

LinksPlatform's Platform.Bot Class Library

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

```
1  using csharp;
2  using Interfaces;
3  using Octokit;
4  using Platform.Exceptions;
5  using Platform.IO;
6  using Storage.Local;
7  using Storage.Remote.GitHub;
8  using System;
9  using System.Collections.Generic;
10
11 namespace Platform.Bot
12 {
13     /// <summary>
14     /// <para>
15     /// Represents the program.
16     /// </para>
17     /// <para></para>
18     /// </summary>
19     internal class Program
20     {
21         /// <summary>
22         /// <para>
23         /// Main the args.
24         /// </para>
25         /// <para></para>
26         /// </summary>
27         /// <param name="args">
28         /// <para>The args.</para>
29         /// <para></para>
30         /// </param>
31         private static void Main(string[] args)
32         {
33             using var cancellation = new ConsoleCancellation();
34             var argumentIndex = 0;
35             var username = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Username", args);
36             var token = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Token", args);
37             var appName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "App Name", args);
38             var databaseFileName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Database
39                 ↪ file name", args);
40             var fileSetName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "File set name
41                 ↪ ", args);
42             var dbContext = new FileStorage(databaseFileName);
43             Console.WriteLine($"Bot has been started. {Environment.NewLine}Press CTRL+C to
44                 ↪ close");
45             try
46             {
47                 var api = new GitHubStorage(username, token, appName);
48                 new ProgrammerRole(new List<ITrigger<Issue>> { new HelloWorldTrigger(api,
49                     ↪ dbContext, fileSetName), new OrganizationLastMonthActivityTrigger(api) },
50                     ↪ api).Start(cancellation.Token);
51             }
52             catch (Exception ex)
53             {
54                 Console.WriteLine(ex.ToStringWithAllInnerExceptions());
55             }
56         }
57     }
58 }
```

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

```
1  using Interfaces;
2  using Octokit;
3  using Storage.Remote.GitHub;
4  using System;
5  using System.Collections.Generic;
6  using System.Threading;
7
8  namespace Platform.Bot
9  {
10     /// <summary>
11     /// <para>
12     /// Represents the programmer role.
13     /// </para>
14     /// <para></para>
15     /// </summary>
16     public class ProgrammerRole
17     {
```

```

18     /// <summary>
19     /// <para>
20     /// The git hub api.
21     /// </para>
22     /// <para></para>
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
63     /// <summary>
64     /// <para>
65     /// Processes the issues using the specified token.
66     /// </para>
67     /// <para></para>
68     /// </summary>
69     /// <param name="token">
70     /// <para>The token.</para>
71     /// <para></para>
72     /// </param>
73     private void ProcessIssues(Cancellation_token token)
74     {
75         while (!token.IsCancellationRequested)
76         {
77             foreach (var trigger in Triggers)
78             {
79                 foreach (var issue in GitHubAPI.GetIssues())
80                 {
81                     if (trigger.Condition(issue))
82                     {
83                         trigger.Action(issue);
84                     }
85                 }
86                 Thread.Sleep(MinimumInteractionInterval);
87             }
88         }
89     }
90
91     /// <summary>
92     /// <para>
93     /// Starts the cancellation token.
94     /// </para>
95     /// <para></para>
96     /// </summary>

```

```

97     /// <param name="cancellationToken">
98     /// <para>The cancellation token.</para>
99     /// <para></para>
100    /// </param>
101    public void Start(Cancellation token cancellationToken) =>
        ↪ ProcessIssues(cancellationToken);
102    }
103 }

```

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

```

1  using Interfaces;
2  using Octokit;
3  using Storage.Local;
4  using Storage.Remote.GitHub;
5
6  namespace csharp
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         /// <summary>
18         /// <para>
19         /// The git hub api.
20         /// </para>
21         /// <para></para>
22         /// </summary>
23         private readonly GitHubStorage gitHubAPI;
24
25         /// <summary>
26         /// <para>
27         /// The file storage.
28         /// </para>
29         /// <para></para>
30         /// </summary>
31         private readonly FileStorage fileStorage;
32
33         /// <summary>
34         /// <para>
35         /// The file set name.
36         /// </para>
37         /// <para></para>
38         /// </summary>
39         private readonly string fileSetName;
40
41         /// <summary>
42         /// <para>
43         /// Initializes a new <see cref="HelloWorldTrigger" /> instance.
44         /// </para>
45         /// <para></para>
46         /// </summary>
47         /// <param name="gitHubAPI">
48         /// <para>A git hub api.</para>
49         /// <para></para>
50         /// </param>
51         /// <param name="fileStorage">
52         /// <para>A file storage.</para>
53         /// <para></para>
54         /// </param>
55         /// <param name="fileSetName">
56         /// <para>A file set name.</para>
57         /// <para></para>
58         /// </param>
59         public HelloWorldTrigger(GitHubStorage gitHubAPI, FileStorage fileStorage, string
            ↪ fileSetName)
60         {
61             this.gitHubAPI = gitHubAPI;
62             this.fileStorage = fileStorage;
63             this.fileSetName = fileSetName;
64         }
65
66         /// <summary>
67         /// <para>
68         /// Actions the obj.

