

## 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         private static void Main(string[] args)
21         {
22             using var cancellation = new ConsoleCancellation();
23             var argumentIndex = 0;
24             var username = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Username", args);
25             var token = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Token", args);
26             var appName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "App Name", args);
27             var databaseFileName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Database
28                 ↪ file name", args);
29             var fileSetName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "File set name
30                 ↪ ", args);
31             var dbContext = new FileStorage(databaseFileName);
32             Console.WriteLine($"Bot has been started. {Environment.NewLine}Press CTRL+C to
33                 ↪ close");
34             try
35             {
36                 var api = new GitHubStorage(username, token, appName);
37                 new ProgrammerRole(
38                     new List<ITrigger<Issue>> {
39                         new HelloWorldTrigger(api, dbContext, fileSetName),
40                         new OrganizationLastMonthActivityTrigger(api),
41                         new LastCommitActivityTrigger(api),
42                         new ProtectMainBranchTrigger(api)
43                     },
44                     api
45                 ).Start(cancellation.Token);
46             }
47             catch (Exception ex)
48             {
49                 Console.WriteLine(ex.ToStringWithAllInnerExceptions());
50             }
51         }
52     }
53 }
```

### 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     private void ProcessIssues(CancellationTokens token)
63     {
64         while (!token.IsCancellationRequested)
65         {
66             foreach (var issue in GitHubAPI.GetIssues())
67             {
68                 foreach (var trigger in Triggers)
69                 {
70                     if (trigger.Condition(issue))
71                     {
72                         trigger.Action(issue);
73                     }
74                 }
75             }
76             Thread.Sleep(MinimumInteractionInterval);
77         }
78     }
79
80     /// <summary>
81     /// <para>
82     /// Starts the cancellation token.
83     /// </para>
84     /// <para></para>
85     /// </summary>
86     /// <param name="cancellationToken">
87     /// <para>The cancellation token.</para>
88     /// <para></para>
89     /// </param>
90     public void Start(CancellationTokens cancellationToken) =>
91     {
92         ProcessIssues(cancellationToken);
93     }

```

### 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         private readonly GitHubStorage gitHubAPI;
18         private readonly FileStorage fileStorage;
19         private readonly string fileSetName;
20
21         /// <summary>
22         /// <para>
23         /// Initializes a new <see cref="HelloWorldTrigger"/> instance.
24         /// </para>
25         /// <para></para>
26         /// </summary>
27         /// <param name="gitHubAPI">
28         /// <para>A git hub api.</para>
29         /// <para></para>
30         /// </param>
31         /// <param name="fileStorage">
32         /// <para>A file storage.</para>
33         /// <para></para>
34         /// </param>
35         /// <param name="fileSetName">
36         /// <para>A file set name.</para>
37         /// <para></para>
38         /// </param>
39         public HelloWorldTrigger(GitHubStorage gitHubAPI, FileStorage fileStorage, string
40         ↪ fileSetName)
41         {
42             this.gitHubAPI = gitHubAPI;
43             this.fileStorage = fileStorage;
44             this.fileSetName = fileSetName;
45         }
46
47         /// <summary>
48         /// <para>
49         /// Actions the obj.
50         /// </para>
51         /// <para></para>
52         /// </summary>
53         /// <param name="obj">
54         /// <para>The obj.</para>
55         /// <para></para>
56         /// </param>
57
58         public void Action(Issue obj)
59         {
60             foreach (var file in fileStorage.GetFilesFromSet(fileSetName))
61             {
62                 gitHubAPI.CreateOrUpdateFile(obj.Repository.Name, obj.Repository.DefaultBranch,
63                 ↪ file);
64             }
65             gitHubAPI.CloseIssue(obj);
66         }
67
68         /// <summary>
69         /// <para>

```

