

Creating OWASP ZAP Extensions and Add-ons

Table of Contents

Introduction	1
What are zap extensions?	1
Code conventions for developing new extensions	2
My first Extension	3
Step 1: Download source code and Build ZAP	3
Step 2: Create new extension	3
Step 3: Testing Extension code	10
Step 4 (optional) :Add New Libraries	10
Creating an add-on	11
Libraries in add-on	14
Build the add on.....	14
Deploy the add-on	16

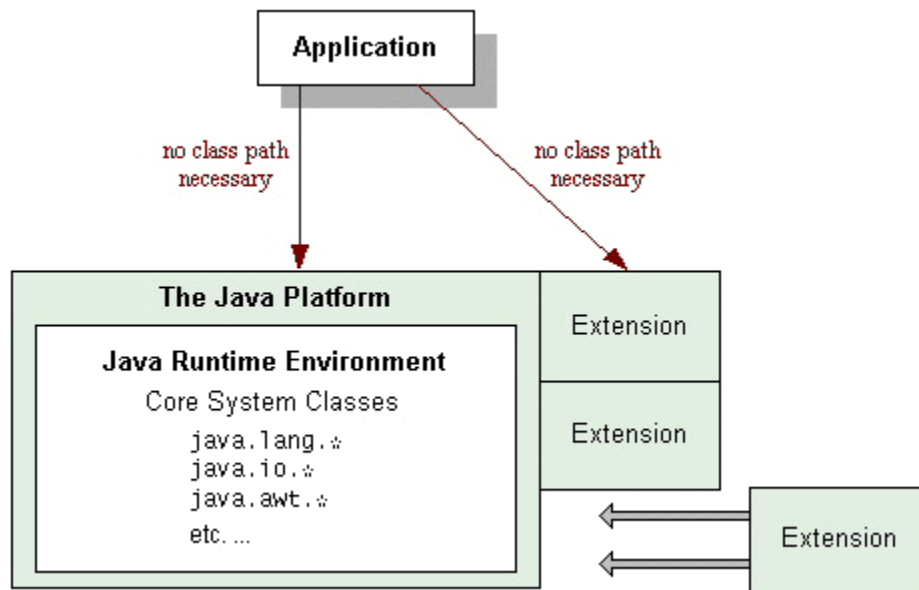
Introduction

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 extensions.

```

└─ 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 is a concept proposed by Simon Bennett (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:

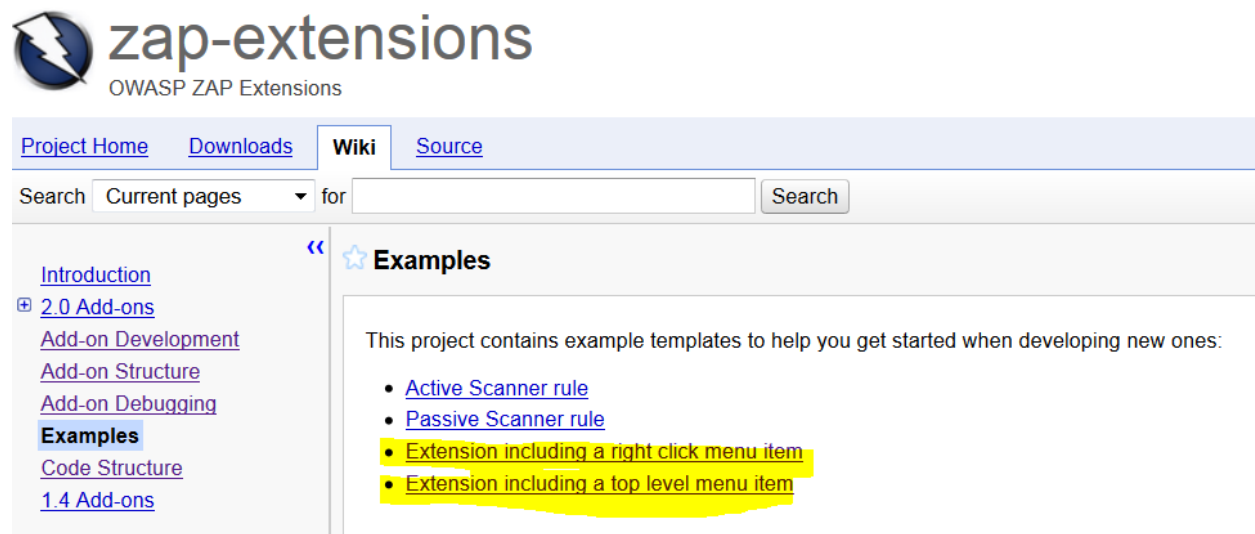
- All significant new code should be under the [org.zaproxy.zap](#) package.

