

## 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 using System.Threading;
10 using Platform.Bot.Trackers;
11 using Platform.Bot.Triggers;
12
13 namespace Platform.Bot
14 {
15     /// <summary>
16     /// <para>
17     /// Represents the program.
18     /// </para>
19     /// <para></para>
20     /// </summary>
21     internal class Program
22     {
23         private static void Main(string[] args)
24         {
25             using var cancellation = new ConsoleCancellation();
26             var argumentIndex = 0;
27             var username = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Username", args);
28             var token = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Token", args);
29             var appName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "App Name", args);
30             var databaseFileName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Database
31                 ↪ file name", args);
32             var fileSetName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "File set name",
33                 ↪ args);
34             var minimumInteractionIntervalStringInputInSeconds =
35                 ↪ ConsoleHelpers.GetOrReadArgument(argumentIndex, "Minimum interaction interval in
36                 ↪ seconds", args);
37             var minimumInteractionInterval = TimeSpan.FromSeconds(Int32.Parse(minimumInteraction
38                 ↪ IntervalStringInputInSeconds));
39             var dbContext = new FileStorage(databaseFileName);
40             Console.WriteLine($"Bot has been started. {Environment.NewLine}Press CTRL+C to
41                 ↪ close");
42             try
43             {
44                 while (true)
45                 {
46                     var api = new GitHubStorage(username, token, appName);
47                     new IssueTracker(new List<ITrigger<Issue>>
48                     {
49                         new HelloWorldTrigger(api, dbContext, fileSetName),
50                         new OrganizationLastMonthActivityTrigger(api),
51                         new LastCommitActivityTrigger(api),
52                         new ProtectMainBranchTrigger(api),
53                     }, api).Start(cancellation.Token);
54                     new PullRequestTracker(new List<ITrigger<PullRequest>> { new
55                         ↪ MergeDependabotBumpsTrigger(api) }, api).Start(cancellation.Token);
56                     Thread.Sleep(minimumInteractionInterval);
57                 }
58             }
59             catch (Exception ex)
60             {
61                 Console.WriteLine(ex.ToStringWithAllInnerExceptions());
62             }
63         }
64     }
65 }
```

### 1.2 ./csharp/Platform.Bot/Trackers/IssueTracker.cs

```
1 using Interfaces;
2 using Octokit;
3 using Storage.Remote.GitHub;
4 using System.Collections.Generic;
5 using System.Threading;
6
7 namespace Platform.Bot.Trackers
8 {
9     /// <summary>
10     /// <para>
```

```

11  /// Represents the programmer role.
12  /// </para>
13  /// <para></para>
14  /// </summary>
15  public class IssueTracker : ITracker<Issue>
16  {
17      /// <summary>
18      /// <para>
19      /// The git hub api.
20      /// </para>
21      /// <para></para>
22      /// </summary>
23      public GitHubStorage Storage { get; }
24
25      /// <summary>
26      /// <para>
27      /// The triggers.
28      /// </para>
29      /// <para></para>
30      /// </summary>
31      public List<ITrigger<Issue>> Triggers { get; }
32
33      /// <summary>
34      /// <para>
35      /// Initializes a new <see cref="IssueTracker"/> instance.
36      /// </para>
37      /// <para></para>
38      /// </summary>
39      /// <param name="triggers">
40      /// <para>A triggers.</para>
41      /// <para></para>
42      /// </param>
43      /// <param name="gitHubApi">
44      /// <para>A git hub api.</para>
45      /// <para></para>
46      /// </param>
47      public IssueTracker(List<ITrigger<Issue>> triggers, GitHubStorage gitHubApi)
48      {
49          Storage = gitHubApi;
50          Triggers = triggers;
51      }
52
53      /// <summary>
54      /// <para>
55      /// Starts the cancellation token.
56      /// </para>
57      /// <para></para>
58      /// </summary>
59      /// <param name="cancellationToken">
60      /// <para>The cancellation token.</para>
61      /// <para></para>
62      /// </param>
63      public void Start(CancellationToken cancellationToken)
64      {
65          foreach (var trigger in Triggers)
66          {
67              foreach (var issue in Storage.GetIssues())
68              {
69                  if (cancellationToken.IsCancellationRequested)
70                  {
71                      return;
72                  }
73                  if (trigger.Condition(issue))
74                  {
75                      trigger.Action(issue);
76                  }
77              }
78          }
79      }
80  }
81 }

```

### 1.3 ./csharp/Platform.Bot/Trackers/PullRequestTracker.cs

```

1  using Interfaces;
2  using Octokit;
3  using Storage.Remote.GitHub;
4  using System.Collections.Generic;
5  using System.Threading;
6  using Platform.Threading;

```

