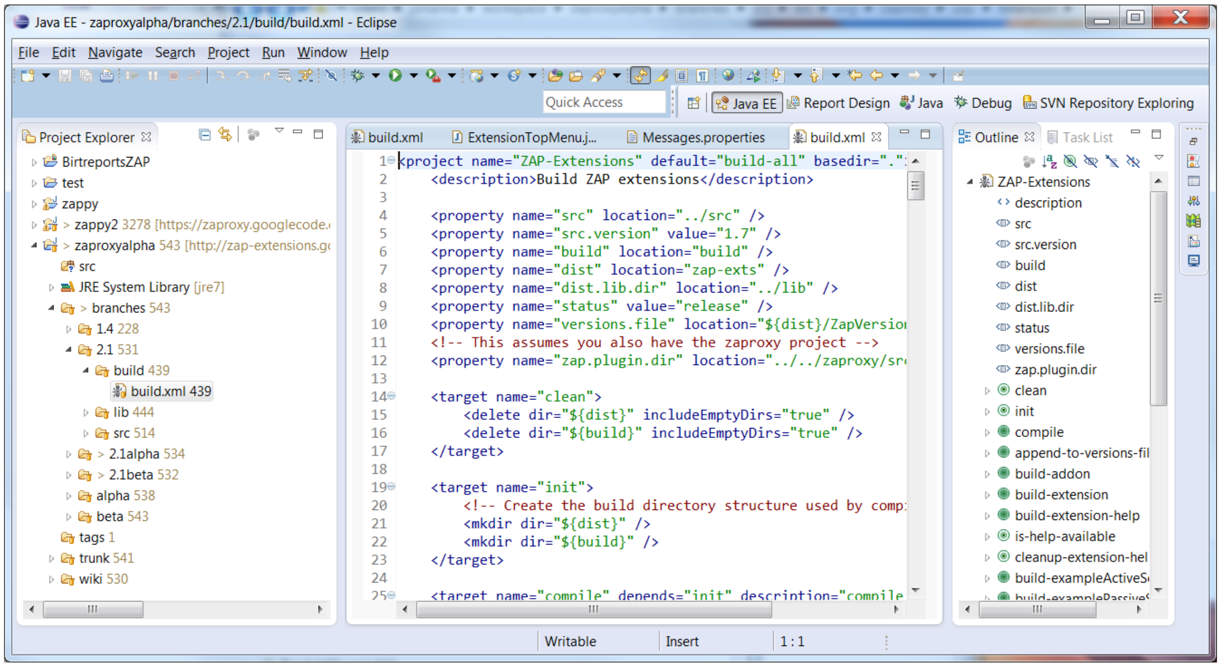
```
http://zap-extensions.googlecode.com/svn/trunk/
    trunk 541
    branches 543
        1.4 228
        2.1 531
        2.1alpha 534
        2.1beta 532
        alpha 538
        beta 543
    tags 1
    wiki 530
    ROOT 543
    REVISIONS
```

In the following branch you can find the extension code ready to be built as an add-on:



```
SVN Repositories
http://zap-extensions.googlecode.com/svn/trunk/
  trunk 541
  branches 543
    1.4 228
    2.1 531
      build 439
      lib 444
      src 514
        com 256
        org 514
          zaproxy 514
            zap 514
              extension 514
                ascanrules 514
                exampleActiveScan 182
                examplePassiveScan 182
                exampleRightClickMsg 182
                exampleTopMenu 182
                  ExtensionTopMenu.java 182
                  Messages.properties 13
                fuzzdb 353
                pscanrules 447
                quickstart 509
                websocket 484
                ZapAddOn.xml 342
```

Go to your workspace where you have setup this repository and look for the build.xml file in the relevant branch



