

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(
49                     new List<ITrigger<Issue>> {
50                         new HelloWorldTrigger(api, dbContext, fileSetName),
51                         new OrganizationLastMonthActivityTrigger(api),
52                         new LastCommitActivityTrigger(api)
53                     },
54                     api
55                 ).Start(cancellation.Token);
56             }
57             catch (Exception ex)
58             {
59                 Console.WriteLine(ex.ToStringWithAllInnerExceptions());
60             }
61         }
62     }
63 }
```

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              }
87              Thread.Sleep(MinimumInteractionInterval);
88          }
89      }
90
91

```

```

92     /// <summary>
93     /// <para>
94     /// Starts the cancellation token.
95     /// </para>
96     /// <para></para>
97     /// </summary>
98     /// <param name="cancellationToken">
99     /// <para>The cancellation token.</para>
100    /// <para></para>
101    /// </param>
102    public void Start(Cancellation token cancellationToken) =>
        ↪ ProcessIssues(cancellationToken);
103    }
104 }

```

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

```

1 using System.Collections.Generic;
2
3 namespace 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 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>
54     /// </param>
55     /// <param name="fileSetName">
56     /// <para>A file set name.</para>
57     /// </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
67     /// <summary>
68     /// <para>
69     /// Actions the obj.
70     /// </para>
71     /// </para>
72     /// </summary>
73     /// <param name="obj">
74     /// <para>The obj.</para>
75     /// </para>
76     /// </param>
77
78     public void Action(Issue obj)
79     {
80         foreach (var file in fileStorage.GetFilesFromSet(fileSetName))
81         {
82             gitHubAPI.CreateOrUpdateFile(obj.Repository.Name, obj.Repository.DefaultBranch,
                ↳ file);
83         }
84         gitHubAPI.CloseIssue(obj);
85     }
86
87
88     /// <summary>
89     /// <para>
90     /// Determines whether this instance condition.
91     /// </para>
92     /// </para>
93     /// </summary>
94     /// <param name="obj">
95     /// <para>The obj.</para>
96     /// </para>
97     /// </param>
98     /// <returns>
99     /// <para>The bool</para>
100    /// </para>
101    /// </returns>
102    public bool Condition(Issue obj) => obj.Title.ToLower() == "hello world";
103 }
104 }

```

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 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     sb.Append("`");
29     foreach (var user in users)
30     {
31         sb.AppendLine($"{Environment.NewLine}" + user.Url.Replace("api.",
32             ↳ "").Replace("users/", ""));
33         foreach (var repo in user.Repositories)
34         {
35             sb.AppendLine(repo.Replace("api.", "").Replace("repos/", ""));
36         }
37     }
38     Console.WriteLine(sb.Append("`").ToString());
39     issueService.Comment.Create(owner, issue.Repository.Name, issue.Number,
40     ↳ sb.Append("`").ToString());
41     Storage.CloseIssue(issue);
42 }
43
44 public HashSet<string> GetIgnoredRepositories(IList<Link> links)
45 {
46     HashSet<string> ignoredRepos = new() { };
47     foreach (var link in links)
48     {
49         var values = link.Values;
50         if (values != null && values.Count == 3 && string.Equals(values.First().Id,
51             ↳ "ignore", StringComparison.OrdinalIgnoreCase) &&
52             ↳ string.Equals(values.Last().Id.Trim('.'), "repository",
53             ↳ StringComparison.OrdinalIgnoreCase))
54         {
55             ignoredRepos.Add(values[1].Id);
56         }
57     }
58     return ignoredRepos;
59 }
60
61 public HashSet<Activity> GetActivities(HashSet<string> ignoredRepositories, string owner)
62 {
63     HashSet<Activity> activeUsers = new();
64     foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
65     {
66         if (ignoredRepositories.Contains(repository.Name))
67         {
68             continue;
69         }
70         foreach (var commit in Storage.GetCommits(repository.Owner.Login,
71             ↳ repository.Name, DateTime.Today.AddMonths(-3)))
72         {
73             if (!activeUsers.Any(x => x.Url == commit.Author.Url))
74             {
75                 activeUsers.Add(new Activity() { Url = commit.Author.Url, Repositories =
76                     ↳ new List<string> { repository.Url } });
77             }
78             else
79             {
80                 if (!activeUsers.Any(x => x.Repositories.Any(y => y == repository.Url)
81                     ↳ == true))
82                 {
83                     activeUsers.FirstOrDefault(x => x.Url ==
84                     ↳ commit.Author.Url).Repositories.Add(repository.Url);
85                 }
86             }
87         }
88     }
89     return activeUsers;
90 }
91 }
92 }
93 }

```

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         /// <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 }

