

GRIMSHAW

connect Dynamo  
to the cloud

@radugidei

@radugidei

GRIMSHAW

# Radu Gidei

## Profile

Hi, I'm Radu Gidei and I work as a BIM Manager for Grimshaw in London.

**GRIMSHAW**  
Design Technology

# Grimshaw

## Profile

Founded in 1980, Grimshaw operates worldwide, with offices in London, New York, Doha, Kuala Lumpur, Melbourne and Sydney, employing over 400 staff.

Our international portfolio covers various sectors, with substantial experience in:

- Aviation
- Leisure schemes
- Transport
- Offices
- Education
- Sports
- Industrial sector
- master planning

Total Global Staff

447

Years in Business

36

Awards for our work

207

Nationalities Represented

50

# Grimshaw

## Profile



# What we'll look at

- > Why
- > How

to connect Dynamo to cloud services.

## We've all used Dynamo for...

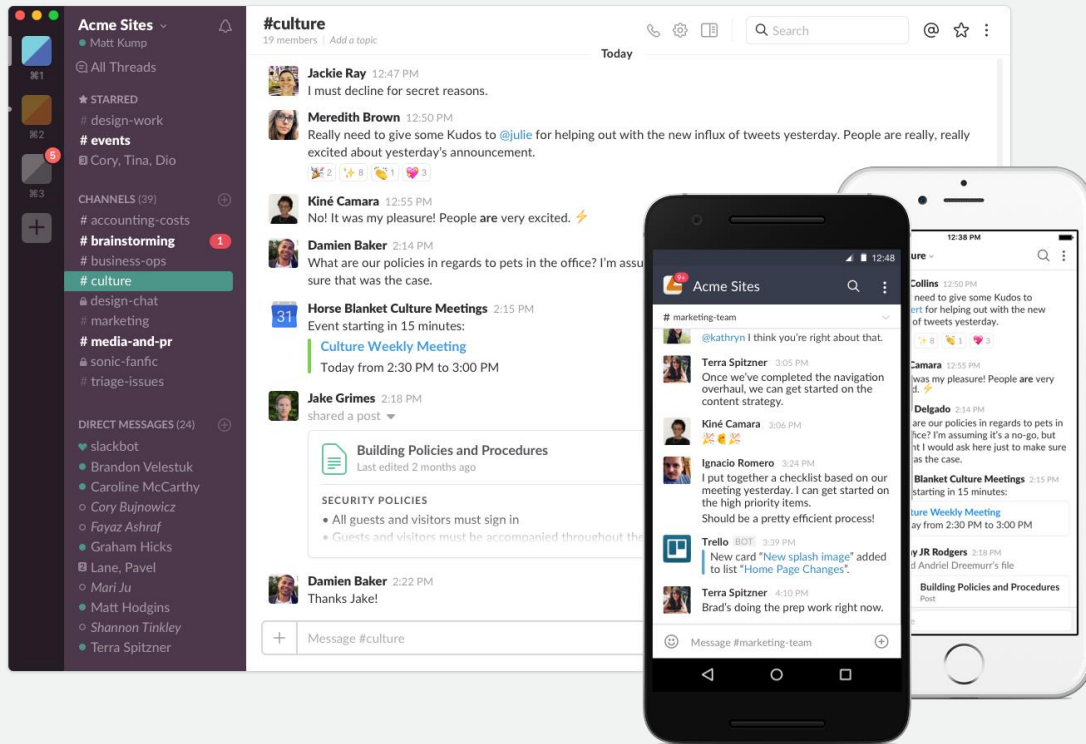
- > Data management
- > Data extraction
- > Geometry rationalisation
- > Geometry generation
- > Automating repetitive Revit tasks
- > Interop
- > Auditing