```

7
8 namespace Platform.Bot.Trackers
9 {
10     using TContext = PullRequest;
11     /// <summary>
12     /// <para>
13     /// Represents the programmer role.
14     /// </para>
15     /// <para></para>
16     /// </summary>
17     public class PullRequestTracker : ITracker<TContext>
18     {
19         /// <summary>
20         /// <para>
21         /// The git hub api.
22         /// </para>
23         /// <para></para>
24         /// </summary>
25         public GitHubStorage Storage { get; }
26
27         /// <summary>
28         /// <para>
29         /// The triggers.
30         /// </para>
31         /// <para></para>
32         /// </summary>
33         public List<ITrigger<TContext>> Triggers { get; }
34
35         /// <summary>
36         /// <para>
37         /// Initializes a new <see cref="IssueTracker"/> instance.
38         /// </para>
39         /// <para></para>
40         /// </summary>
41         /// <param name="triggers">
42         /// <para>A triggers.</para>
43         /// <para></para>
44         /// </param>
45         /// <param name="storage">
46         /// <para>A git hub api.</para>
47         /// <para></para>
48         /// </param>
49         public PullRequestTracker(List<ITrigger<TContext>> triggers, GitHubStorage storage)
50         {
51             Storage = storage;
52             Triggers = triggers;
53         }
54
55         /// <summary>
56         /// <para>
57         /// Starts the cancellation token.
58         /// </para>
59         /// <para></para>
60         /// </summary>
61         /// <param name="cancellationToken">
62         /// <para>The cancellation token.</para>
63         /// <para></para>
64         /// </param>
65         public void Start(Cancellation token)
66         {
67             foreach (var trigger in Triggers)
68             {
69                 foreach (var repository in
70                     ↪ Storage.Client.Repository.GetAllForOrg("linksplatform").AwaitResult())
71                 {
72                     foreach (var pullRequest in Storage.Client.PullRequest.GetAllForRepository(r
73                     ↪ epository.Id).AwaitResult())
74                     {
75                         if (cancellationToken.IsCancellationRequested)
76                         {
77                             return;
78                         }
79                         if (trigger.Condition(pullRequest))
80                         {
81                             trigger.Action(pullRequest);
82                         }
83                     }
84                 }
85             }
86         }
87     }
88 }

```

```

83     }
84 }
85 }
86 }

```

#### 1.4 ./csharp/Platform.Bot/Triggers/Activity.cs

```

1 using System.Collections.Generic;
2
3 namespace Platform.Bot.Triggers
4 {
5     internal class Activity
6     {
7         public string Url { get; set; }
8
9         public List<string> Repositories { get; set; }
10    }
11 }

```

#### 1.5 ./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.Triggers
7 {
8     using TContext = Issue;
9     /// <summary>
10    /// <para>
11    /// Represents the hello world trigger.
12    /// </para>
13    /// <para></para>
14    /// </summary>
15    /// <seealso cref="ITrigger{TContext}"/>
16    internal class HelloWorldTrigger : ITrigger<TContext>
17    {
18        private readonly GitHubStorage _storage;
19        private readonly FileStorage _fileStorage;
20        private readonly string _fileSetName;
21
22        /// <summary>
23        /// <para>
24        /// Initializes a new <see cref="HelloWorldTrigger"/> instance.
25        /// </para>
26        /// <para></para>
27        /// </summary>
28        /// <param name="storage">
29        /// <para>A git hub api.</para>
30        /// <para></para>
31        /// </param>
32        /// <param name="fileStorage">
33        /// <para>A file storage.</para>
34        /// <para></para>
35        /// </param>
36        /// <param name="fileSetName">
37        /// <para>A file set name.</para>
38        /// <para></para>
39        /// </param>
40        public HelloWorldTrigger(GitHubStorage storage, FileStorage fileStorage, string
41        ↪ fileSetName)
42        {
43            this._storage = storage;
44            this._fileStorage = fileStorage;
45            this._fileSetName = fileSetName;
46        }
47
48        /// <summary>
49        /// <para>
50        /// Actions the context.
51        /// </para>
52        /// <para></para>
53        /// </summary>
54        /// <param name="context">
55        /// <para>The context.</para>
56        /// <para></para>
57        /// </param>
58
59        public void Action(TContext context)

