

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

## 1.2 ./csharp/Platform.Bot/Trackers/IssueTracker.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.Trackers
9 {
10     /// <summary>
11     /// <para>
12     /// Represents the programmer role.
13     /// </para>
14     /// <para></para>
15     /// </summary>
16     public class IssueTracker : ITracker<Issue>
17     {
18         /// <summary>
```

```

19     /// <para>
20     /// The git hub api.
21     /// </para>
22     /// <para></para>
23     /// </summary>
24     public GitHubStorage GitHubApi { get; }
25
26     /// <summary>
27     /// <para>
28     /// The minimum interaction interval.
29     /// </para>
30     /// <para></para>
31     /// </summary>
32     public TimeSpan MinimumInteractionInterval { get; }
33
34     /// <summary>
35     /// <para>
36     /// The triggers.
37     /// </para>
38     /// <para></para>
39     /// </summary>
40     public List<ITrigger<Issue>> Triggers { get; }
41
42     /// <summary>
43     /// <para>
44     /// Initializes a new <see cref="IssueTracker"/> 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 IssueTracker(List<ITrigger<Issue>> triggers, GitHubStorage gitHubApi)
57     {
58         GitHubApi = gitHubApi;
59         Triggers = triggers;
60         MinimumInteractionInterval = gitHubApi.MinimumInteractionInterval;
61     }
62
63     /// <summary>
64     /// <para>
65     /// Starts the cancellation token.
66     /// </para>
67     /// <para></para>
68     /// </summary>
69     /// <param name="cancellationToken">
70     /// <para>The cancellation token.</para>
71     /// <para></para>
72     /// </param>
73     public void Start(CancellationToken cancellationToken)
74     {
75         while (!cancellationToken.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 }

```

### 1.3 ./csharp/Platform.Bot/Trackers/PullRequestTracker.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  using Platform.Threading;
8
9  namespace Platform.Bot.Trackers
10 {
11     using TContext = PullRequest;
12     /// <summary>
13     /// <para>
14     /// Represents the programmer role.
15     /// </para>
16     /// <para></para>
17     /// </summary>
18     public class PullRequestTracker : ITracker<TContext>
19     {
20         /// <summary>
21         /// <para>
22         /// The git hub api.
23         /// </para>
24         /// <para></para>
25         /// </summary>
26         public GitHubStorage GitHubApi { get; }
27
28         /// <summary>
29         /// <para>
30         /// The minimum interaction interval.
31         /// </para>
32         /// <para></para>
33         /// </summary>
34         public TimeSpan MinimumInteractionInterval { get; }
35
36         /// <summary>
37         /// <para>
38         /// The triggers.
39         /// </para>
40         /// <para></para>
41         /// </summary>
42         public List<ITrigger<TContext>> Triggers { get; }
43
44         /// <summary>
45         /// <para>
46         /// Initializes a new <see cref="IssueTracker"/> instance.
47         /// </para>
48         /// <para></para>
49         /// </summary>
50         /// <param name="triggers">
51         /// <para>A triggers.</para>
52         /// <para></para>
53         /// </param>
54         /// <param name="gitHubApi">
55         /// <para>A git hub api.</para>
56         /// <para></para>
57         /// </param>
58         public PullRequestTracker(List<ITrigger<TContext>> triggers, GitHubStorage gitHubApi)
59         {
60             GitHubApi = gitHubApi;
61             Triggers = triggers;
62             MinimumInteractionInterval = gitHubApi.MinimumInteractionInterval;
63         }
64
65         /// <summary>
66         /// <para>
67         /// Starts the cancellation token.
68         /// </para>
69         /// <para></para>
70         /// </summary>
71         /// <param name="cancellationToken">
72         /// <para>The cancellation token.</para>
73         /// <para></para>
74         /// </param>
75         public void Start(CancellationTokens cancellationToken)
76         {
77             while (!cancellationToken.IsCancellationRequested)
78             {
79                 foreach (var trigger in Triggers)
80                 {
81

```

```

83         foreach (var repository in
            ↪ GitHubApi.Client.Repository.GetAllForOrg("linksplatform").AwaitResult())
84         {
85             foreach (var pullRequest in GitHubApi.Client.PullRequest.GetAllForReposi_
            ↪ tory(repository.Id).AwaitResult())
86             {
87                 if (trigger.Condition(pullRequest))
88                 {
89                     trigger.Action(pullRequest);
90                 }
91             }
92         }
93     }
94     Thread.Sleep(MinimumInteractionInterval);
95 }
96 }
97 }
98 }

```

#### 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 gitHubApi;
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="gitHubApi">
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 gitHubApi, FileStorage fileStorage, string
            ↪ fileSetName)
41        {
42            this.gitHubApi = gitHubApi;
43            this.fileStorage = fileStorage;
44            this.fileSetName = fileSetName;
45        }

```

```

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             gitHubApi.CreateOrUpdateFile(context.Repository.Name,
64                 ↪ context.Repository.DefaultBranch, file);
65         }
66         gitHubApi.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         }
40     }
41 }

```