```

70     /// Determines whether this instance condition.
71     /// </para>
72     /// <para></para>
73     /// </summary>
74     /// <param name="obj">
75     /// <para>The obj.</para>
76     /// <para></para>
77     /// </param>
78     /// <returns>
79     /// <para>The bool</para>
80     /// <para></para>
81     /// </returns>
82     public bool Condition(Issue obj) => obj.Title.ToLower() == "hello world";
83 }
84 }

```

## 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
21         ↳ activity";
22
23         public void Action(Issue issue)
24         {
25             var issueService = Storage.Client.Issue;
26             var owner = issue.Repository.Owner.Login;
27             var ignoredRepositories =
28                 issue.Body != null ? GetIgnoredRepositories(Parser.Parse(issue.Body)) : default;
29             var users = GetActivities(owner, ignoredRepositories);
30             StringBuilder sb = new();
31             foreach (var user in users)
32             {
33                 sb.AppendLine($"- **{user.Url.Replace("api.", "").Replace("users/", "")}**");
34                 foreach (var repo in user.Repositories)
35                 {
36                     sb.AppendLine($"- {repo.Replace("api.", "").Replace("repos/", "")}");
37                 }
38                 // Break line
39                 sb.AppendLine("-----");
40             }
41             var result = sb.ToString();
42             var comment = issueService.Comment.Create(owner, issue.Repository.Name,
43                 ↳ issue.Number, result);
44             comment.Wait();
45             Console.WriteLine($"Last commit activity comment is added:
46             ↳ {comment.Result.HtmlUrl}");
47             Storage.CloseIssue(issue);
48         }
49
50         public HashSet<string> GetIgnoredRepositories(IList<Link> links)
51         {
52             HashSet<string> ignoredRepos = new() { };
53             foreach (var link in links)
54             {
55                 var values = link.Values;
56                 if (values != null && values.Count == 3 && string.Equals(values.First().Id,
57                     ↳ "ignore", StringComparison.OrdinalIgnoreCase) &&
58                     ↳ string.Equals(values.Last().Id.Trim('.'), "repository",
59                     ↳ StringComparison.OrdinalIgnoreCase))
60                 {
61                     ignoredRepos.Add(values[1].Id);
62                 }
63             }
64         }
65     }
66 }

```

```

57     }
58     return ignoredRepos;
59 }
60
61 public HashSet<Activity> GetActivities(string owner, HashSet<string> ignoredRepositories
62     ↪ = default)
63 {
64     HashSet<Activity> activeUsers = new();
65     foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
66     {
67         if (ignoredRepositories?.Contains(repository.Name) ?? false)
68         {
69             continue;
70         }
71         foreach (var commit in Storage.GetCommits(repository.Owner.Login,
72             ↪ repository.Name, DateTime.Today.AddMonths(-3)))
73         {
74             if (!Storage.Client.Organization.Member.CheckMember(owner,
75                 ↪ commit.Author.Login).Result)
76             {
77                 continue;
78             }
79             if (!activeUsers.Any(x => x.Url == commit.Author.Url))
80             {
81                 activeUsers.Add(new Activity() { Url = commit.Author.Url, Repositories =
82                     ↪ new List<string> { repository.Url } });
83             }
84             else
85             {
86                 if (!activeUsers.Any(x => x.Repositories.Any(y => y == repository.Url)
87                     ↪ == true))
88                 {
89                     activeUsers.FirstOrDefault(x => x.Url ==
90                         ↪ commit.Author.Url).Repositories.Add(repository.Url);
91                 }
92             }
93         }
94     }
95     return activeUsers;
96 }
97
98 }
99
100 }

```

## 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         private readonly GitHubStorage Storage;
21         private readonly Parser Parser = new();
22
23         /// <summary>
24         /// <para>
25         /// Initializes a new <see cref="OrganizationLastMonthActivityTrigger"/> instance.
26         /// </para>
27         /// <para></para>
28         /// </summary>
29         /// <param name="storage">
30         /// <para>A storage.</para>
31         /// <para></para>
32         /// </param>
33         public OrganizationLastMonthActivityTrigger(GitHubStorage storage) => Storage = storage;
34
35         /// <summary>

```

