LinksPlatform's Platform.Bot Class Library

1.1 ./csharp/Platform.Bot/Program.cs

```csharp
using Interfaces;
using Octokit;
using Platform.Exceptions;
using Platform.IO;
using Storage.Local;
using Storage.Remote.GitHub;
using System;
using System.Collections.Generic;

namespace Platform.Bot
{
    /// <summary>
    /// <para>
    /// Represents the program.
    /// </para>
    /// <para></para>
    /// </summary>
    internal class Program
    {
        private static void Main(string[] args)
        {
            using var cancellation = new ConsoleCancellation();
            var argumentIndex = 0;
            var username = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Username", args);
            var token = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Token", args);
            var appName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "App Name", args);
            var databaseFileName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Database
                file name", args);
            var fileSetName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "File set name
                ", args);
            var dbContext = new FileStorage(databaseFileName);
            Console.WriteLine($"Bot has been started. {Environment.NewLine}Press CTRL+C to
                close");
            try
            {
                var api = new GitHubStorage(username, token, appName);
                new ProgrammerRole(
                    new List<ITrigger<Issue>> {
                        new HelloWorldTrigger(api, dbContext, fileSetName),
                        new OrganizationLastMonthActivityTrigger(api),
                        new LastCommitActivityTrigger(api),
                        new ProtectMainBranchTrigger(api)
                    },
                    api
                ).Start(cancellation.Token);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToStringWithAllInnerExceptions());
            }
        }
    }
}
```

1.2 ./csharp/Platform.Bot/ProgrammerRole.cs

```csharp
using Interfaces;
using Octokit;
using Storage.Remote.GitHub;
using System;
using System.Collections.Generic;
using System.Threading;

namespace Platform.Bot
{
    /// <summary>
    /// <para>
    /// Represents the programmer role.
    /// </para>
    /// <para></para>
    /// </summary>
    public class ProgrammerRole
    {
        /// <summary>
        /// <para>
        /// The git hub api.
        /// </para>
        /// <para></para>
```

```csharp
23          /// </summary>
24          public readonly GitHubStorage GitHubAPI;
25
26          /// <summary>
27          /// <para>
28          /// The minimum interaction interval.
29          /// </para>
30          /// <para></para>
31          /// </summary>
32          public readonly TimeSpan MinimumInteractionInterval;
33
34          /// <summary>
35          /// <para>
36          /// The triggers.
37          /// </para>
38          /// <para></para>
39          /// </summary>
40          public readonly List<ITrigger<Issue>> Triggers;
41
42          /// <summary>
43          /// <para>
44          /// Initializes a new <see cref="ProgrammerRole"/> instance.
45          /// </para>
46          /// <para></para>
47          /// </summary>
48          /// <param name="triggers">
49          /// <para>A triggers.</para>
50          /// <para></para>
51          /// </param>
52          /// <param name="gitHubAPI">
53          /// <para>A git hub api.</para>
54          /// <para></para>
55          /// </param>
56          public ProgrammerRole(List<ITrigger<Issue>> triggers, GitHubStorage gitHubAPI)
57          {
58              GitHubAPI = gitHubAPI;
59              Triggers = triggers;
60              MinimumInteractionInterval = gitHubAPI.MinimumInteractionInterval;
61          }
62          private void ProcessIssues(CancellationToken token)
63          {
64              while (!token.IsCancellationRequested)
65              {
66                  foreach (var trigger in Triggers)
67                  {
68                      foreach (var issue in GitHubAPI.GetIssues())
69                      {
70                          if (trigger.Condition(issue))
71                          {
72                              trigger.Action(issue);
73                          }
74                      }
75                  }
76                  Thread.Sleep(MinimumInteractionInterval);
77              }
78          }
79
80
81          /// <summary>
82          /// <para>
83          /// Starts the cancellation token.
84          /// </para>
85          /// <para></para>
86          /// </summary>
87          /// <param name="cancellationToken">
88          /// <para>The cancellation token.</para>
89          /// <para></para>
90          /// </param>
91          public void Start(CancellationToken cancellationToken) =>
     ↪      ProcessIssues(cancellationToken);
92      }
93  }
```

1.3 ./csharp/Platform.Bot/Triggers/Activity.cs

```csharp
1   using System.Collections.Generic;
2
3   namespace Platform.Bot
4   {
5       internal class Activity
```

```
 6        {
 7            public string Url { get; set; }
 8
 9            public List<string> Repositories { get; set; }
10        }
11  }
```