```

```

69     /// </para>
70     /// <para></para>
71     /// </summary>
72     /// <param name="obj">
73     /// <para>The obj.</para>
74     /// <para></para>
75     /// </param>
76     public void Action(Issue obj)
77     {
78         foreach (var file in fileStorage.GetFilesFromSet(fileSetName))
79         {
80             gitHubAPI.CreateOrUpdateFile(obj.Repository.Name, obj.Repository.DefaultBranch,
81                 ↪ file);
82         }
83         gitHubAPI.CloseIssue(obj);
84     }
85     /// <summary>
86     /// <para>
87     /// Determines whether this instance condition.
88     /// </para>
89     /// <para></para>
90     /// </summary>
91     /// <param name="obj">
92     /// <para>The obj.</para>
93     /// <para></para>
94     /// </param>
95     /// <returns>
96     /// <para>The bool</para>
97     /// <para></para>
98     /// </returns>
99     public bool Condition(Issue obj) => obj.Title.ToLower() == "hello world";
100 }
101 }

```

1.4 ./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         /// <summary>
21         /// <para>
22         /// The storage.
23         /// </para>
24         /// <para></para>
25         /// </summary>
26         private readonly GitHubStorage Storage;
27
28         /// <summary>
29         /// <para>
30         /// The parser.
31         /// </para>
32         /// <para></para>
33         /// </summary>
34         private readonly Parser Parser = new();
35
36         /// <summary>
37         /// <para>
38         /// Initializes a new <see cref="OrganizationLastMonthActivityTrigger"/> instance.
39         /// </para>
40         /// <para></para>
41         /// </summary>
42         /// <param name="storage">
43         /// <para>A storage.</para>

```