```

```

81     }
82
83     /// <summary>
84     /// <para>
85     /// Gets the ignored repositories using the specified links.
86     /// </para>
87     /// <para></para>
88     /// </summary>
89     /// <param name="links">
90     /// <para>The links.</para>
91     /// <para></para>
92     /// </param>
93     /// <returns>
94     /// <para>The ignored repos.</para>
95     /// <para></para>
96     /// </returns>
97     public HashSet<string> GetIgnoredRepositories(IList<Link> links)
98     {
99         HashSet<string> ignoredRepos = new() { };
100         foreach (var link in links)
101         {
102             var values = link.Values;
103             if (values != null && values.Count == 3 && string.Equals(values.First().Id,
104                 ↪ "ignore", StringComparison.OrdinalIgnoreCase) &&
105                 ↪ string.Equals(values.Last().Id.Trim('.'), "repository",
106                 ↪ StringComparison.OrdinalIgnoreCase))
107             {
108                 ignoredRepos.Add(values[1].Id);
109             }
110         }
111         return ignoredRepos;
112     }
113
114     /// <summary>
115     /// <para>
116     /// Gets the active users using the specified ignored repositories.
117     /// </para>
118     /// <para></para>
119     /// </summary>
120     /// <param name="ignoredRepositories">
121     /// <para>The ignored repositories.</para>
122     /// <para></para>
123     /// </param>
124     /// <param name="owner">
125     /// <para>The owner.</para>
126     /// <para></para>
127     /// </param>
128     /// <returns>
129     /// <para>The active users.</para>
130     /// <para></para>
131     /// </returns>
132     public HashSet<string> GetActiveUsers(HashSet<string> ignoredRepositories, string owner)
133     {
134         HashSet<string> activeUsers = new();
135         foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
136         {
137             if (ignoredRepositories.Contains(repository.Name))
138             {
139                 continue;
140             }
141             foreach (var commit in Storage.GetCommits(repository.Owner.Login,
142                 ↪ repository.Name, date))
143             {
144                 activeUsers.Add(commit.Author.Login);
145             }
146             foreach (var pullRequest in Storage.GetPullRequests(repository.Owner.Login,
147                 ↪ repository.Name))
148             {
149                 foreach (var reviewer in pullRequest.RequestedReviewers)
150                 {
151                     if (pullRequest.CreatedAt < date || pullRequest.UpdatedAt < date ||
152                         ↪ pullRequest.ClosedAt < date || pullRequest.MergedAt < date)
153                     {
154                         activeUsers.Add(reviewer.Login);
155                     }
156                 }
157             }
158         }
159     }

```

```
153     }
154     foreach (var createdIssue in Storage.GetIssues(repository.Owner.Login,
155     ↪ repository.Name))
156     {
157         activeUsers.Add(createdIssue.User.Login);
158     }
159     return activeUsers;
160 }
161 }
162 }
```


Index

- ./csharp/Platform.Bot/Program.cs, 1
- ./csharp/Platform.Bot/ProgrammerRole.cs, 1
- ./csharp/Platform.Bot/Triggers/Activity.cs, 3
- ./csharp/Platform.Bot/Triggers/HelloWorldTrigger.cs, 3
- ./csharp/Platform.Bot/Triggers/LastCommitActivityTrigger.cs, 4
- ./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs, 5