- 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.



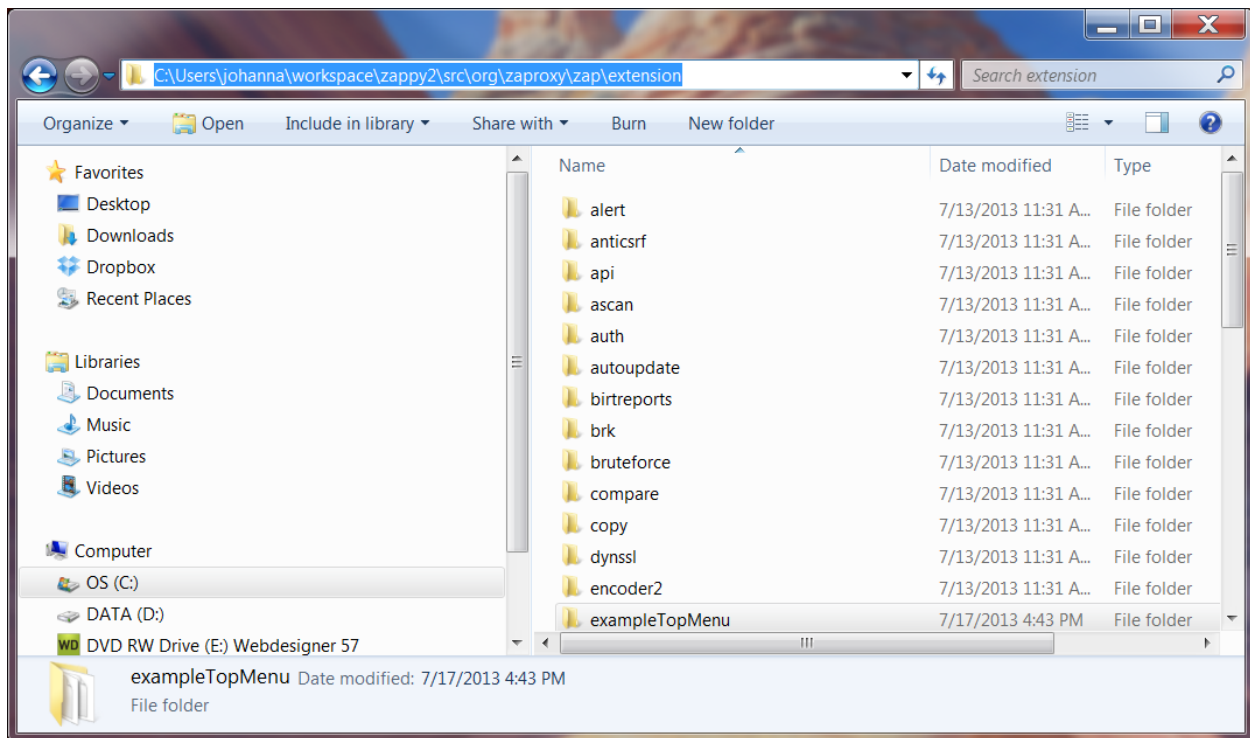
From this, we will create the TOP-LEVEL menu item

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\zaprox\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

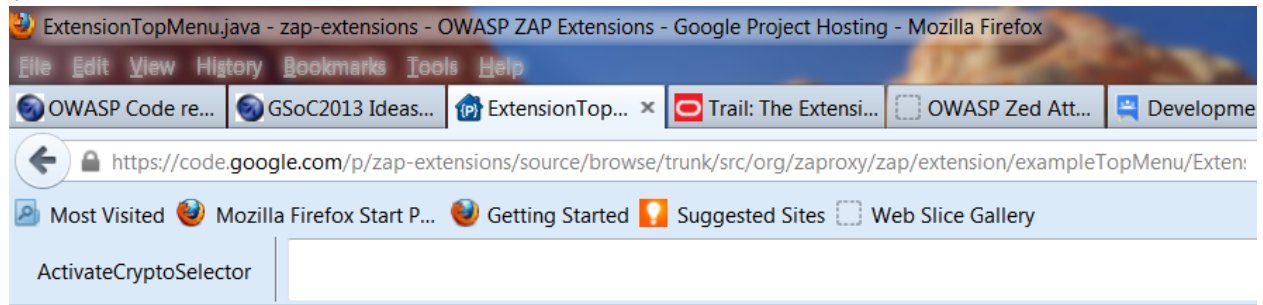


<https://code.google.com/p/zap-extensions/source/browse/trunk/src/org/zaproxy/zap/extension/exampleTopMenu/ExtensionTopMenu>

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

.java



[Project Home](#) [Downloads](#) [Wiki](#) **[Source](#)**

[Checkout](#) **[Browse](#)** [Changes](#)

Source path: [svn/](#) [trunk/](#) [src/](#) [org/](#) [zaproxy/](#) [zap/](#) [extension/](#) [exampleTopMenu/](#) ExtensionTopMenu.java