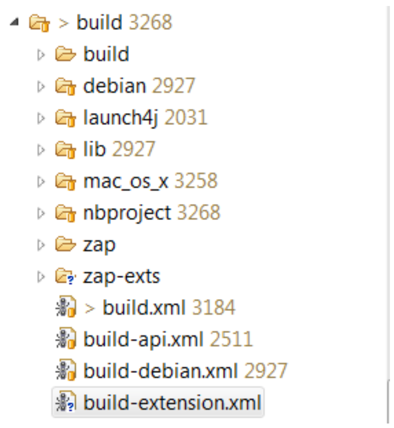
Java EE - zaproxyalpha/branches/2.1/build/build.xml - Eclipse

```
Project Explorer
  BirtreportsZAP
  test
  zappy
  > zappy2 3278 [https://zaproxy.googlecode.
  > zaproxyalpha 543 [http://zap-extensions.g
    src
    JRE System Library [jre7]
    > branches 543
      1.4 228
      2.1 531
        build 439
          build.xml 439
        lib 444
        src 514
      > 2.1alpha 534
      > 2.1beta 532
      alpha 538
      beta 543
    tags 1
    trunk 541
    wiki 530
```

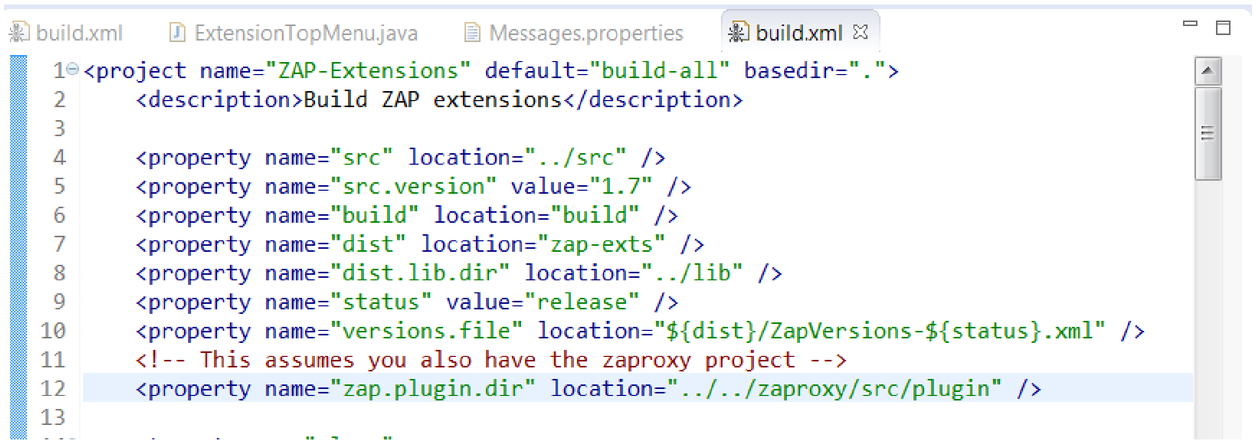
```
<project name="ZAP-Extensions" default="build-all" basedir=".":
    <description>Build ZAP extensions</description>

    <property name="src" location="../src" />
    <property name="src.version" value="1.7" />
    <property name="build" location="build" />
    <property name="dist" location="zap-exts" />
    <property name="dist.lib.dir" location="../lib" />
    <property name="status" value="release" />
    <property name="versions.file" location="${dist}/ZapVersio
    <!-- This assumes you also have the zaproxy project -->
    <property name="zap.plugin.dir" location="../../zaproxy/sr

    <target name="clean">
        <delete dir="${dist}" includeEmptyDirs="true" />
        <delete dir="${build}" includeEmptyDirs="true" />
    </target>

    <target name="init">
        <!-- Create the build directory structure used by comp:
        <mkdir dir="${dist}" />
        <mkdir dir="${build}" />
    </target>

    <target name="compile" depends="init" description="compile
```

```
Outline
  ZAP-Extensions
    description
    src
    src.version
    build
    dist
    dist.lib.dir
    status
    versions.file
    zap.plugin.dir
    clean
    init
    compile
    append-to-versions-fil
    build-addon
    build-extension
    build-extension-help
    is-help-available
    cleanup-extension-hel
    build-exampleActiveS
```

Copy paste this Build.xml file in the workspace repository where you need to build your add-on extension. Since there is a build.xml file, rename it for example to 'build-extension.xml'

