

LinksPlatform's Platform.Bot Class Library

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

```
1 using Interfaces;
2 using Octokit;
3 using Platform.Exceptions;
4 using Platform.IO;
5 using Storage.Local;
6 using Storage.Remote.GitHub;
7 using System;
8 using System.Collections.Generic;
9
10 namespace Platform.Bot
11 {
12     /// <summary>
13     /// <para>
14     /// Represents the program.
15     /// </para>
16     /// <para></para>
17     /// </summary>
18     internal class Program
19     {
20         /// <summary>
21         /// <para>
22         /// Main the args.
23         /// </para>
24         /// <para></para>
25         /// </summary>
26         /// <param name="args">
27         /// <para>The args.</para>
28         /// <para></para>
29         /// </param>
30         private static void Main(string[] args)
31         {
32             using var cancellation = new ConsoleCancellation();
33             var argumentIndex = 0;
34             var username = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Username", args);
35             var token = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Token", args);
36             var appName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "App Name", args);
37             var databaseFileName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Database
38                 ↪ file name", args);
39             var fileSetName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "File set name
40                 ↪ ", args);
41             var dbContext = new FileStorage(databaseFileName);
42             Console.WriteLine($"Bot has been started. {Environment.NewLine}Press CTRL+C to
43                 ↪ close");
44             try
45             {
46                 var api = new GitHubStorage(username, token, appName);
47                 new ProgrammerRole(
48                     new List<ITrigger<Issue>> {
49                         new HelloWorldTrigger(api, dbContext, fileSetName),
50                         new OrganizationLastMonthActivityTrigger(api),
51                         new LastCommitActivityTrigger(api),
52                         new ProtectMainBranchTrigger(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 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        /// <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     /// </summary>
72     /// <param name="obj">
73     /// <para>The obj.</para>
74     /// </para>
75     /// </param>
76
77     public void Action(Issue obj)
78     {
79         foreach (var file in fileStorage.GetFilesFromSet(fileSetName))
80         {
81             gitHubAPI.CreateOrUpdateFile(obj.Repository.Name, obj.Repository.DefaultBranch,
82                 ↳ file);
83         }
84         gitHubAPI.CloseIssue(obj);
85     }
86
87
88     /// <summary>
89     /// <para>
90     /// Determines whether this instance condition.
91     /// </para>
92     /// </summary>
93     /// <param name="obj">
94     /// <para>The obj.</para>
95     /// </para>
96     /// </param>
97     /// <returns>
98     /// <para>The bool</para>
99     /// </returns>
100
101     public bool Condition(Issue obj) => obj.Title.ToLower() == "hello world";
102 }
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 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     }

```

```

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,
40     ↪ issue.Number, result);
41     comment.Wait();
42     Console.WriteLine($"Last commit activity comment is added:
43     ↪ {comment.Result.HtmlUrl}");
44     Storage.CloseIssue(issue);
45 }
46
47 public HashSet<string> GetIgnoredRepositories(IList<Link> links)
48 {
49     HashSet<string> ignoredRepos = new() { };
50     foreach (var link in links)
51     {
52         var values = link.Values;
53         if (values != null && values.Count == 3 && string.Equals(values.First().Id,
54         ↪ "ignore", StringComparison.OrdinalIgnoreCase) &&
55         ↪ string.Equals(values.Last().Id.Trim('.'), "repository",
56         ↪ StringComparison.OrdinalIgnoreCase))
57         {
58             ignoredRepos.Add(values[1].Id);
59         }
60     }
61     return ignoredRepos;
62 }
63
64 public HashSet<Activity> GetActivities(HashSet<string> ignoredRepositories, string owner)
65 {
66     HashSet<Activity> activeUsers = new();
67     foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
68     {
69         if (ignoredRepositories.Contains(repository.Name))
70         {
71             continue;
72         }
73         foreach (var commit in Storage.GetCommits(repository.Owner.Login,
74         ↪ repository.Name, DateTime.Today.AddMonths(-3)))
75         {
76             if (!activeUsers.Any(x => x.Url == commit.Author.Url))
77             {
78                 activeUsers.Add(new Activity() { Url = commit.Author.Url, Repositories =
79                 ↪ new List<string> { repository.Url } });
80             }
81             else
82             {
83                 if (!activeUsers.Any(x => x.Repositories.Any(y => y == repository.Url)
84                 ↪ == true))
85                 {
86                     activeUsers.FirstOrDefault(x => x.Url ==
87                     ↪ commit.Author.Url).Repositories.Add(repository.Url);
88                 }
89             }
90         }
91     }
92     return activeUsers;
93 }
94 }
95 }
96 }

```

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",
        ↳ GetActiveUsers(GetIgnoredRepositories(Parser.Parse(issue.Body)), owner));

```

```

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

```

```

150         {
151             activeUsers.Add(reviewer.Login);
152         }
153     }
154 }
155 foreach (var createdIssue in Storage.GetIssues(repository.Owner.Login,
156 ↪ repository.Name))
157 {
158     activeUsers.Add(createdIssue.User.Login);
159 }
160 return activeUsers;
161 }
162 }
163 }

```

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

```

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
17 ↪ in all organization's repositories";
18
19         public void Action(Issue issue)
20         {
21             var repositories =
22 ↪ Storage.Client.Repository.GetAllForOrg(issue.Repository.Owner.Login).Result;
23             var results = UpdateRepositoriesDefaultBranchProtection(repositories);
24             StringBuilder failedRepositoriesComment = new(repositories.Count *
25 ↪ repositories[0].Name.Length);
26             foreach (var result in results.Where(result => !result.Value))
27             {
28                 failedRepositoriesComment.AppendLine($"{result.Key}");
29             }
30             if (failedRepositoriesComment.Length != 0)
31             {
32                 failedRepositoriesComment.AppendLine(
33 ↪ "TODO: Fix default branch protection of these repositories. Failed
34 ↪ repositories:");
35                 Storage.Client.Issue.Comment.Create(issue.Repository.Id, issue.Number,
36 ↪ failedRepositoriesComment.ToString());
37             }
38             else
39             {
40                 Storage.Client.Issue.Comment.Create(issue.Repository.Id, issue.Number, "Success.
41 ↪ All repositories default branch protection is updated.");
42                 Storage.CloseIssue(issue);
43             }
44         }
45
46         public Dictionary<string, bool>
47 ↪ UpdateRepositoriesDefaultBranchProtection(IReadOnlyList<Repository> repositories)
48         {
49             Dictionary<string, bool> result = new(repositories.Count);
50             foreach (var repository in repositories)
51             {
52                 if (repository.Private)
53                 {
54                     continue;
55                 }
56                 var update = new BranchProtectionSettingsUpdate(new
57 ↪ BranchProtectionPushRestrictionsUpdate());
58                 var request =
59 ↪ Storage.Client.Repository.Branch.UpdateBranchProtection(repository.Id,
60 ↪ repository.DefaultBranch, update);
61                 request.Wait();
62                 result.Add(repository.Name, request.IsCompletedSuccessfully);
63             }
64         }
65     }
66 }

```



```
55     }
56     return result;
57 }
58 }
59 }
```

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
- ./csharp/Platform.Bot/Triggers/ProtectMainBranchTrigger.cs, 8