## We've not used Dynamo extensively for...

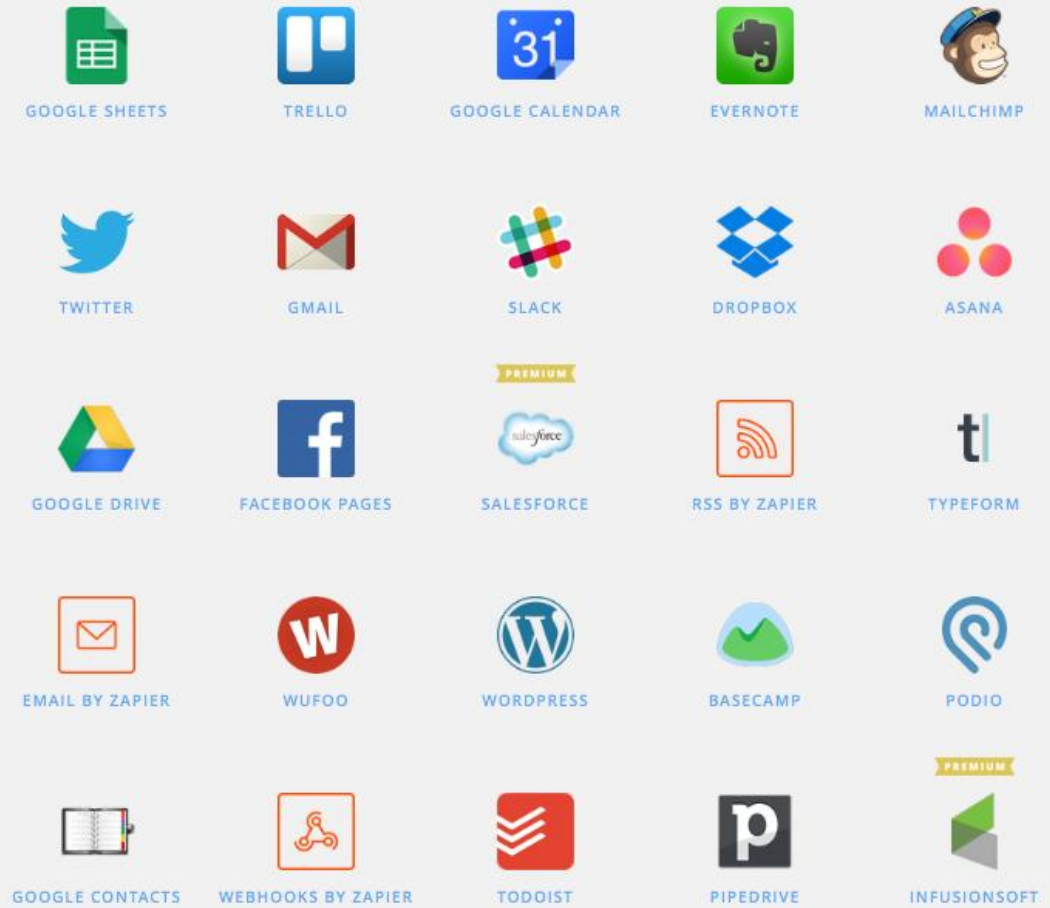
- › Project management
- › Practice or project-wide workflows
- › Connecting to business systems