```

44     /// <para></para>
45     /// </param>
46     public OrganizationLastMonthActivityTrigger(GitHubStorage storage) => Storage = storage;
47
48     /// <summary>
49     /// <para>
50     /// Determines whether this instance condition.
51     /// </para>
52     /// <para></para>
53     /// </summary>
54     /// <param name="issue">
55     /// <para>The issue.</para>
56     /// <para></para>
57     /// </param>
58     /// <returns>
59     /// <para>The bool</para>
60     /// <para></para>
61     /// </returns>
62     public bool Condition(Issue issue) => issue.Title.ToLower() == "organization last month
    ↳ activity";
63
64     /// <summary>
65     /// <para>
66     /// Actions the issue.
67     /// </para>
68     /// <para></para>
69     /// </summary>
70     /// <param name="issue">
71     /// <para>The issue.</para>
72     /// <para></para>
73     /// </param>
74     public void Action(Issue issue)
75     {
76         var issueService = Storage.Client.Issue;
77         var owner = issue.Repository.Owner.Login;
78         var activeUsersString = string.Join("\n",
79             ↳ GetActiveUsers(GetIgnoredRepositories(Parser.Parse(issue.Body)), owner));
80         issueService.Comment.Create(owner, issue.Repository.Name, issue.Number,
81             ↳ activeUsersString);
82         Storage.CloseIssue(issue);
83     }
84
85     /// <summary>
86     /// <para>
87     /// Gets the ignored repositories using the specified links.
88     /// </para>
89     /// <para></para>
90     /// </summary>
91     /// <param name="links">
92     /// <para>The links.</para>
93     /// <para></para>
94     /// </param>
95     /// <returns>
96     /// <para>The ignored repos.</para>
97     /// <para></para>
98     /// </returns>
99     public HashSet<string> GetIgnoredRepositories(ICollection<Link> links)
100     {
101         HashSet<string> ignoredRepos = new() { };
102         foreach (var link in links)
103         {
104             var values = link.Values;
105             if (values != null && values.Count == 3 && string.Equals(values.First().Id,
106                 ↳ "ignore", StringComparison.OrdinalIgnoreCase) &&
107                 ↳ string.Equals(values.Last().Id.Trim('.'), "repository",
108                     ↳ StringComparison.OrdinalIgnoreCase))
109             {
110                 ignoredRepos.Add(values[1].Id);
111             }
112         }
113         return ignoredRepos;
114     }
115
116     /// <summary>
117     /// <para>
118     /// Gets the active users using the specified ignored repositories.
119     /// </para>
120     /// <para></para>
121     /// </summary>
122     /// <param name="ignoredRepos">
123     /// <para>The ignored repositories.</para>
124     /// <para></para>
125     /// </param>
126     /// <returns>
127     /// <para>The active users.</para>
128     /// <para></para>
129     /// </returns>
130     public string GetActiveUsers(HashSet<string> ignoredRepos)
131     {
132         var activeUsers = new List<string>();
133         foreach (var user in Storage.Users)
134         {
135             if (!ignoredRepos.Contains(user.Login))
136             {
137                 activeUsers.Add(user.Login);
138             }
139         }
140         return string.Join("\n", activeUsers);
141     }
142
143     /// <summary>
144     /// <para>
145     /// Gets the ignored repositories using the specified links.
146     /// </para>
147     /// <para></para>
148     /// </summary>
149     /// <param name="links">
150     /// <para>The links.</para>
151     /// <para></para>
152     /// </param>
153     /// <returns>
154     /// <para>The ignored repos.</para>
155     /// <para></para>
156     /// </returns>
157     public HashSet<string> GetIgnoredRepositories(ICollection<Link> links)
158     {
159         HashSet<string> ignoredRepos = new() { };
160         foreach (var link in links)
161         {
162             var values = link.Values;
163             if (values != null && values.Count == 3 && string.Equals(values.First().Id,
164                 ↳ "ignore", StringComparison.OrdinalIgnoreCase) &&
165                 ↳ string.Equals(values.Last().Id.Trim('.'), "repository",
166                     ↳ StringComparison.OrdinalIgnoreCase))
167             {
168                 ignoredRepos.Add(values[1].Id);
169             }
170         }
171         return ignoredRepos;
172     }
173
174     /// <summary>
175     /// <para>
176     /// Gets the active users using the specified ignored repositories.
177     /// </para>
178     /// <para></para>
179     /// </summary>
180     /// <param name="ignoredRepos">
181     /// <para>The ignored repositories.</para>
182     /// <para></para>
183     /// </param>
184     /// <returns>
185     /// <para>The active users.</para>
186     /// <para></para>
187     /// </returns>
188     public string GetActiveUsers(HashSet<string> ignoredRepos)
189     {
190         var activeUsers = new List<string>();
191         foreach (var user in Storage.Users)
192         {
193             if (!ignoredRepos.Contains(user.Login))
194             {
195                 activeUsers.Add(user.Login);
196             }
197         }
198         return string.Join("\n", activeUsers);
199     }
200
201     /// <summary>
202     /// <para>
203     /// Gets the ignored repositories using the specified links.
204     /// </para>
205     /// <para></para>
206     /// </summary>
207     /// <param name="links">
208     /// <para>The links.</para>
209     /// <para></para>
210     /// </param>
211     /// <returns>
212     /// <para>The ignored repos.</para>
213     /// <para></para>
214     /// </returns>
215     public HashSet<string> GetIgnoredRepositories(ICollection<Link> links)
216     {
217         HashSet<string> ignoredRepos = new() { };
218         foreach (var link in links)
219         {
220             var values = link.Values;
221             if (values != null && values.Count == 3 && string.Equals(values.First().Id,
222                 ↳ "ignore", StringComparison.OrdinalIgnoreCase) &&
223                 ↳ string.Equals(values.Last().Id.Trim('.'), "repository",
224                     ↳ StringComparison.OrdinalIgnoreCase))
225             {
226                 ignoredRepos.Add(values[1].Id);
227             }
228         }
229         return ignoredRepos;
230     }
231
232     /// <summary>
233     /// <para>
234     /// Gets the active users using the specified ignored repositories.
235     /// </para>
236     /// <para></para>
237     /// </summary>
238     /// <param name="ignoredRepos">
239     /// <para>The ignored repositories.</para>
240     /// <para></para>
241     /// </param>
242     /// <returns>
243     /// <para>The active users.</para>
244     /// <para></para>
245     /// </returns>
246     public string GetActiveUsers(HashSet<string> ignoredRepos)
247     {
248         var activeUsers = new List<string>();
249         foreach (var user in Storage.Users)
250         {
251             if (!ignoredRepos.Contains(user.Login))
252             {
253                 activeUsers.Add(user.Login);
254             }
255         }
256         return string.Join("\n", activeUsers);
257     }

```

```

116    /// </summary>
117    /// <param name="ignoredRepositories">
118    /// <para>The ignored repositories.</para>
119    /// <para></para>
120    /// </param>
121    /// <param name="owner">
122    /// <para>The owner.</para>
123    /// <para></para>
124    /// </param>
125    /// <returns>
126    /// <para>The active users.</para>
127    /// <para></para>
128    /// </returns>
129    public HashSet<string> GetActiveUsers(HashSet<string> ignoredRepositories, string owner)
130    {
131        HashSet<string> activeUsers = new();
132        foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
133        {
134            if (ignoredRepositories.Contains(repository.Name))
135            {
136                continue;
137            }
138            foreach (var commit in Storage.GetCommits(repository.Owner.Login,
139                ↪ repository.Name))
140            {
141                activeUsers.Add(commit.Author.Login);
142            }
143            foreach (var pullRequest in Storage.GetPullRequests(repository.Owner.Login,
144                ↪ repository.Name))
145            {
146                foreach (var reviewer in pullRequest.RequestedReviewers)
147                {
148                    activeUsers.Add(reviewer.Login);
149                }
150            }
151            foreach (var createdIssue in Storage.GetIssues(repository.Owner.Login,
152                ↪ repository.Name))
153            {
154                activeUsers.Add(createdIssue.User.Login);
155            }
156        }
157        return activeUsers;
158    }
159 }

```

Index

- ./csharp/Platform.Bot/Program.cs, 1
- ./csharp/Platform.Bot/ProgrammerRole.cs, 1
- ./csharp/Platform.Bot/Triggers/HelloWorldTrigger.cs, 3
- ./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs, 4