```
1  /*
2   * Zed Attack Proxy (ZAP) and its related class files.
3   *
4   * ZAP is an HTTP/HTTPS proxy for assessing web application security.
5   *
6   * Licensed under the Apache License, Version 2.0 (the "License");
7   * you may not use this file except in compliance with the License.
8   * You may obtain a copy of the License at
9   *
10  *   http://www.apache.org/licenses/LICENSE-2.0
11  *
12  * Unless required by applicable law or agreed to in writing, software
13  * distributed under the License is distributed on an "AS IS" BASIS,
14  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  * See the License for the specific language governing permissions and
16  * limitations under the License.
17  */
18  package org.zaproxy.zap.extension.exampleTopMenu;
19
20  import java.net.MalformedURLException;
21  import java.net.URL;
22  import java.util.ResourceBundle;
```

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 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;
        }
    }
}
}

```

Message.properties file:

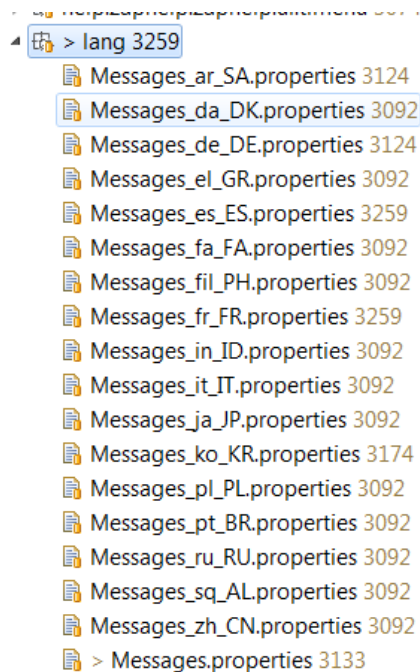
```

# 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

```

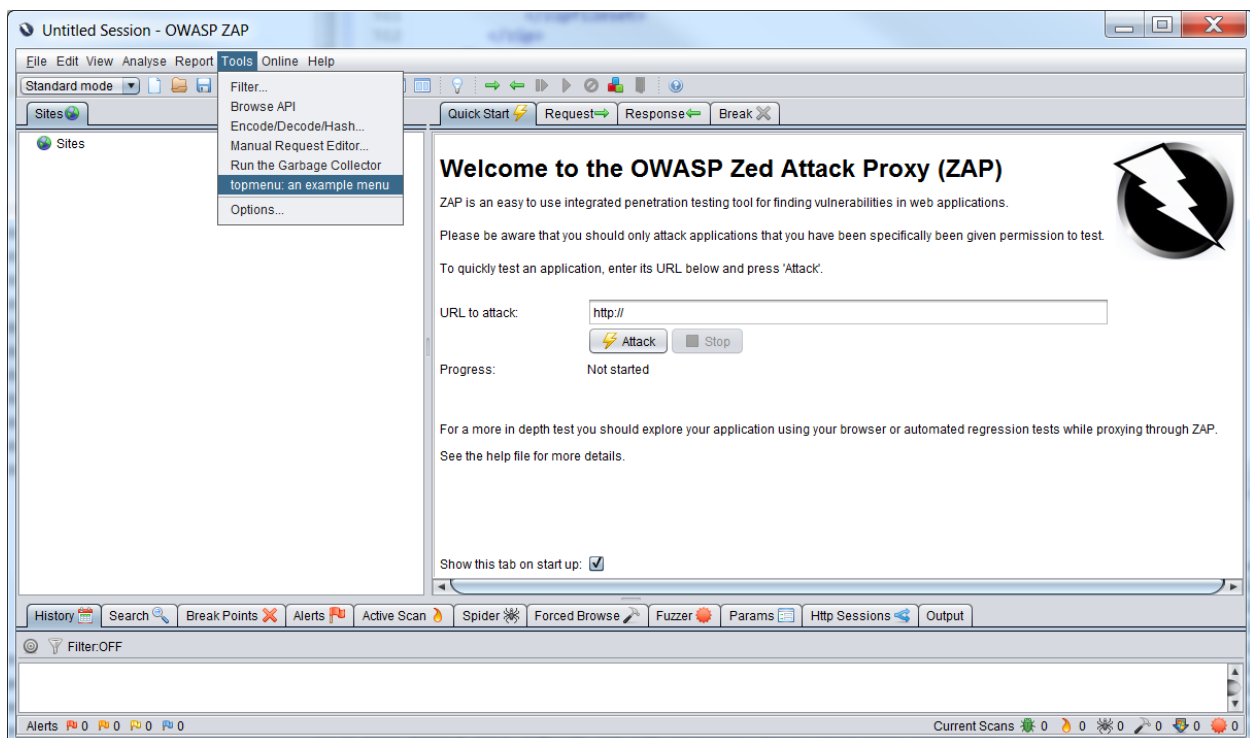
This text is used in the menu application. In fact, once you get acquaintance with the OWASP ZAP code, you will find this file and multiple translation text files under 'lang' package



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.

