Plugins Packaging

This page describes how plugins should be packaged by plugin developers. See Installing Additional Plugins for instruction on plugin installation.

To write TeamCity plugin it is beneficial to have some knowledge about Spring Framework. Server-side and agent-side TeamCity plugins are initialized via Spring container. This means that every plugin should have a Spring bean definition file describing main plugin services.

There is a convention how bean definition file should be named:

- build-server-plugin-<plugin name>*.xml — for server side plugins
- build-agent-plugin-<plugin name>*.xml — for agent side plugins

where an asterisk can be replaced with any text, for example: build-server-plugin-cvs.xml. Bean definition files should be placed into the `META-INF` folder of the JAR archive containing plugin classes.

Web resources packaging

In most cases plugin is just a number of classes packed into a JAR file, but if you wish to write custom page for TeamCity, most likely you'll need to place images, CSS, JavaScript, and JSP files somewhere. Files that you want to access via hyperlinks and JSP pages you should place into the buildServerResources subfolder. Upon server startup these files will be extracted from the archive and moved into the <TeamCity home>/webapps/ROOT/plugins/<plugin name> directory (read more on how to construct paths to your JSP files).

Installation of TeamCity plugins

TeamCity is able to load plugin from the following directories:

- `<TeamCity data directory>/plugins` – user-installed plugins
- `<TeamCity web application>/WEB-INF/plugins` — default directory for bundled TeamCity plugins

Plugins with the same name (for example, newer version) put into `<TeamCity data directory>/plugins` will override plugins put into `<TeamCity web application>/WEB-INF/plugins` directory.

You can put plugin into `plugins` directory as a separate folder or as a zip archive. Note that server must be restarted to load your plugin.

If you use a separate folder:

- TeamCity will use the folder name as plugin name

If you use a zip file:

- TeamCity will use name of the zip file as the plugin name
- Plugin zip file will be automatically unpacked on server startup to directory with the same name

Inside plugin directory there should be the following structure:

```
agent
  |
  --> <agent plugin zip>
server
  |
  --> <server plugin jar files>
teamcity-plugin.xml
```

agent directory must contain one file only: <agent plugin zip> which should be prepared so that all files and directories are placed into the single root directory. That is, there must be one root folder in the archive (<plugin top level directory> in the diagram below), and there should not be other files at the top level. Usually for convenience the name of <plugin top level directory> is the same as the plugin name. All jar files required by the plugin on the agent must be placed to the lib subfolder:

```
<plugin top level directory>
 |
 --> lib
      |
      --> <jar files>
```

server folder contains server side part of the plugin, that is, a bunch of jar files.

teamcity-plugin.xml contains some meta information about plugin, like its name and version, see below.

Plugins Loading

On the agent side all plugins and their libraries are always loaded by shared classloader.

On the server side by default plugins are loaded in the shared classloader too, however it is possible to load plugin in separate classloader by specifying special parameters in the `teamcity-plugin.xml` file.

Agent upgrade

When server starts it places all agent plugins into the `<TeamCity home>/webapps/ROOT/update/plugins` folder. After that if content of the folder has changed, agents will receive upgrade command from the server and download updated files automatically. Note that server tracks changes in this folder and can initiate agent upgrade procedure on the fly. In practice it means that if you want to deploy updated agent part of your plugin without server restart you can put your agent plugin to this folder.

After successful upgrade your plugin will be unpacked into the `<Agent home>/plugins/` folder. Note that if agent is busy running build it won't upgrade until the build finishes. Also no new builds will start on the agent if it should be updated.

Plugin Descriptor (**`teamcity-plugin.xml`**)

`teamcity-plugin.xml` file should be placed to the root of plugin folder. You can refer to XSD schema for this file which is unpacked to `<TeamCity data directory>/config/teamcity-plugin-descriptor.xsd`

An example of teamcity-plugin.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<teamcity-plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:noNamespaceSchemaLocation="urn:shemas-jetbrains-com:teamcity-plugin-v1-xml">
  <info>
    <name>PluginName</name> <!-- the name of plugin used in teamcity -->
    <display-name>This name may be used for UI</display-name>
    <version>0.239.42</version>
  </info>
  <deployment use-separate-classloader="true" /> <!-- load server plugin's classes in separate
classloader-->
</teamcity-plugin>
```

⚠ Please, note that `use-separate-classloader="true"` parameter is for server-side plugins only, it does not affect agent-side plugins. Until you are using some libraries which clash with TeamCity libraries, it is recommended to leave default behavior, that is, to use shared classloader.

Server side plugins also can obtain jetbrains.buildServer.web.openapi.PluginDescriptor implementation with help of SpringFramework Injecting dependencies,
Autowiring collaborators autowiring feature.

Starting from TeamCity 5.0 it is allowed to define plugin parameters in the teamcity-plugin.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<teamcity-plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:noNamespaceSchemaLocation="urn:shemas-jetbrains-com:teamcity-plugin-v1-xml">
  <info>
    <name>PluginName</name> <!-- the name of plugin used in teamcity -->
    <display-name>This name may be used for UI</display-name>
    <version>0.239.42</version>
  </info>
  <deployment use-separate-classloader="true" /> <!-- load server plugin's classes in separate
classloader-->
  <parameters>
    <parameter name="key">value</parameter>
    <!-- ... -->
  </parameters>
</teamcity-plugin>
```

Plugin parameters can be accessed from jetbrains.buildServer.web.openapi.PluginDescriptor#getParameterValue(String) metho
d.