# Use cases

## Practice workflows



## Connect to business systems

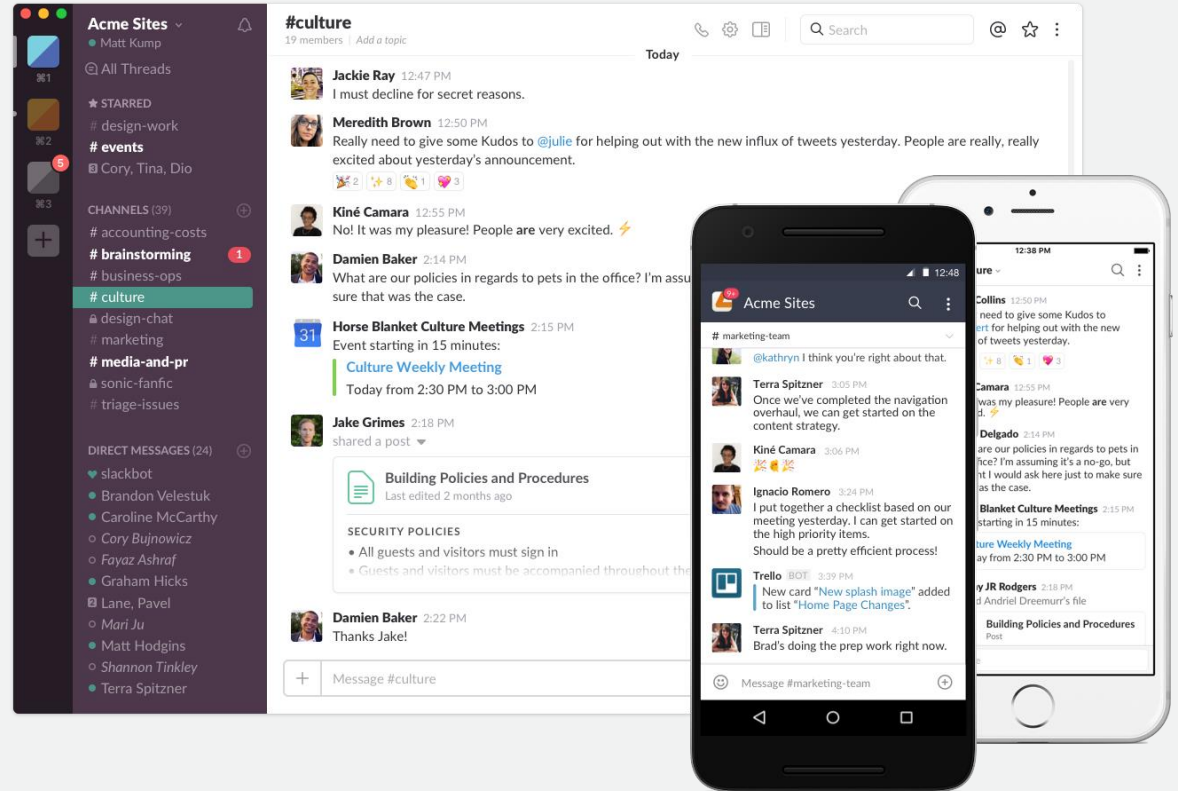




# Practice workflow

## Slack

- › Event-based notifications
- › Running log notifications
- › Bot interaction

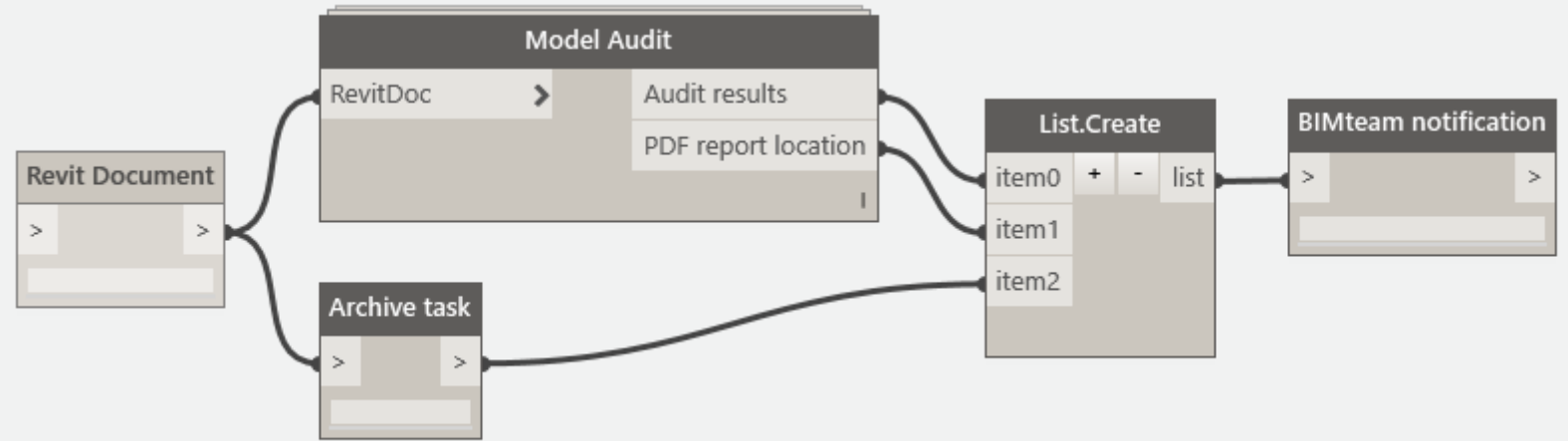


# Slack usage scenario


## Notification of model audit

self-serving Dynamo definition for model audit

- > Revit model is archived
- > auditing is performed
- > #BIMteam is notified of audit & results