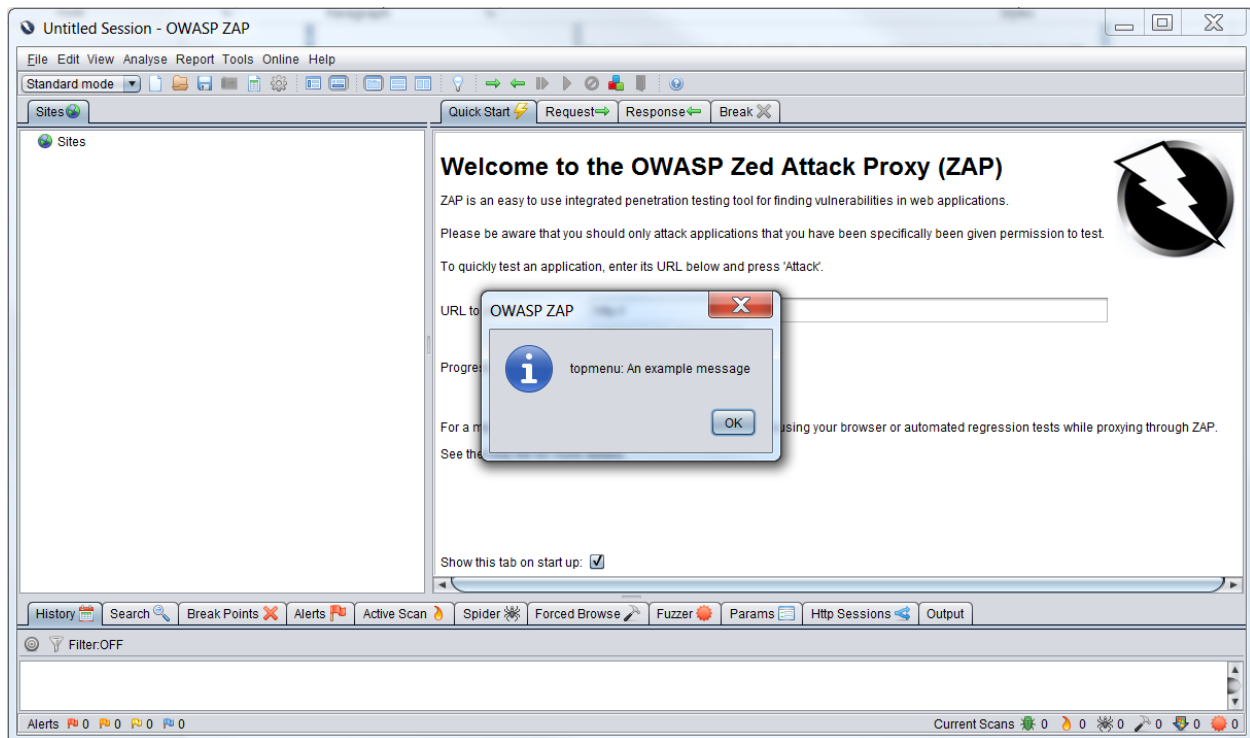
Right click the ‘src’ again and select Run

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:

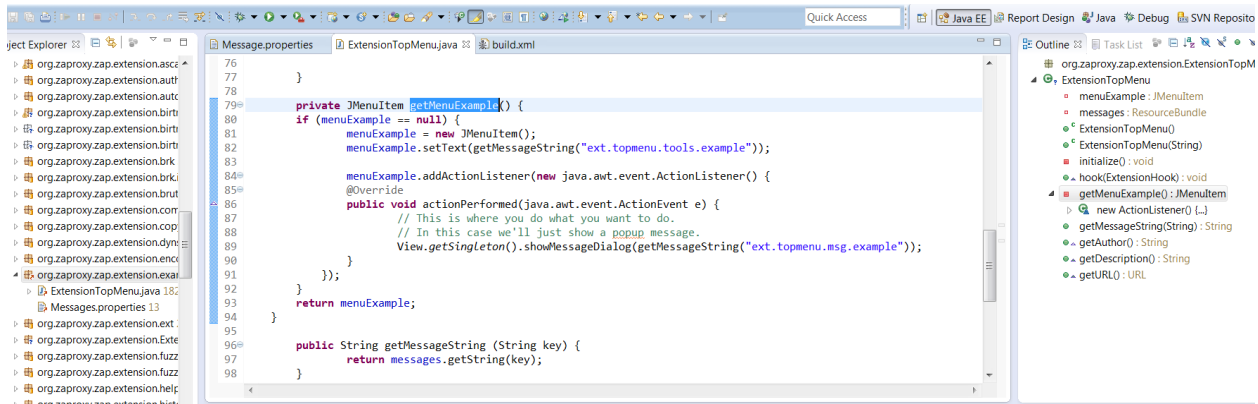


Step 3: Testing Extension code

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



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:



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

Step 4 (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\zaproxylib`” then select the Jar files and add them 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 also based on the version of ZAP.