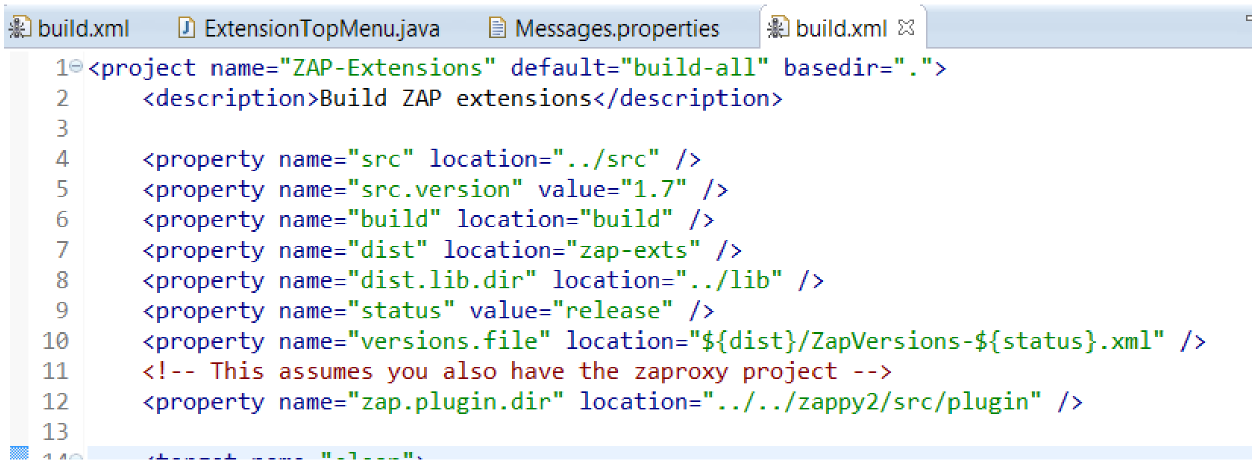

```
> build 3268
    build
    debian 2927
    launch4j 2031
    lib 2927
    mac_os_x 3258
    nbproject 3268
    zap
    zap-exts
    > build.xml 3184
    build-api.xml 2511
    build-debian.xml 2927
    build-extension.xml
```

In the build(extension).xml file look for this code which is important, so the extension add on can be built in the workspace where you have OWASP ZAP configured:



```
<project name="ZAP-Extensions" default="build-all" basedir=".">
    <description>Build ZAP extensions</description>

    <property name="src" location="../src" />
    <property name="src.version" value="1.7" />
    <property name="build" location="build" />
    <property name="dist" location="zap-exts" />
    <property name="dist.lib.dir" location="../lib" />
    <property name="status" value="release" />
    <property name="versions.file" location="${dist}/ZapVersions-${status}.xml" />
    <!-- This assumes you also have the zaproxy project -->
    <property name="zap.plugin.dir" location="../../zaproxy/src/plugin" />

```

Rename in case is necessary (zaproxy) for example like this (in my case my project is called zappy2



```
<project name="ZAP-Extensions" default="build-all" basedir=".">
    <description>Build ZAP extensions</description>

    <property name="src" location="../src" />
    <property name="src.version" value="1.7" />
    <property name="build" location="build" />
    <property name="dist" location="zap-exts" />
    <property name="dist.lib.dir" location="../lib" />
    <property name="status" value="release" />
    <property name="versions.file" location="${dist}/ZapVersions-${status}.xml" />
    <!-- This assumes you also have the zaproxy project -->
    <property name="zap.plugin.dir" location="../../zappy2/src/plugin" />

```

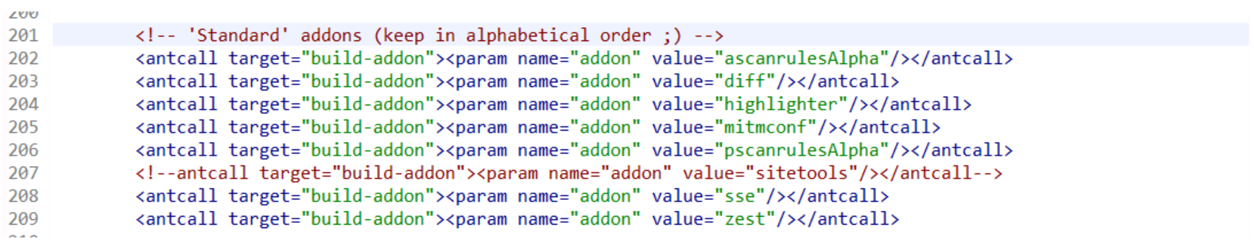
Libraries in add-on

For development purposes we have set the necessary new jars in the lib folder of the application, but for building this extension as an add on, we need to create a new lib folder under the new extension, and place the jar files in here.