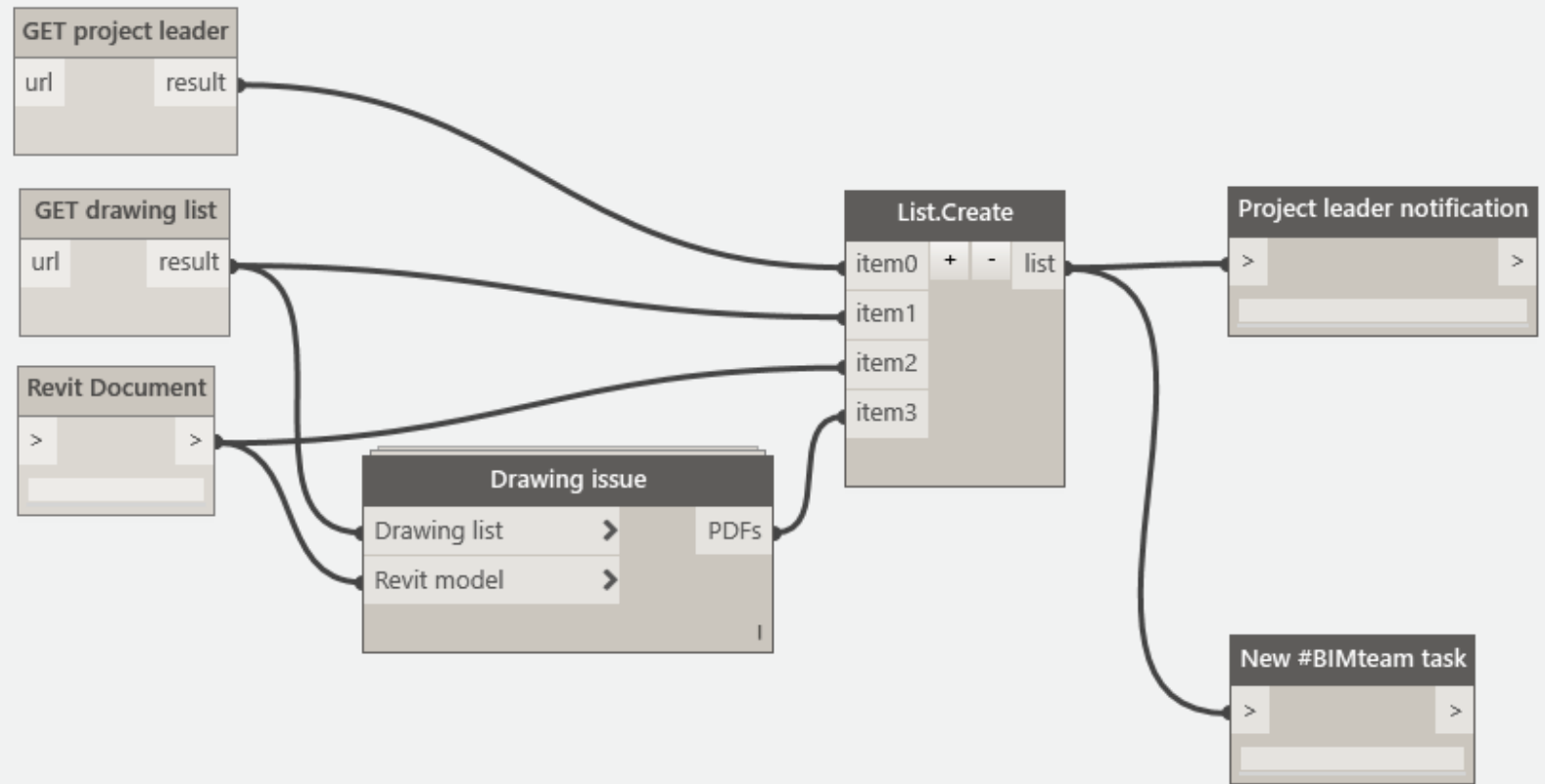
new messages

 **DynaBot** APP 1:34 PM  
Project number 17001 was just audited.  
Result : **FAIL**  
Archive : `\\srv\wip\17001\archive\170228-17001-audit_radu.rvt`