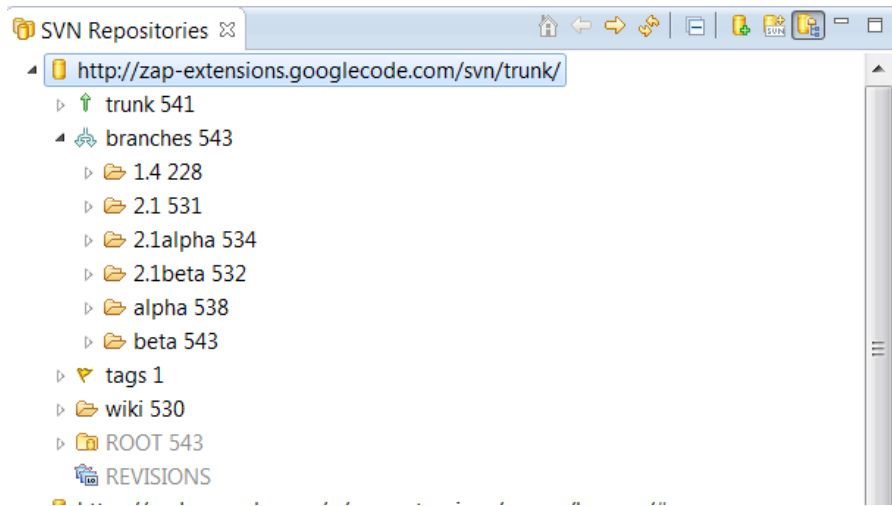
On the wiki you can find some basic information about building add-on and their relevant code source

<http://code.google.com/p/zaproxy/wiki/ZapAddOns>

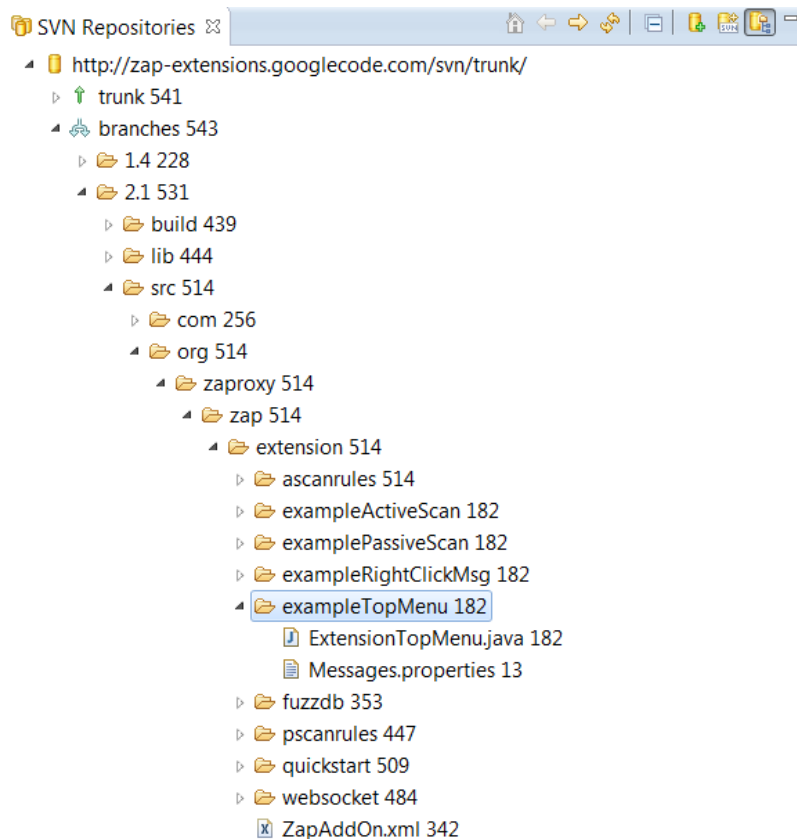
The extension that we have created needs an additional file, a "ZapAddOn.xml" .The ZAP add-on file is a standard jar file, but ideally should include a [ZapAddOn.xml](#) file at the top level - this contains information about the contents of the add-on and allows it to be loaded and unloaded dynamically.

```
<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>
```