Build the add on

Configure a new Add-on using Alpha branch - Build.xml

In the Build.xml found in the alpha branch, you just need to add a line to build your add-on:



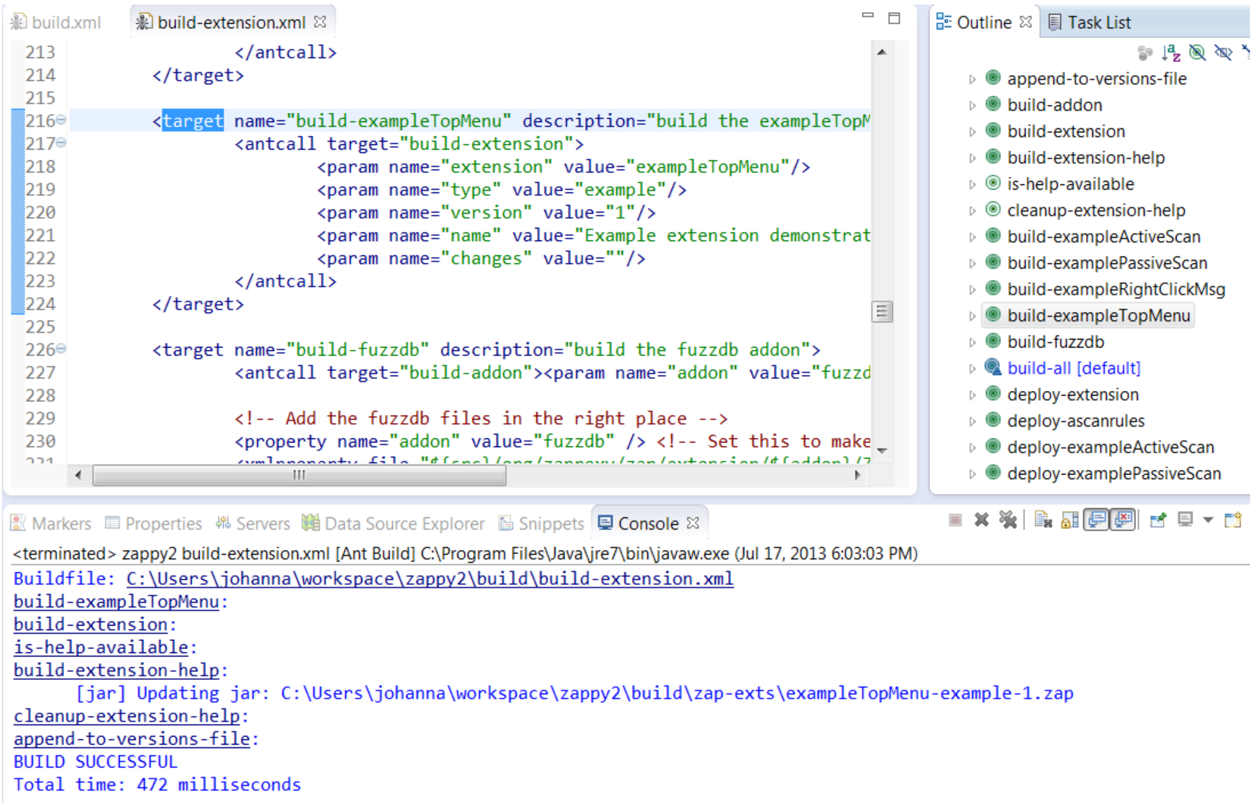
```
        <!-- 'Standard' addons (keep in alphabetical order ;) -->
        <antcall target="build-addon"><param name="addon" value="ascanrulesAlpha"/></antcall>
        <antcall target="build-addon"><param name="addon" value="diff"/></antcall>
        <antcall target="build-addon"><param name="addon" value="highlighter"/></antcall>
        <antcall target="build-addon"><param name="addon" value="mitmconf"/></antcall>
        <antcall target="build-addon"><param name="addon" value="pscanrulesAlpha"/></antcall>
        <!--antcall target="build-addon"><param name="addon" value="sitetools"/></antcall-->
        <antcall target="build-addon"><param name="addon" value="sse"/></antcall>
        <antcall target="build-addon"><param name="addon" value="zest"/></antcall>
```

In this case just add the value, which is the name of your extension.