```

```

60     {
61         foreach (var file in _fileStorage.GetFilesFromSet(_fileSetName))
62         {
63             _storage.CreateOrUpdateFile(context.Repository.Name,
64                 ↪ context.Repository.DefaultBranch, file);
65         }
66         _storage.CloseIssue(context);
67     }
68
69     /// <summary>
70     /// <para>
71     /// Determines whether this instance condition.
72     /// </para>
73     /// <para></para>
74     /// </summary>
75     /// <param name="context">
76     /// <para>The context.</para>
77     /// <para></para>
78     /// </param>
79     /// <returns>
80     /// <para>The bool</para>
81     /// <para></para>
82     /// </returns>
83     public bool Condition(TContext context) => context.Title.ToLower() == "hello world";
84 }
85 }

```

## 1.6 ./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.Triggers
11 {
12     using TContext = Issue;
13     internal class LastCommitActivityTrigger : ITrigger<TContext>
14     {
15         private readonly GitHubStorage _storage;
16
17         private readonly Parser _parser = new();
18
19         public LastCommitActivityTrigger(GitHubStorage storage) => _storage = storage;
20
21         public bool Condition(TContext context) => context.Title.ToLower() == "last 3 months
22             ↪ commit activity";
23
24         public void Action(TContext context)
25         {
26             var issueService = _storage.Client.Issue;
27             var owner = context.Repository.Owner.Login;
28             var ignoredRepositories =
29                 context.Body != null ? GetIgnoredRepositories(_parser.Parse(context.Body)) :
30                 ↪ default;
31             var users = GetActivities(owner, ignoredRepositories);
32             StringBuilder sb = new();
33             foreach (var user in users)
34             {
35                 sb.AppendLine($"- **{user.Url.Replace("api.", "").Replace("users/", "")}**");
36                 // Break line
37                 sb.AppendLine("-----");
38             }
39             var result = sb.ToString();
40             var comment = issueService.Comment.Create(owner, context.Repository.Name,
41                 ↪ context.Number, result);
42             comment.Wait();
43             Console.WriteLine($"Last commit activity comment is added:
44                 ↪ {comment.Result.HtmlUrl}");
45             _storage.CloseIssue(context);
46         }
47
48         public HashSet<string> GetIgnoredRepositories(IList<Link> links)
49         {
50             HashSet<string> ignoredRepos = new() { };
51         }
52     }
53 }

```

```

47         foreach (var link in links)
48         {
49             var values = link.Values;
50             if (values != null && values.Count == 3 && string.Equals(values.First().Id,
                    ↪ "ignore", StringComparison.OrdinalIgnoreCase) &&
                    ↪ string.Equals(values.Last().Id.Trim('.'), "repository",
                    ↪ StringComparison.OrdinalIgnoreCase))
51             {
52                 ignoredRepos.Add(values[1].Id);
53             }
54         }
55         return ignoredRepos;
56     }
57
58     public HashSet<Activity> GetActivities(string owner, HashSet<string> ignoredRepositories
59     ↪ = default)
60     {
61         HashSet<Activity> activeUsers = new();
62         foreach (var repository in _storage.Client.Repository.GetAllForOrg(owner).Result)
63         {
64             if (ignoredRepositories?.Contains(repository.Name) ?? false)
65             {
66                 continue;
67             }
68             foreach (var commit in _storage.GetCommits(repository.Owner.Login,
69             ↪ repository.Name, DateTime.Today.AddMonths(-3)))
70             {
71                 if (!_storage.Client.Organization.Member.CheckMember(owner,
72                 ↪ commit.Author.Login).Result)
73                 {
74                     continue;
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     }
93     return activeUsers;
94 }
95
96 }
97
98 }

```

### 1.7 ./csharp/Platform.Bot/Triggers/MergeDependabotBumpsTrigger.cs

```

1 using System;
2 using Interfaces;
3 using Octokit;
4 using Platform.Threading;
5 using Storage.Remote.GitHub;
6
7 namespace Platform.Bot.Triggers
8 {
9     using TContext = PullRequest;
10    public class MergeDependabotBumpsTrigger : ITrigger<TContext>
11    {
12        private const int DependabotId = 49699333;
13        private readonly GitHubStorage _storage;
14        public MergeDependabotBumpsTrigger(GitHubStorage storage)
15        {
16            _storage = storage;
17        }
18
19        public bool Condition(TContext context)
20        {
21            var isDependabotAuthor = DependabotId == context.User.Id;
22            var isTestAndDeployCompleted = false;
23            var repositoryId = context.Base.Repository.Id;
24            var checks = _storage.Client.Check.Run.GetAllForReference(repositoryId,
                ↪ context.Head.Sha).AwaitResult();

```