```

36    /// <para>
37    /// Determines whether this instance condition.
38    /// </para>
39    /// <para></para>
40    /// </summary>
41    /// <param name="issue">
42    /// <para>The issue.</para>
43    /// <para></para>
44    /// </param>
45    /// <returns>
46    /// <para>The bool</para>
47    /// <para></para>
48    /// </returns>
49    public bool Condition(Issue issue) => issue.Title.ToLower() == "organization last month
    ↳ activity";
50
51    /// <summary>
52    /// <para>
53    /// Actions the issue.
54    /// </para>
55    /// <para></para>
56    /// </summary>
57    /// <param name="issue">
58    /// <para>The issue.</para>
59    /// <para></para>
60    /// </param>
61    public void Action(Issue issue)
62    {
63        var issueService = Storage.Client.Issue;
64        var owner = issue.Repository.Owner.Login;
65        var activeUsersString = string.Join("\n",
        ↳ GetActiveUsers(GetIgnoredRepositories(Parser.Parse(issue.Body)), owner));
66        issueService.Comment.Create(owner, issue.Repository.Name, issue.Number,
        ↳ activeUsersString);
67        Storage.CloseIssue(issue);
68    }
69
70    /// <summary>
71    /// <para>
72    /// Gets the ignored repositories using the specified links.
73    /// </para>
74    /// <para></para>
75    /// </summary>
76    /// <param name="links">
77    /// <para>The links.</para>
78    /// <para></para>
79    /// </param>
80    /// <returns>
81    /// <para>The ignored repos.</para>
82    /// <para></para>
83    /// </returns>
84    public HashSet<string> GetIgnoredRepositories(ICollection<Link> links)
85    {
86        HashSet<string> ignoredRepos = new() { };
87        foreach (var link in links)
88        {
89            var values = link.Values;
90            if (values != null && values.Count == 3 && string.Equals(values.First().Id,
            ↳ "ignore", StringComparison.OrdinalIgnoreCase) &&
            ↳ string.Equals(values.Last().Id.Trim('.'), "repository",
            ↳ StringComparison.OrdinalIgnoreCase))
            {
91                ignoredRepos.Add(values[1].Id);
92            }
93        }
94        return ignoredRepos;
95    }
96
97
98    /// <summary>
99    /// <para>
100    /// Gets the active users using the specified ignored repositories.
101    /// </para>
102    /// <para></para>
103    /// </summary>
104    /// <param name="ignoredRepositories">
105    /// <para>The ignored repositories.</para>
106    /// <para></para>
107    /// </param>

```

```

108     /// <param name="owner">
109     /// <para>The owner.</para>
110     /// <para></para>
111     /// </param>
112     /// <returns>
113     /// <para>The active users.</para>
114     /// <para></para>
115     /// </returns>
116     public HashSet<string> GetActiveUsers(HashSet<string> ignoredRepositories, string owner)
117     {
118         HashSet<string> activeUsers = new();
119         var date = DateTime.Now.AddMonths(-1);
120         foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
121         {
122             if (ignoredRepositories.Contains(repository.Name))
123             {
124                 continue;
125             }
126             foreach (var commit in Storage.GetCommits(repository.Owner.Login,
127                 ↪ 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                 foreach (var createdIssue in Storage.GetIssues(repository.Owner.Login,
143                     ↪ repository.Name))
144                 {
145                     activeUsers.Add(createdIssue.User.Login);
146                 }
147             }
148             return activeUsers;
149         }
150     }

```

## 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)

```

```

28     {
29         failedRepositoriesComment.AppendLine(
30             "TODO: Fix default branch protection of these repositories. Failed
            ↳ repositories:");
31         Storage.Client.Issue.Comment.Create(issue.Repository.Id, issue.Number,
            ↳ failedRepositoriesComment.ToString());
32     }
33     else
34     {
35         Storage.Client.Issue.Comment.Create(issue.Repository.Id, issue.Number, "Success.
            ↳ All repositories default branch protection is updated.");
36         Storage.CloseIssue(issue);
37     }
38 }
39
40 public Dictionary<string, bool>
    ↳ UpdateRepositoriesDefaultBranchProtection(IReadOnlyList<Repository> repositories)
41 {
42     Dictionary<string, bool> result = new(repositories.Count);
43     foreach (var repository in repositories)
44     {
45         if (repository.Private)
46         {
47             continue;
48         }
49         var update = new BranchProtectionSettingsUpdate(new
            ↳ 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 }
59 }

```



## Index

- ./csharp/Platform.Bot/Program.cs, 1
- ./csharp/Platform.Bot/ProgrammerRole.cs, 1
- ./csharp/Platform.Bot/Triggers/Activity.cs, 2
- ./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, 7