# Cloud app scenario

## Query & Post to business app

- > retrieve information from an app
- > use information in Dynamo
- > post results to a different business app



### 17001 ProjectName :

- ✓ Review LIDAR survey specs >
- ✓ Drawing issue sheet >

Ok...but how ?

## Web protocols

Not recommended :

- > proprietary protocols

Recommended :

- > REST APIs
- > **webhooks**

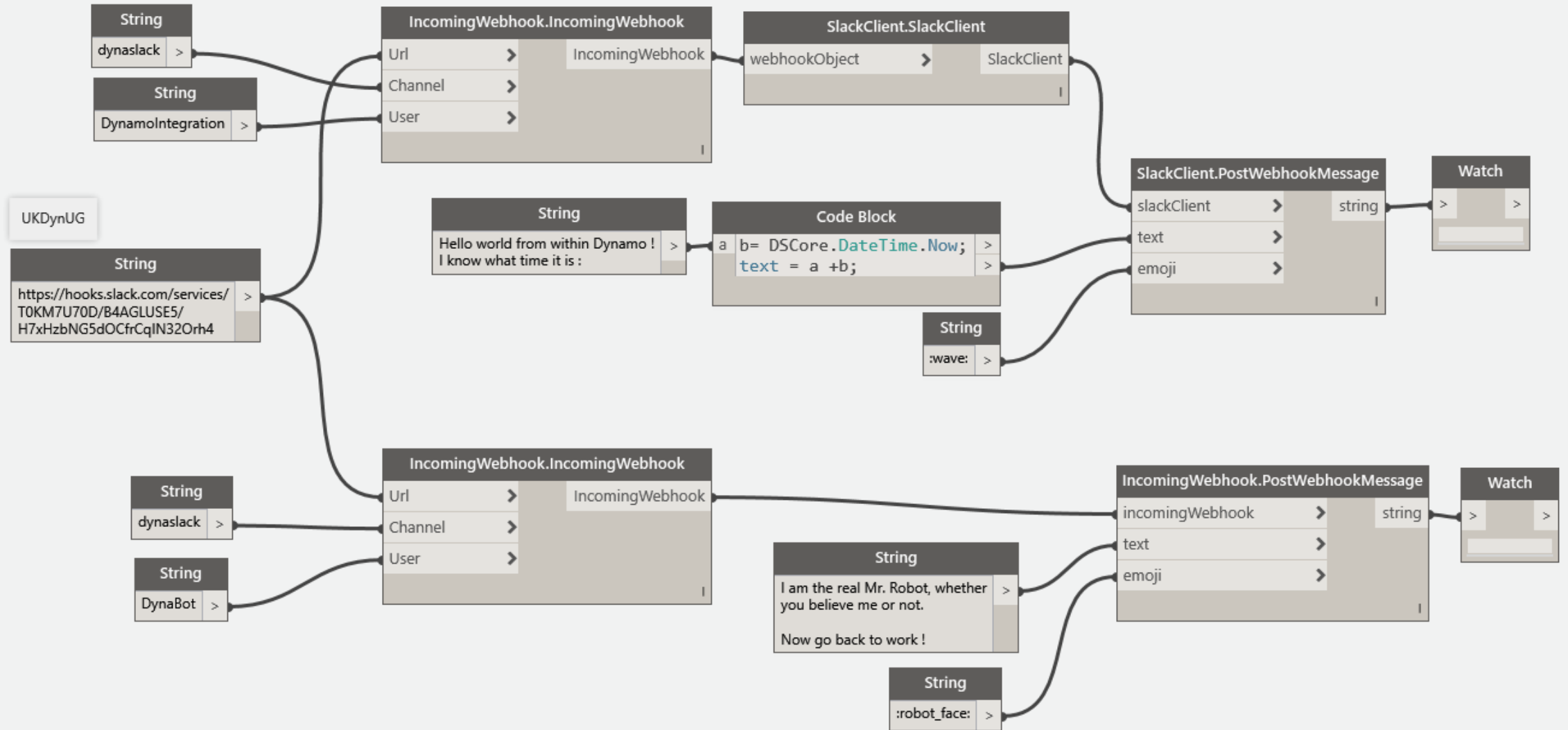


## Scripting

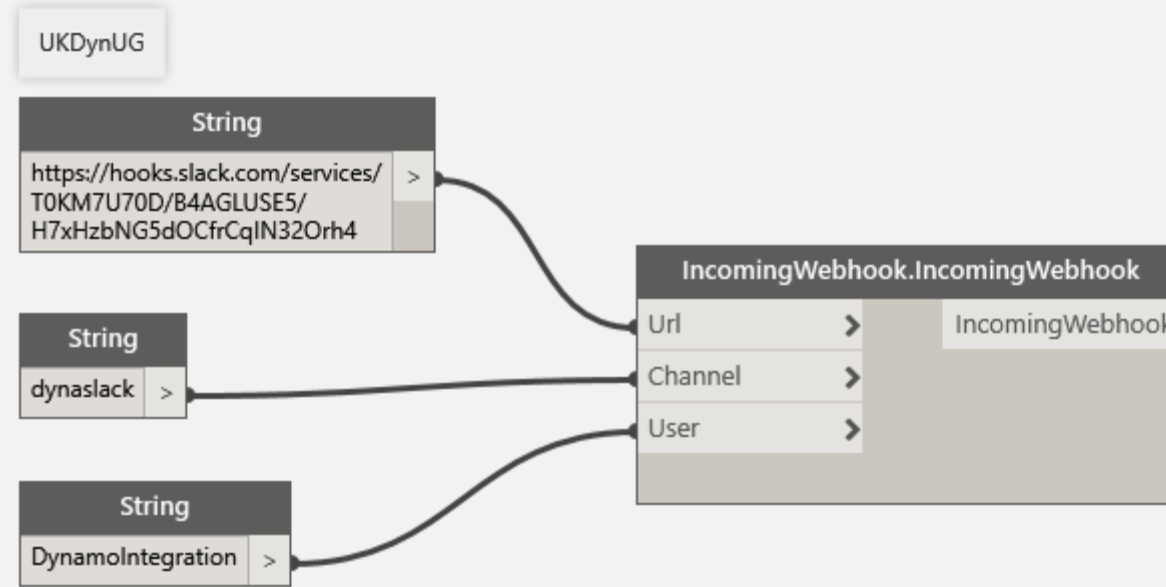
- > Python
- > **Zero-touch ( C# )**



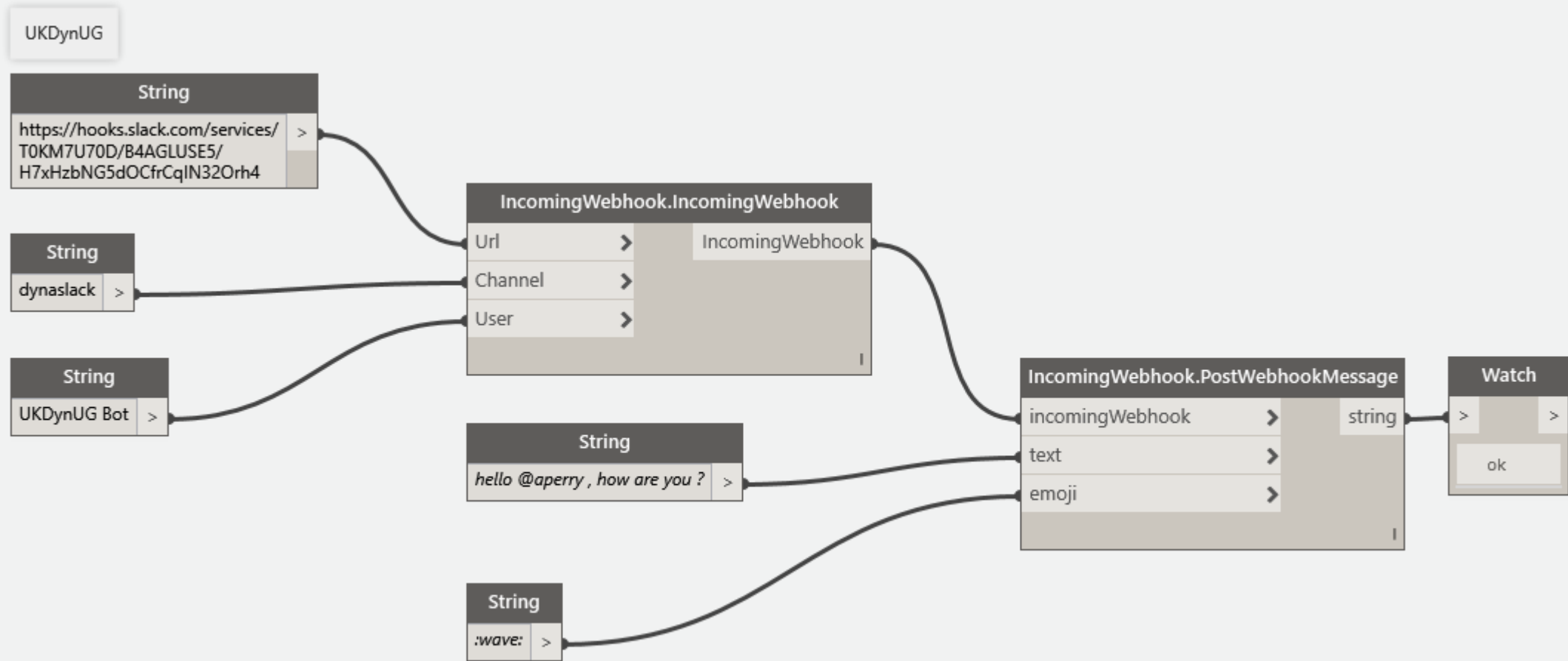