```

25         foreach (var checkRun in checks.CheckRuns)
26         {
27             if (checkRun.Name == "testAndDeploy" && checkRun.Status.Value ==
                ↳ CheckStatus.Completed)
28             {
29                 isTestAndDeployCompleted = true;
30             }
31         }
32         return isDependabotAuthor && isTestAndDeployCompleted;
33     }
34
35     public void Action(TContext context)
36     {
37         var repositoryId = context.Base.Repository.Id;
38         var prMerge = _storage.Client.PullRequest.Merge(repositoryId, context.Number, new
                ↳ MergePullRequest()).AwaitResult();
39         Console.WriteLine($"{context.HtmlUrl} is {(prMerge.Merged ? "successfully":"not
                ↳ successfully")} merged.");
40     }
41 }
42 }

```

## 1.8 ./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using Interfaces;
5  using Octokit;
6  using Platform.Communication.Protocol.Lino;
7  using Storage.Remote.GitHub;
8
9  namespace Platform.Bot.Triggers
10 {
11     using TContext = Issue;
12     /// <summary>
13     /// <para>
14     /// Represents the organization last month activity trigger.
15     /// </para>
16     /// <para></para>
17     /// </summary>
18     /// <seealso cref="ITrigger{Issue}"/>
19     internal class OrganizationLastMonthActivityTrigger : ITrigger<TContext>
20     {
21         private readonly GitHubStorage _storage;
22         private readonly Parser _parser = new();
23
24         /// <summary>
25         /// <para>
26         /// Initializes a new <see cref="OrganizationLastMonthActivityTrigger"/> instance.
27         /// </para>
28         /// <para></para>
29         /// </summary>
30         /// <param name="storage">
31         /// <para>A storage.</para>
32         /// <para></para>
33         /// </param>
34         public OrganizationLastMonthActivityTrigger(GitHubStorage storage) => _storage = storage;
35
36         /// <summary>
37         /// <para>
38         /// Determines whether this instance condition.
39         /// </para>
40         /// <para></para>
41         /// </summary>
42         /// <param name="context">
43         /// <para>The context.</para>
44         /// <para></para>
45         /// </param>
46         /// <returns>
47         /// <para>The bool</para>
48         /// <para></para>
49         /// </returns>
50         public bool Condition(TContext context) => context.Title.ToLower() == "organization last
                ↳ month activity";
51
52         /// <summary>
53         /// <para>
54         /// Actions the context.
55         /// </para>

```

```

56     /// <para></para>
57     /// </summary>
58     /// <param name="context">
59     /// <para>The context.</para>
60     /// <para></para>
61     /// </param>
62     public void Action(TContext context)
63     {
64         var issueService = _storage.Client.Issue;
65         var owner = context.Repository.Owner.Login;
66         var activeUsersString = string.Join("\n",
        ↪ GetActiveUsers(GetIgnoredRepositories(_parser.Parse(context.Body)), owner));
67         issueService.Comment.Create(owner, context.Repository.Name, context.Number,
        ↪ activeUsersString);
68         _storage.CloseIssue(context);
69     }
70
71     /// <summary>
72     /// <para>
73     /// Gets the ignored repositories using the specified links.
74     /// </para>
75     /// <para></para>
76     /// </summary>
77     /// <param name="links">
78     /// <para>The links.</para>
79     /// <para></para>
80     /// </param>
81     /// <returns>
82     /// <para>The ignored repos.</para>
83     /// <para></para>
84     /// </returns>
85     public HashSet<string> GetIgnoredRepositories(IList<Link> links)
86     {
87         HashSet<string> ignoredRepos = new() { };
88         foreach (var link in links)
89         {
90             var values = link.Values;
91             if (values != null && values.Count == 3 && string.Equals(values.First().Id,
        ↪ "ignore", StringComparison.OrdinalIgnoreCase) &&
        ↪ string.Equals(values.Last().Id.Trim('.'), "repository",
        ↪ StringComparison.OrdinalIgnoreCase))
92             {
93                 ignoredRepos.Add(values[1].Id);
94             }
95         }
96         return ignoredRepos;
97     }
98
99     /// <summary>
100    /// <para>
101    /// Gets the active users using the specified ignored repositories.
102    /// </para>
103    /// <para></para>
104    /// </summary>
105    /// <param name="ignoredRepositories">
106    /// <para>The ignored repositories.</para>
107    /// <para></para>
108    /// </param>
109    /// <param name="owner">
110    /// <para>The owner.</para>
111    /// <para></para>
112    /// </param>
113    /// <returns>
114    /// <para>The active users.</para>
115    /// <para></para>
116    /// </returns>
117    public HashSet<string> GetActiveUsers(HashSet<string> ignoredRepositories, string owner)
118    {
119        HashSet<string> activeUsers = new();
120        var date = DateTime.Now.AddMonths(-1);
121        foreach (var repository in _storage.Client.Repository.GetAllForOrg(owner).Result)
122        {
123            if (ignoredRepositories.Contains(repository.Name))
124            {
125                continue;
126            }
127            foreach (var commit in _storage.GetCommits(repository.Owner.Login,
        ↪ repository.Name))