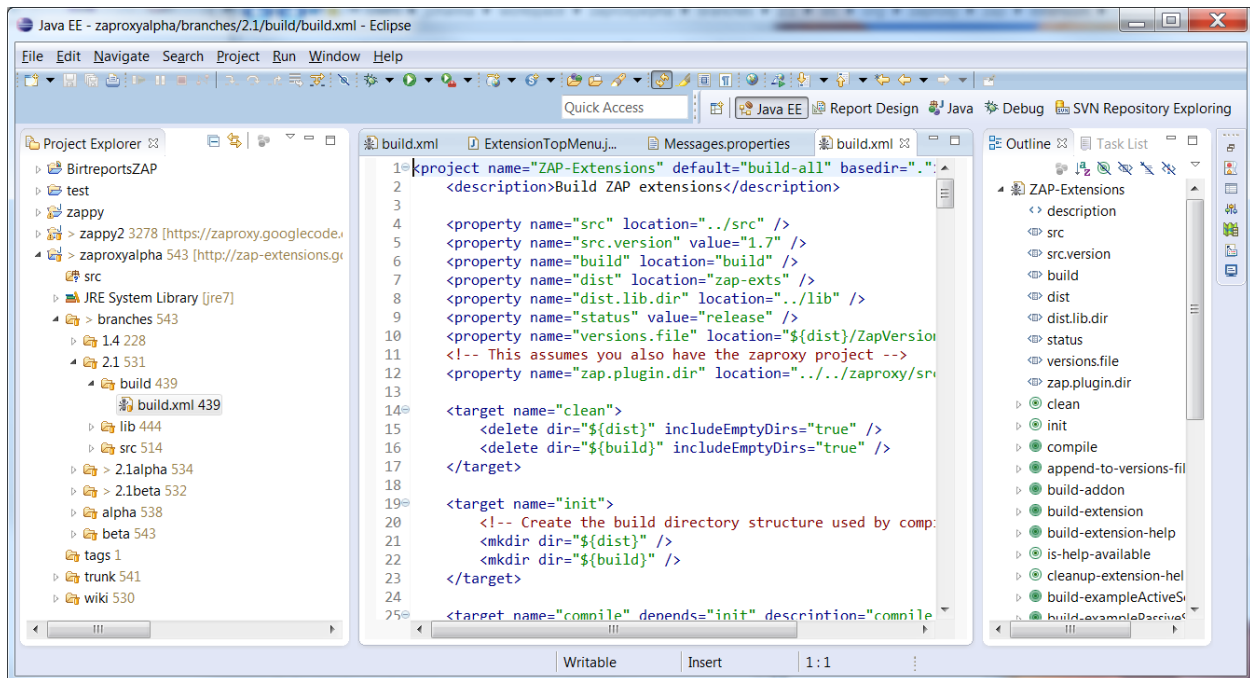
For building an add-on, you need actually another build file found in “zap-extensions” project. These zap-extensions repositories are found under <http://zap-extensions.googlecode.com/svn>



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



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



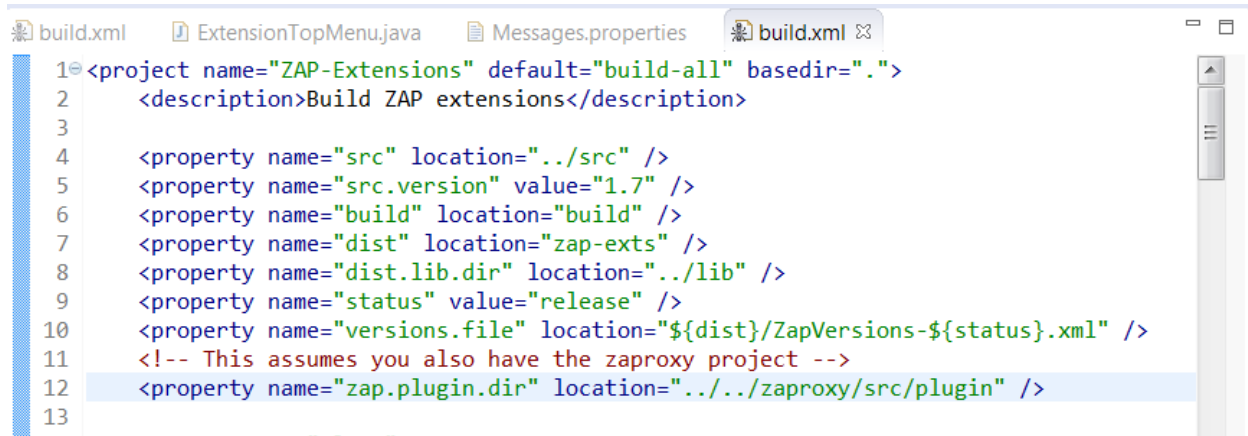
Copy paste this Build.xml file in the workspace repository where you need to build your add-on extension. Since there is a build.xml file, rename it for example to 'build-extension.xml'

```

└─ 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

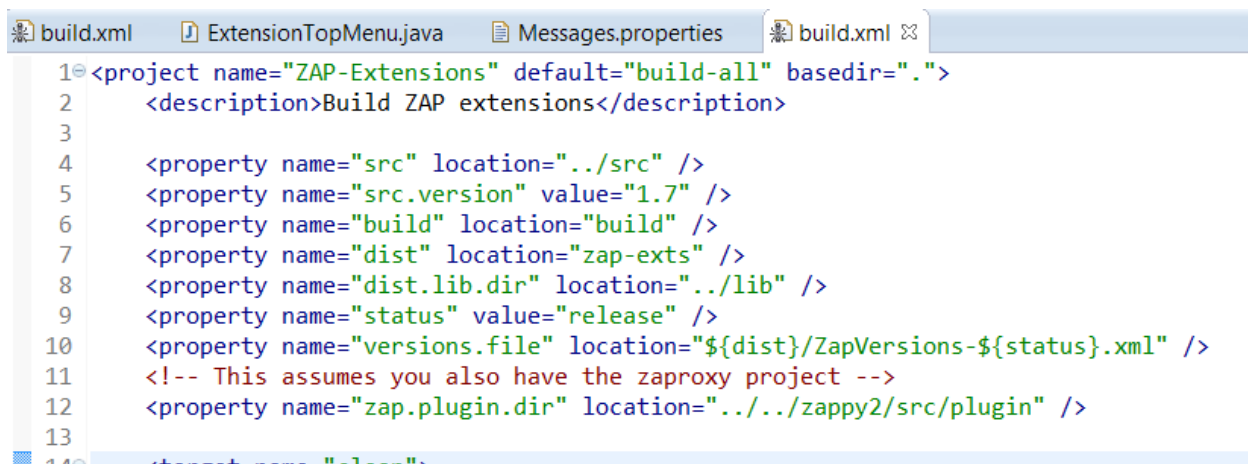
```