```

36     }
37     var result = sb.ToString();
38     var comment = issueService.Comment.Create(owner, context.Repository.Name,
39         ↪ context.Number, result);
40     comment.Wait();
41     Console.WriteLine($"Last commit activity comment is added:
42         ↪ {comment.Result.HtmlUrl}");
43     Storage.CloseIssue(context);
44 }
45
46 public HashSet<string> GetIgnoredRepositories(IList<Link> links)
47 {
48     HashSet<string> ignoredRepos = new() { };
49     foreach (var link in links)
50     {
51         var values = link.Values;
52         if (values != null && values.Count == 3 && string.Equals(values.First().Id,
53             ↪ "ignore", StringComparison.OrdinalIgnoreCase) &&
54             ↪ string.Equals(values.Last().Id.Trim('.'), "repository",
55             ↪ StringComparison.OrdinalIgnoreCase))
56         {
57             ignoredRepos.Add(values[1].Id);
58         }
59     }
60     return ignoredRepos;
61 }
62
63 public HashSet<Activity> GetActivities(string owner, HashSet<string> ignoredRepositories
64     ↪ = default)
65 {
66     HashSet<Activity> activeUsers = new();
67     foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
68     {
69         if (ignoredRepositories?.Contains(repository.Name) ?? false)
70         {
71             continue;
72         }
73         foreach (var commit in Storage.GetCommits(repository.Owner.Login,
74             ↪ repository.Name, DateTime.Today.AddMonths(-3)))
75         {
76             if (!Storage.Client.Organization.Member.CheckMember(owner,
77                 ↪ commit.Author.Login).Result)
78             {
79                 continue;
80             }
81             if (!activeUsers.Any(x => x.Url == commit.Author.Url))
82             {
83                 activeUsers.Add(new Activity() { Url = commit.Author.Url, Repositories =
84                     ↪ new List<string> { repository.Url } });
85             }
86             else
87             {
88                 if (!activeUsers.Any(x => x.Repositories.Any(y => y == repository.Url)
89                     ↪ == true))
90                 {
91                     activeUsers.FirstOrDefault(x => x.Url ==
92                         ↪ commit.Author.Url).Repositories.Add(repository.Url);
93                 }
94             }
95         }
96     }
97     return activeUsers;
98 }
99 }
100 }

```

## 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 readonly 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,
25             ↪ context.Head.Sha).AwaitResult();
26         foreach (var checkRun in checks.CheckRuns)
27         {
28             if (checkRun.Name == "testAndDeploy" && checkRun.Status.Value ==
29                 ↪ CheckStatus.Completed)
30             {
31                 isTestAndDeployCompleted = true;
32             }
33         }
34         return isDependabotAuthor && isTestAndDeployCompleted;
35     }
36
37     public void Action(TContext context)
38     {
39         var repositoryId = context.Base.Repository.Id;
40         var prMerge = Storage.Client.PullRequest.Merge(repositoryId, context.Number, new
41             ↪ MergePullRequest()).AwaitResult();
42         Console.WriteLine($"{context.HtmlUrl} is {(prMerge.Merged ? "successfully":"not
43             ↪ successfully")} merged.");
44     }
45 }

```

## 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,
128                 ↪ repository.Name))
129             {
130                 activeUsers.Add(commit.Author.Login);
131             }
132             foreach (var pullRequest in Storage.GetPullRequests(repository.Owner.Login,
133                 ↪ repository.Name))
134             {
135                 foreach (var reviewer in pullRequest.RequestedReviewers)
136                 {
137                     if (pullRequest.CreatedAt < date || pullRequest.UpdatedAt < date ||
138                         ↪ pullRequest.ClosedAt < date || pullRequest.MergedAt < date)
139                     {
140                         activeUsers.Add(reviewer.Login);
141                     }
142                 }
143             }
144             foreach (var createdIssue in Storage.GetIssues(repository.Owner.Login,
145                 ↪ repository.Name))
146             {
147                 activeUsers.Add(createdIssue.User.Login);
148             }
149         }
150         return activeUsers;
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         }
38     }
39 }

```

```

31     }
32     else
33     {
34         Storage.Client.Issue.Comment.Create(context.Repository.Id, context.Number,
35             ↳ "Success. All repositories default branch protection is updated.");
36         Storage.CloseIssue(context);
37     }
38 }
39 public Dictionary<string, bool>
↳ UpdateRepositoriesDefaultBranchProtection(IReadOnlyList<Repository> repositories)
40 {
41     Dictionary<string, bool> result = new(repositories.Count);
42     foreach (var repository in repositories)
43     {
44         if (repository.Private)
45         {
46             continue;
47         }
48         var update = new BranchProtectionSettingsUpdate(new
↳ BranchProtectionPushRestrictionsUpdate());
49         var request =
50             Storage.Client.Repository.Branch.UpdateBranchProtection(repository.Id,
51                 repository.DefaultBranch, update);
52         request.Wait();
53         result.Add(repository.Name, request.IsCompletedSuccessfully);
54     }
55     return result;
56 }
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