1.4 ./csharp/Platform.Bot/Triggers/HelloWorldTrigger.cs

```
 1  using Interfaces;
 2  using Octokit;
 3  using Storage.Local;
 4  using Storage.Remote.GitHub;
 5
 6  namespace Platform.Bot
 7  {
 8      /// <summary>
 9      /// <para>
10      /// Represents the hello world trigger.
11      /// </para>
12      /// <para></para>
13      /// </summary>
14      /// <seealso cref="ITrigger{Issue}"/>
15      internal class HelloWorldTrigger : ITrigger<Issue>
16      {
17          private readonly GitHubStorage gitHubAPI;
18          private readonly FileStorage fileStorage;
19          private readonly string fileSetName;
20
21          /// <summary>
22          /// <para>
23          /// Initializes a new <see cref="HelloWorldTrigger"/> instance.
24          /// </para>
25          /// <para></para>
26          /// </summary>
27          /// <param name="gitHubAPI">
28          /// <para>A git hub api.</para>
29          /// <para></para>
30          /// </param>
31          /// <param name="fileStorage">
32          /// <para>A file storage.</para>
33          /// <para></para>
34          /// </param>
35          /// <param name="fileSetName">
36          /// <para>A file set name.</para>
37          /// <para></para>
38          /// </param>
39          public HelloWorldTrigger(GitHubStorage gitHubAPI, FileStorage fileStorage, string
             ↪  fileSetName)
40          {
41              this.gitHubAPI = gitHubAPI;
42              this.fileStorage = fileStorage;
43              this.fileSetName = fileSetName;
44          }
45
46
47          /// <summary>
48          /// <para>
49          /// Actions the obj.
50          /// </para>
51          /// <para></para>
52          /// </summary>
53          /// <param name="obj">
54          /// <para>The obj.</para>
55          /// <para></para>
56          /// </param>
57
58          public void Action(Issue obj)
59          {
60              foreach (var file in fileStorage.GetFilesFromSet(fileSetName))
61              {
62                  gitHubAPI.CreateOrUpdateFile(obj.Repository.Name, obj.Repository.DefaultBranch,
                     ↪  file);
63              }
64              gitHubAPI.CloseIssue(obj);
65          }
66
67
68          /// <summary>
69          /// <para>
```

```
70    /// Determines whether this instance condition.
71    /// </para>
72    /// <para></para>
73    /// </summary>
74    /// <param name="obj">
75    /// <para>The obj.</para>
76    /// <para></para>
77    /// </param>
78    /// <returns>
79    /// <para>The bool</para>
80    /// <para></para>
81    /// </returns>
82    public bool Condition(Issue obj) => obj.Title.ToLower() == "hello world";
83    }
84 }
```

1.5 ./csharp/Platform.Bot/Triggers/LastCommitActivityTrigger.cs

```
1  using Interfaces;
2  using Octokit;
3  using Platform.Communication.Protocol.Lino;
4  using Storage.Remote.GitHub;
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9
10 namespace Platform.Bot
11 {
12     internal class LastCommitActivityTrigger : ITrigger<Issue>
13     {
14         private readonly GitHubStorage Storage;
15
16         private readonly Parser Parser = new();
17
18         public LastCommitActivityTrigger(GitHubStorage storage) => Storage = storage;
19
20         public bool Condition(Issue issue) => issue.Title.ToLower() == "last 3 months commit
            ↪  activity";
21
22         public void Action(Issue issue)
23         {
24             var issueService = Storage.Client.Issue;
25             var owner = issue.Repository.Owner.Login;
26             var users = GetActivities(GetIgnoredRepositories(Parser.Parse(issue.Body)), owner);
27             StringBuilder sb = new();
28             foreach (var user in users)
29             {
30                 sb.AppendLine($"- **{user.Url.Replace("api.", "").Replace("users/", "")}**");
31                 foreach (var repo in user.Repositories)
32                 {
33                     sb.AppendLine($"  - {repo.Replace("api.", "").Replace("repos/", "")}");
34                 }
35                 // Break line
36                 sb.AppendLine("------------------");
37             }
38             var result = sb.ToString();
39             var comment = issueService.Comment.Create(owner, issue.Repository.Name,
                ↪  issue.Number, result);
40             comment.Wait();
41             Console.WriteLine($"Last commit activity comment is added:
                ↪  {comment.Result.HtmlUrl}");
42             Storage.CloseIssue(issue);
43         }
44
45         public HashSet<string> GetIgnoredRepositories(IList<Link> links)
46         {
47             HashSet<string> ignoredRepos = new() { };
48             foreach (var link in links)
49             {
50                 var values = link.Values;
51                 if (values != null && values.Count == 3 && string.Equals(values.First().Id,
                    ↪  "ignore", StringComparison.OrdinalIgnoreCase) &&
                    ↪  string.Equals(values.Last().Id.Trim('.'), "repository",
                    ↪  StringComparison.OrdinalIgnoreCase))
52                 {
53                     ignoredRepos.Add(values[1].Id);
54                 }
55             }
56             return ignoredRepos;
```

```
57              }
58
59          public HashSet<Activity> GetActivities(HashSet<string> ignoredRepositories, string owner)
60          {
61              HashSet<Activity> activeUsers = new();
62              foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
63              {
64                  if (ignoredRepositories.Contains(repository.Name))
65                  {
66                      continue;
67                  }
68                  foreach (var commit in Storage.GetCommits(repository.Owner.Login,
                    ↪  repository.Name, DateTime.Today.AddMonths(-3)))
69                  {
70                      if (!activeUsers.Any(x => x.Url == commit.Author.Url))
71                      {
72                          activeUsers.Add(new Activity() { Url = commit.Author.Url, Repositories =
                        ↪  new List<string> { repository.Url } });
73                      }
74                      else
75                      {
76                          if (!activeUsers.Any(x => x.Repositories.Any(y => y == repository.Url)
                        ↪  == true))
77                          {
78                              activeUsers.FirstOrDefault(x => x.Url ==
                            ↪  commit.Author.Url).Repositories.Add(repository.Url);
79                          }
80                      }
81                  }
82              }
83              return activeUsers;
84          }
85      }
86  }
```