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:



```
1 <project name="ZAP-Extensions" default="build-all" basedir=".">
2   <description>Build ZAP extensions</description>
3
4   <property name="src" location="../src" />
5   <property name="src.version" value="1.7" />
6   <property name="build" location="build" />
7   <property name="dist" location="zap-exts" />
8   <property name="dist.lib.dir" location="../lib" />
9   <property name="status" value="release" />
10  <property name="versions.file" location="${dist}/ZapVersions-${status}.xml" />
11  <!-- This assumes you also have the zapproxy project -->
12  <property name="zap.plugin.dir" location="../../zapproxy/src/plugin" />
13
```

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



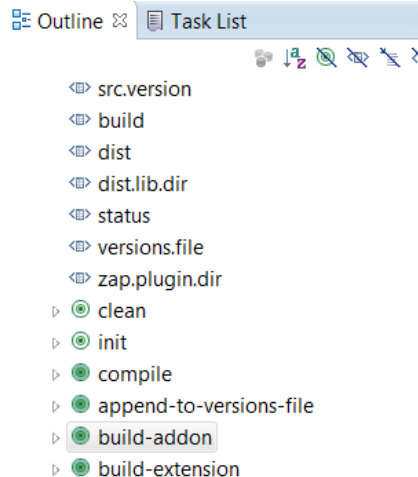
```
1 <project name="ZAP-Extensions" default="build-all" basedir=".">
2   <description>Build ZAP extensions</description>
3
4   <property name="src" location="../src" />
5   <property name="src.version" value="1.7" />
6   <property name="build" location="build" />
7   <property name="dist" location="zap-exts" />
8   <property name="dist.lib.dir" location="../lib" />
9   <property name="status" value="release" />
10  <property name="versions.file" location="${dist}/ZapVersions-${status}.xml" />
11  <!-- This assumes you also have the zapproxy project -->
12  <property name="zap.plugin.dir" location="../../zappy2/src/plugin" />
13
```

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

Go to outline of the build file and look for the task "build-addon"



The code of this must be adapted so it builds the add on you need

```
<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>
```

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>

```

Go to the task into your outline window and look for the task “build-exampleTopMenu”. Right click and run the ant-tasks

The screenshot shows an IDE with the following components:

- build.xml file:**

```

213         </antcall>
214     </target>
215
216     <target name="build-exampleTopMenu" description="build the exampleTopMenu">
217         <antcall target="build-extension">
218             <param name="extension" value="exampleTopMenu"/>
219             <param name="type" value="example"/>
220             <param name="version" value="1"/>
221             <param name="name" value="Example extension demonstrating to top
level menu"/>
222             <param name="changes" value=""/>
223         </antcall>
224     </target>
225
226     <target name="build-fuzzdb" description="build the fuzzdb addon">
227         <antcall target="build-addon"><param name="addon" value="fuzzdb"></antcall>
228
229         <!-- Add the fuzzdb files in the right place -->
230         <property name="addon" value="fuzzdb" /> <!-- Set this to make
231         <!-- Set this to make

```
- Task List:** A list of tasks including 'append-to-versions-file', 'build-addon', 'build-extension', 'build-extension-help', 'is-help-available', 'cleanup-extension-help', 'build-exampleActiveScan', 'build-examplePassiveScan', 'build-exampleRightClickMsg', 'build-exampleTopMenu' (selected), 'build-fuzzdb', 'build-all [default]', 'deploy-extension', 'deploy-ascanrules', 'deploy-exampleActiveScan', and 'deploy-examplePassiveScan'.
- Console Output:**