Go to outline of the build file and look for the task 'buil-all' and run the ant task.

Build Add-On exampleTopMenu

Go to the task into your outline window and look for the task "build-exampleTopMenu". Right click and run the ant-taks



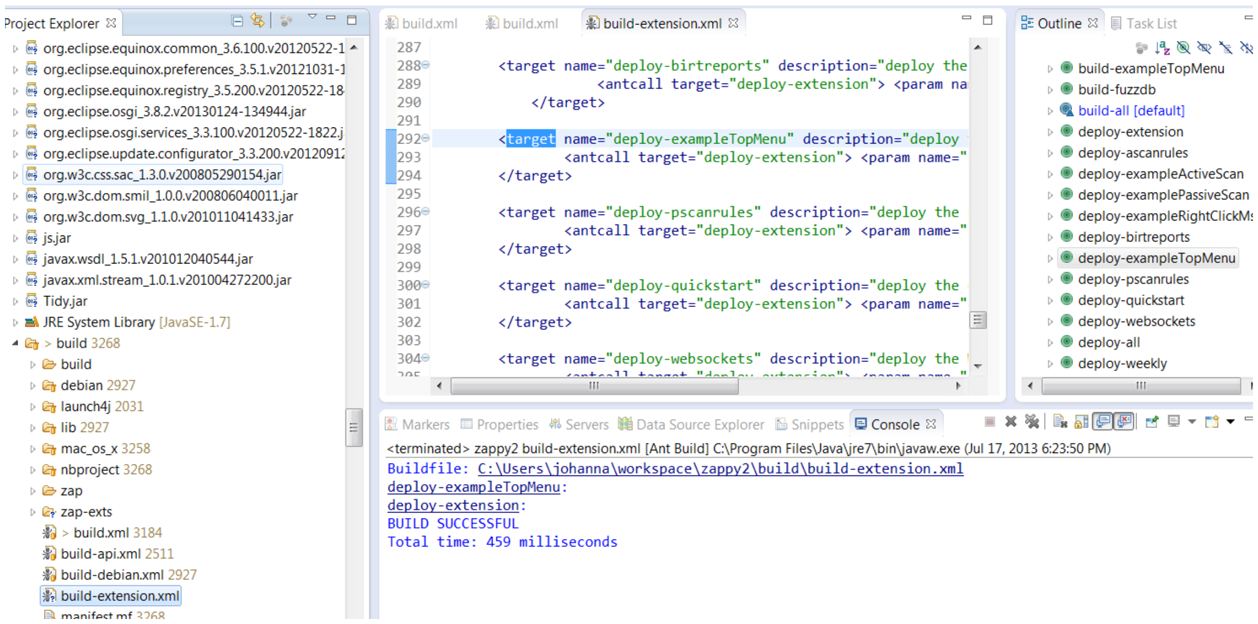
```
                </antcall>
        </target>

        <target name="build-exampleTopMenu" description="build the exampleTopM
                <antcall target="build-extension">
                        <param name="extension" value="exampleTopMenu"/>
                        <param name="type" value="example"/>
                        <param name="version" value="1"/>
                        <param name="name" value="Example extension demonstrat
                        <param name="changes" value=""/>
                </antcall>
        </target>

        <target name="build-fuzzdb" description="build the fuzzdb addon">
                <antcall target="build-addon"><param name="addon" value="fuzzd

                <!-- Add the fuzzdb files in the right place -->
                <property name="addon" value="fuzzdb" /> <!-- Set this to make
```

```
<terminated> zappy2 build-extension.xml [Ant Build] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 17, 2013 6:03:03 PM)
Buildfile: C:\Users\johanna\workspace\zappy2\build\build-extension.xml
build-exampleTopMenu:
build-extension:
is-help-available:
build-extension-help:
      [jar] Updating jar: C:\Users\johanna\workspace\zappy2\build\zap-exts\exampleTopMenu-example-1.zap
cleanup-extension-help:
append-to-versions-file:
BUILD SUCCESSFUL
Total time: 472 milliseconds
```

That's all. You can look at the extension folder "zap-exts" and the add on has been created an a zap file, containing the relevant code of your extension.