1.6 ./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs

```
1   using Interfaces;
2   using Octokit;
3   using Platform.Communication.Protocol.Lino;
4   using Storage.Remote.GitHub;
5   using System;
6   using System.Collections.Generic;
7   using System.Linq;
8
9   namespace Platform.Bot
10  {
11      /// <summary>
12      /// <para>
13      /// Represents the organization last month activity trigger.
14      /// </para>
15      /// <para></para>
16      /// </summary>
17      /// <seealso cref="ITrigger{Issue}"/>
18      internal class OrganizationLastMonthActivityTrigger : ITrigger<Issue>
19      {
20          private readonly GitHubStorage Storage;
21          private readonly Parser Parser = new();
22
23          /// <summary>
24          /// <para>
25          /// Initializes a new <see cref="OrganizationLastMonthActivityTrigger"/> instance.
26          /// </para>
27          /// <para></para>
28          /// </summary>
29          /// <param name="storage">
30          /// <para>A storage.</para>
31          /// <para></para>
32          /// </param>
33          public OrganizationLastMonthActivityTrigger(GitHubStorage storage) => Storage = storage;
34
35          /// <summary>
36          /// <para>
37          /// Determines whether this instance condition.
38          /// </para>
39          /// <para></para>
40          /// </summary>
41          /// <param name="issue">
42          /// <para>The issue.</para>
43          /// <para></para>
```

```csharp
        /// </param>
        /// <returns>
        /// <para>The bool</para>
        /// <para></para>
        /// </returns>
        public bool Condition(Issue issue) => issue.Title.ToLower() == "organization last month
        ↪  activity";

        /// <summary>
        /// <para>
        /// Actions the issue.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="issue">
        /// <para>The issue.</para>
        /// <para></para>
        /// </param>
        public void Action(Issue issue)
        {
            var issueService = Storage.Client.Issue;
            var owner = issue.Repository.Owner.Login;
            var activeUsersString = string.Join("\n",
            ↪  GetActiveUsers(GetIgnoredRepositories(Parser.Parse(issue.Body)), owner));
            issueService.Comment.Create(owner, issue.Repository.Name, issue.Number,
            ↪  activeUsersString);
            Storage.CloseIssue(issue);
        }

        /// <summary>
        /// <para>
        /// Gets the ignored repositories using the specified links.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="links">
        /// <para>The links.</para>
        /// <para></para>
        /// </param>
        /// <returns>
        /// <para>The ignored repos.</para>
        /// <para></para>
        /// </returns>
        public HashSet<string> GetIgnoredRepositories(IList<Link> links)
        {
            HashSet<string> ignoredRepos = new() { };
            foreach (var link in links)
            {
                var values = link.Values;
                if (values != null && values.Count == 3 && string.Equals(values.First().Id,
                ↪  "ignore", StringComparison.OrdinalIgnoreCase) &&
                ↪  string.Equals(values.Last().Id.Trim('.'), "repository",
                ↪  StringComparison.OrdinalIgnoreCase))
                {
                    ignoredRepos.Add(values[1].Id);
                }
            }
            return ignoredRepos;
        }

        /// <summary>
        /// <para>
        /// Gets the active users using the specified ignored repositories.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="ignoredRepositories">
        /// <para>The ignored repositories.</para>
        /// <para></para>
        /// </param>
        /// <param name="owner">
        /// <para>The owner.</para>
        /// <para></para>
        /// </param>
        /// <returns>
        /// <para>The active users.</para>
        /// <para></para>
        /// </returns>
```

```csharp
116        public HashSet<string> GetActiveUsers(HashSet<string> ignoredRepositories, string owner)
117        {
118            HashSet<string> activeUsers = new();
119            var date = DateTime.Now.AddMonths(-1);
120            foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
121            {
122                if (ignoredRepositories.Contains(repository.Name))
123                {
124                    continue;
125                }
126                foreach (var commit in Storage.GetCommits(repository.Owner.Login,
     ↪      repository.Name))
127                {

129                    activeUsers.Add(commit.Author.Login);

131                }
132                foreach (var pullRequest in Storage.GetPullRequests(repository.Owner.Login,
     ↪      repository.Name))
133                {
134                    foreach (var reviewer in pullRequest.RequestedReviewers)
135                    {
136                        if (pullRequest.CreatedAt < date || pullRequest.UpdatedAt < date ||
     ↪          pullRequest.ClosedAt < date || pullRequest.MergedAt < date)
137                        {
138                            activeUsers.Add(reviewer.Login);
139                        }
140                    }
141                }
142                foreach (var createdIssue in Storage.GetIssues(repository.Owner.Login,
     ↪      repository.Name))
143                {
144                    activeUsers.Add(createdIssue.User.Login);
145                }
146            }
147            return activeUsers;
148        }
149    }
150 }
```