```

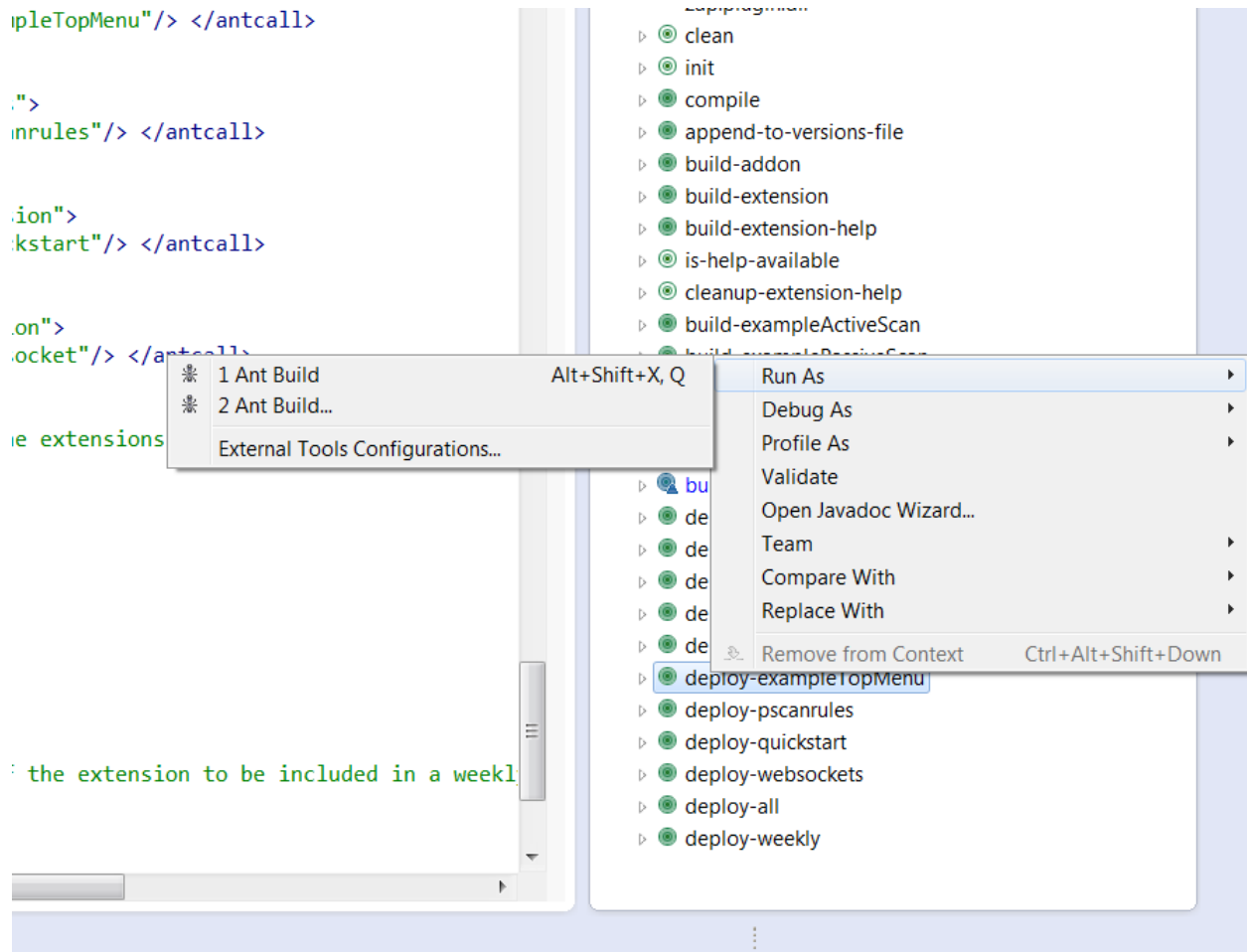
<terminated> zappy2 build-extension.xml [Ant Build] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 17, 2013 6:03:03 PM)
Buildfile: C:\Users\johanna\workspace\zappy2\build\build-extension.xml
build-exampleTopMenu:
build-extension:
is-help-available:
build-extension-help:
[jar] Updating jar: C:\Users\johanna\workspace\zappy2\build\zap-exts\exampleTopMenu-example-1.zip
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.

Deploy the add-on

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



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