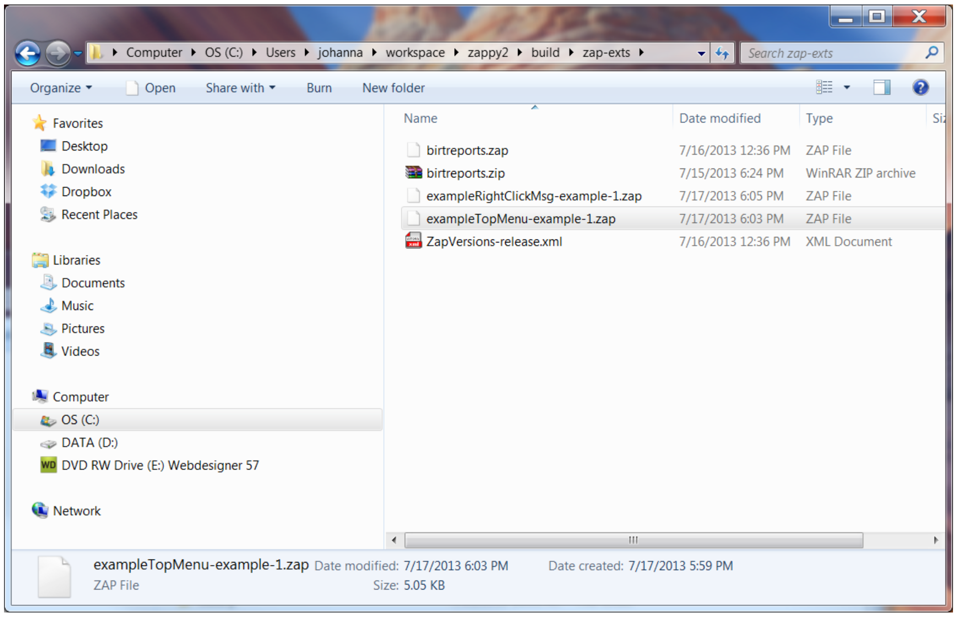
This file already contains the code of the add-on for the "exampleTopMenu" which we already have.

```
<target name="build-exampleTopMenu" description="build the exampleTopMenu extension">
		<antcall target="build-extension">
			<param name="extension" value="exampleTopMenu"/>
			<param name="type" value="example"/>
			<param name="version" value="1"/>
			<param name="name" value="Example extension demonstrating to top 
level menu"/>
			<param name="changes" value=""/>
		</antcall>
	</target>
```


Project Explorer
org.eclipse.equinox.common_3.6.100.v20120522-1
org.eclipse.equinox.preferences_3.5.1.v20121031-1
org.eclipse.equinox.registry_3.5.200.v20120522-18
org.eclipse.osgi_3.8.2.v20130124-134944.jar
org.eclipse.osgi.services_3.3.100.v20120522-1822.j
org.eclipse.update.configurator_3.3.200.v20120912
org.w3c.css.sac_1.3.0.v200805290154.jar
org.w3c.dom.smil_1.0.0.v200806040011.jar
org.w3c.dom.svg_1.1.0.v201011041433.jar
js.jar
javax.wsdl_1.5.1.v201012040544.jar
javax.xml.stream_1.0.1.v201004272200.jar
Tidy.jar
JRE System Library [JavaSE-1.7]
> build 3268
build
debian 2927
launch4j 2031
lib 2927
mac_os_x 3258
nbproject 3268
zap
zap-exts
> build.xml 3184
build-api.xml 2511
build-debian.xml 2927
build-extension.xml
manifest.mf 3268
build.xml
build.xml
build-extension.xml
Outline
Task List
build-exampleTopMenu
build-fuzzdb
build-all [default]
deploy-extension
deploy-ascanrules
deploy-exampleActiveScan
deploy-examplePassiveScan
deploy-exampleRightClickM
deploy-birtreports
deploy-exampleTopMenu
deploy-pscanrules
deploy-quickstart
deploy-websockets
deploy-all
deploy-weekly
Markers
Properties
Servers
Data Source Explorer
Snippets
Console

```

	<target name="deploy-birtreports" description="deploy the
			<antcall target="deploy-extension"> <param na
		</target>

	<target name="deploy-exampleTopMenu" description="deploy
		<antcall target="deploy-extension"> <param name="
	</target>

	<target name="deploy-pscanrules" description="deploy the
		<antcall target="deploy-extension"> <param name="
	</target>

	<target name="deploy-quickstart" description="deploy the
		<antcall target="deploy-extension"> <param name="
	</target>

	<target name="deploy-websockets" description="deploy the
```

```
<terminated> zappy2 build-extension.xml [Ant Build] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 17, 2013 6:23:50 PM)
Buildfile: C:\Users\johanna\workspace\zappy2\build\build-extension.xml
deploy-exampleTopMenu:
deploy-extension:
BUILD SUCCESSFUL
Total time: 459 milliseconds
```

