Relative Project Paths

When you want to save assets from scripts in your Unity project, you may often run into an error explaining that you must use relative paths.

This simple class *(should)* converts all paths into relative paths:

```
using System;
using System.IO;
using System.Linq;
using UnityEditor;
using UnityEditor.PackageManager;
using UnityEditor.PackageManager.Requests;
using UnityEngine;

namespace Varneon.Editor
{
    public static class PathUtility
    {
        /// <summary>
        /// Converts full path to one relative to the project
        /// </summary>
        /// <param name="path">Full path pointing inside the project</param>
        /// <returns>Path relative to the project</returns>
        /// <exception cref="ArgumentException"/>
        public static string ConvertToRelativePath(string path)
        {
            // If string is null or empty, throw an exception
            if (string.IsNullOrEmpty(path)) { throw new ArgumentException("Invalid path!",
nameof(path)); }

            // If the directory is already valid, return original path
            if (AssetDatabase.IsValidFolder(Path.GetDirectoryName(path))) { return path; }

            // Get the 'Assets' directory
            string assetsDirectory = Application.dataPath;

            // Get the project's root directory (Trim 'Assets' from the end of the path)
```

```
            string projectDirectory = assetsDirectory[..^6];

            // Ensure that the path is the full path
            path = Path.GetFullPath(path);

            // Replace backslashes with forward slashes
            path = path.Replace('\\', '/');

            // If path doesn't point inside the project, scan all packages
            if (!path.StartsWith(projectDirectory))
            {
                // Request all packages in offline mode
                ListRequest request = Client.List(true, false);

                // Wait until the request is completed
                while (!request.IsCompleted) { }

                // If the request was successful, proceed with scanning the packages
                if(request.Status == StatusCode.Success)
                {
                    // Try to find a package with same path as the one we are validating
                    UnityEditor.PackageManager.PackageInfo info =
request.Result.FirstOrDefault(p => p.source.Equals(PackageSource.Local) &&
path.StartsWith(p.resolvedPath.Replace('\\', '/')));

                    // If a package with same path exists, return resolved path
                    if (info != null)
                    {
                        string resolvedPackagePath = info.resolvedPath.Replace('\\', '/');

                        string packagePath =
resolvedPackagePath[..resolvedPackagePath.LastIndexOf('/')];

                        return string.Concat("Packages/", info.name,
path[resolvedPackagePath.Length..]);
                    }
                }

                throw new ArgumentException("Path is not located in this project!",
nameof(path));
            }
```

```
            // Return a path relative to the project
            return path.Replace(projectDirectory, string.Empty);
        }
    }
}
```

Revision #3
Created 9 December 2022 19:40:14 by Varneon
Updated 18 November 2023 09:11:38 by Varneon