```



```

128     {
129         activeUsers.Add(commit.Author.Login);
130     }
131     foreach (var pullRequest in _storage.GetPullRequests(repository.Owner.Login,
132         ↪ repository.Name))
133     {
134         foreach (var reviewer in pullRequest.RequestedReviewers)
135         {
136             if (pullRequest.CreatedAt < date || pullRequest.UpdatedAt < date ||
137                 ↪ pullRequest.ClosedAt < date || pullRequest.MergedAt < date)
138             {
139                 activeUsers.Add(reviewer.Login);
140             }
141         }
142     }
143     foreach (var createdIssue in _storage.GetIssues(repository.Owner.Login,
144         ↪ repository.Name))
145     {
146         activeUsers.Add(createdIssue.User.Login);
147     }
148     return activeUsers;
149 }
150 }
151 }

```

### 1.9 ./csharp/Platform.Bot/Triggers/ProtectMainBranchTrigger.cs

```

1  using System.Collections.Generic;
2  using System.Linq;
3  using System.Text;
4  using Interfaces;
5  using Octokit;
6  using Storage.Remote.GitHub;
7
8  namespace Platform.Bot.Triggers
9  {
10     using TContext = Issue;
11     public class ProtectMainBranchTrigger : ITrigger<TContext>
12     {
13         private readonly GitHubStorage _storage;
14         public ProtectMainBranchTrigger(GitHubStorage storage) => _storage = storage;
15         public bool Condition(TContext context) => context.Title.ToLower() == "protect default
16             ↪ branch in all organization's repositories";
17
18         public void Action(TContext context)
19         {
20             var repositories =
21                 ↪ _storage.Client.Repository.GetAllForOrg(context.Repository.Owner.Login).Result;
22             var results = UpdateRepositoriesDefaultBranchProtection(repositories);
23             StringBuilder failedRepositoriesComment = new(repositories.Count *
24                 ↪ repositories[0].Name.Length);
25             foreach (var result in results.Where(result => !result.Value))
26             {
27                 failedRepositoriesComment.AppendLine($"- [ ] {result.Key}");
28             }
29             if (failedRepositoriesComment.Length != 0)
30             {
31                 failedRepositoriesComment.AppendLine(
32                     "TODO: Fix default branch protection of these repositories. Failed
33                     ↪ repositories:");
34                 _storage.Client.Issue.Comment.Create(context.Repository.Id, context.Number,
35                     ↪ failedRepositoriesComment.ToString());
36             }
37             else
38             {
39                 _storage.Client.Issue.Comment.Create(context.Repository.Id, context.Number,
40                     ↪ "Success. All repositories default branch protection is updated.");
41                 _storage.CloseIssue(context);
42             }
43         }
44
45         public Dictionary<string, bool>
46             ↪ UpdateRepositoriesDefaultBranchProtection(IReadOnlyList<Repository> repositories)
47         {
48             Dictionary<string, bool> result = new(repositories.Count);
49             foreach (var repository in repositories)

```

```
43     {
44         if (repository.Private)
45         {
46             continue;
47         }
48         var update = new BranchProtectionSettingsUpdate(new
49             ↪ BranchProtectionPushRestrictionsUpdate());
50         var request =
51             _storage.Client.Repository.Branch.UpdateBranchProtection(repository.Id,
52                 repository.DefaultBranch, update);
53         request.Wait();
54         result.Add(repository.Name, request.IsCompletedSuccessfully);
55     }
56     return result;
57 }
58 }
```

## Index

- ./csharp/Platform.Bot/Program.cs, 1
- ./csharp/Platform.Bot/Trackers/IssueTracker.cs, 1
- ./csharp/Platform.Bot/Trackers/PullRequestTracker.cs, 2
- ./csharp/Platform.Bot/Triggers/Activity.cs, 4
- ./csharp/Platform.Bot/Triggers/HelloWorldTrigger.cs, 4
- ./csharp/Platform.Bot/Triggers/LastCommitActivityTrigger.cs, 5
- ./csharp/Platform.Bot/Triggers/MergeDependabotBumpsTrigger.cs, 6
- ./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs, 7
- ./csharp/Platform.Bot/Triggers/ProtectMainBranchTrigger.cs, 9