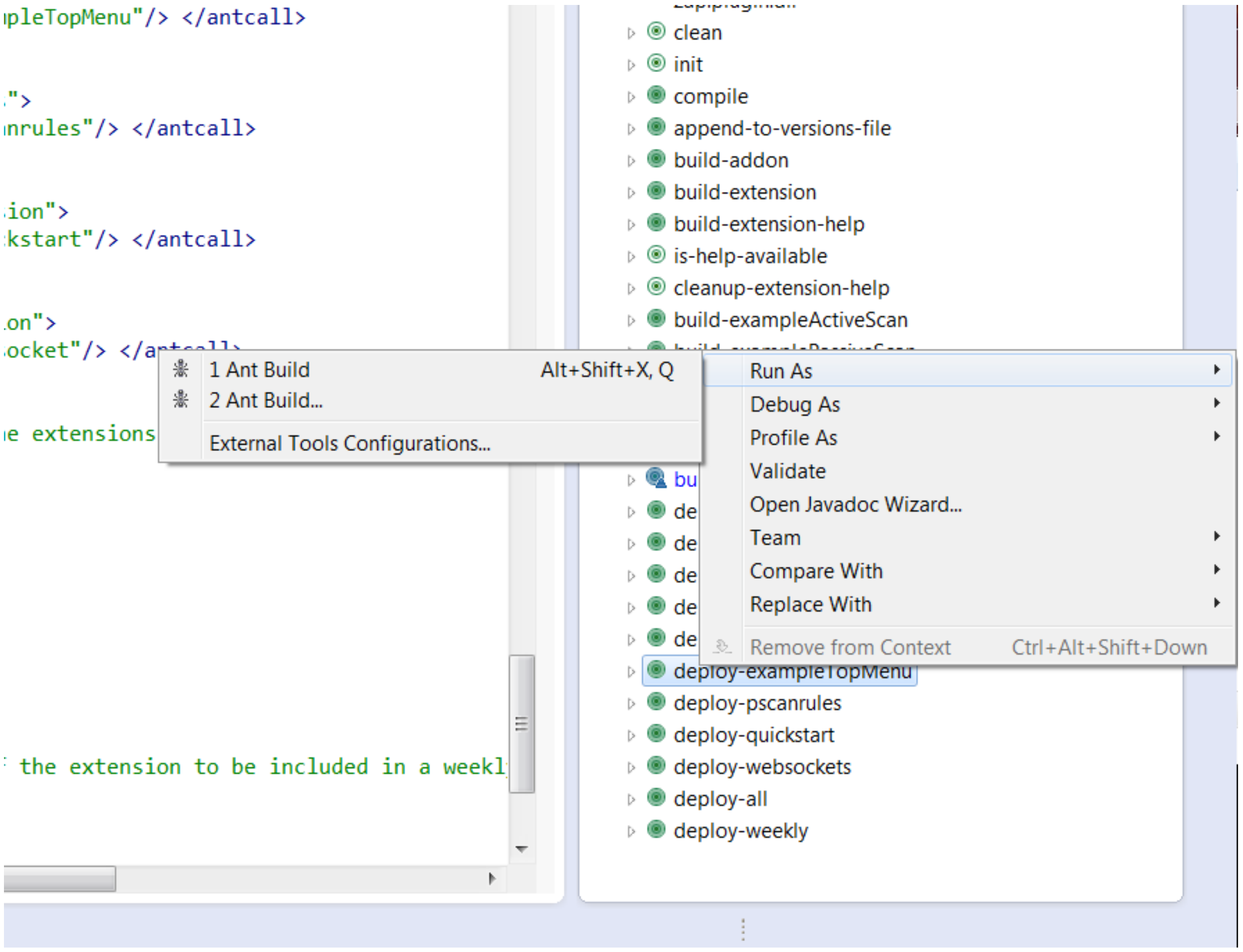


Name	Date modified	Type
birtreports.zap	7/16/2013 12:36 PM	ZAP File
birtreports.zip	7/15/2013 6:24 PM	WinRAR ZIP archive
exampleRightClickMsg-example-1.zap	7/17/2013 6:05 PM	ZAP File
exampleTopMenu-example-1.zap	7/17/2013 6:03 PM	ZAP File
ZapVersions-release.xml	7/16/2013 12:36 PM	XML Document

exampleTopMenu-example-1.zap Date modified: 7/17/2013 6:03 PM Date created: 7/17/2013 5:59 PM
ZAP File Size: 5.05 KB

Deploy the add-on

Go to the task into your outline window and look for the task “deploy-exampleTopMenu”

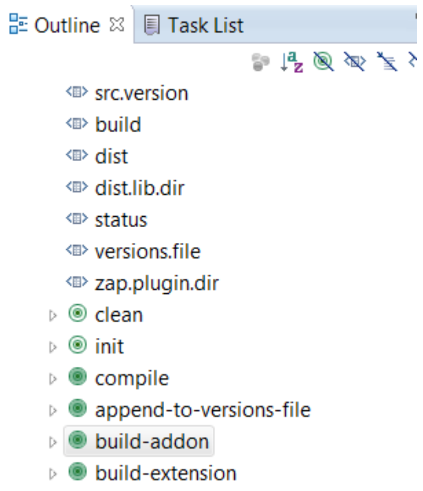


```
ıpleTopMenu"/> </antcall>

.">
ınrules"/> </antcall>

.ion">
:kstart"/> </antcall>

.on">
ocket"/> </antcall>

ıe extensions

 the extension to be included in a weekl
```

The extension can be found in the build folder of your project workspace

Build-Add on code

Go to outline of the build file and look for the task “build-addon” and run the Ant task.



The code of that creates the add on is found in here. Keep in mind that if you want to define a specific build-add on task for your custom add on, you need to create a task for that, but it is much easier to use the alpha branch Build.xml for this purpose.

```
<target name="build-addon" description="build the specified addon">

                <xmlproperty file="${src}/org/zaproxy/zap/extension/${addon}/ZapAddOn.xml"/>
                <property name="file" value="${addon}-${status}-${zapaddon.version}.zap" />

                <antcall target="build-extension-help">
                        <param name="extension" value="${addon}"/>
                </antcall>

                <jar jarfile="${dist}/${file}" update="true" compress="true">
                        <zipfileset dir="${build}" prefix="">