1.7   ./csharp/Platform.Bot/Triggers/ProtectMainBranchTrigger.cs

```csharp
1  using System;
2  using System.Collections.Generic;
3  using System.Diagnostics;
4  using System.Linq;
5  using System.Text;
6  using Interfaces;
7  using Octokit;
8  using Storage.Remote.GitHub;
9
10 namespace Platform.Bot
11 {
12     public class ProtectMainBranchTrigger : ITrigger<Issue>
13     {
14         private readonly GitHubStorage Storage;
15         public ProtectMainBranchTrigger(GitHubStorage storage) => Storage = storage;
16         public bool Condition(Issue issue) => issue.Title.ToLower() == "protect default branch
     ↪      in all organization's repositories";
17
18         public void Action(Issue issue)
19         {
20             var repositories =
     ↪      Storage.Client.Repository.GetAllForOrg(issue.Repository.Owner.Login).Result;
21             var results = UpdateRepositoriesDefaultBranchProtection(repositories);
22             StringBuilder failedRepositoriesComment = new(repositories.Count *
     ↪      repositories[0].Name.Length);
23             foreach (var result in results.Where(result => !result.Value))
24             {
25                 failedRepositoriesComment.AppendLine($"- [ ] {result.Key}");
26             }
27             if (failedRepositoriesComment.Length != 0)
28             {
29                 failedRepositoriesComment.AppendLine(
30                     "TODO: Fix default branch protection of these repositories. Failed
     ↪          repositories:");
31                 Storage.Client.Issue.Comment.Create(issue.Repository.Id, issue.Number,
     ↪      failedRepositoriesComment.ToString());
32             }
```

```csharp
                else
                {
                    Storage.Client.Issue.Comment.Create(issue.Repository.Id, issue.Number, "Success.
                    ↪   All repositories default branch protection is updated.");
                    Storage.CloseIssue(issue);
                }
        }

        public Dictionary<string, bool>
        ↪   UpdateRepositoriesDefaultBranchProtection(IReadOnlyList<Repository> repositories)
        {
            Dictionary<string, bool> result = new(repositories.Count);
            foreach (var repository in repositories)
            {
                if (repository.Private)
                {
                    continue;
                }
                var update = new BranchProtectionSettingsUpdate(new
                ↪   BranchProtectionPushRestrictionsUpdate());
                var request =
                    Storage.Client.Repository.Branch.UpdateBranchProtection(repository.Id,
                        repository.DefaultBranch, update);
                request.Wait();
                result.Add(repository.Name, request.IsCompletedSuccessfully);
            }
            return result;
        }
    }
}
```

Index