                                <include name="org/zaproxy/zap/extension/${addon}/**"/>
                        </zipfileset>
                        <zipfileset dir="${src}" prefix="">
                                <include name="org/zaproxy/zap/extension/${addon}/Messages*"/>
                        </zipfileset>
                        <zipfileset dir="${src}" prefix="">
                                <include name="org/zaproxy/zap/extension/${addon}/resource/**"/>
                        </zipfileset>
                        <zipgroupfileset dir="${src}/org/zaproxy/zap/extension/${addon}/lib/"
includes="*.jar" erroronmissingdir="false"/>
                        <zipfileset dir="${src}"
includes="org/zaproxy/zap/extension/${addon}/ZapAddOn.xml" fullpath="ZapAddOn.xml"/>
                </jar>

                <antcall target="cleanup-extension-help">
                        <param name="extension" value="${addon}"/>
                </antcall>

                <antcall target="append-to-versions-file">
                        <param name="extension" value="${addon}"/>
                        <param name="name" value="${zapaddon.name}"/>
                        <param name="version" value="${zapaddon.version}"/>
                        <param name="description" value="${zapaddon.description}"/>
                        <param name="author" value="${zapaddon.author}"/>
                        <param name="url" value="${zapaddon.url}"/>
                        <param name="changes" value="${zapaddon.changes}"/>
                        <param name="file" value="${file}"/>
                        <param name="not-before-version" value="${zapaddon.not-before-version}"/>
                        <param name="not-from-version" value="${zapaddon.not-from-version}"/>
                </antcall>
        </target>
```