# Today's definition



# Webhook



# Post message





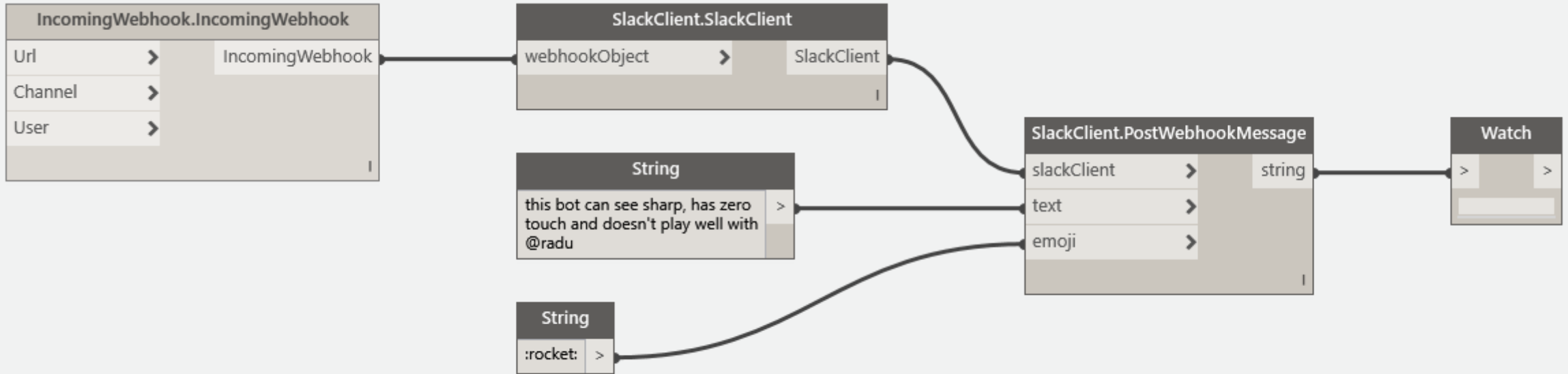
@radugidei

GRIMSHAW

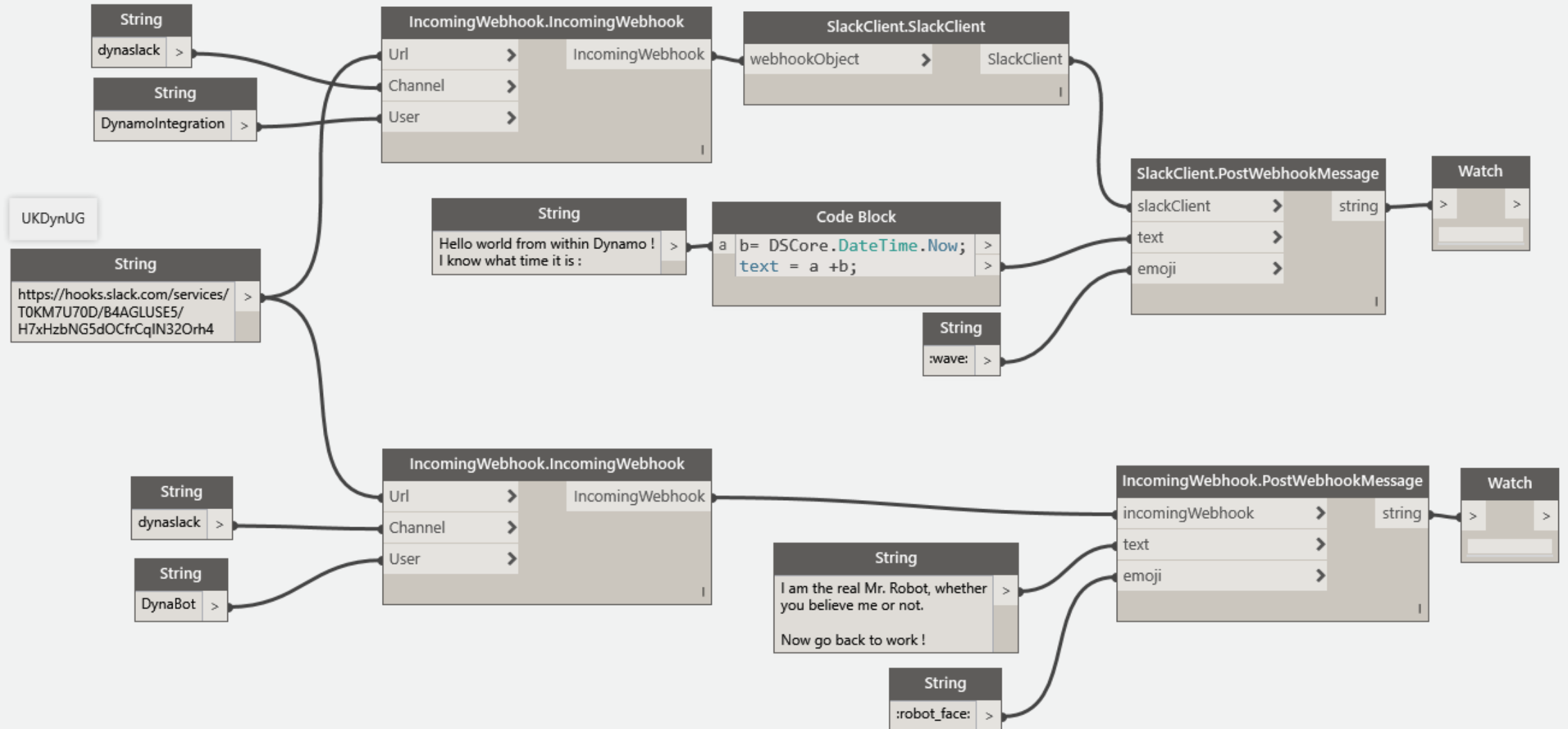
# Post message

demo

# Slack client



# Today's definition

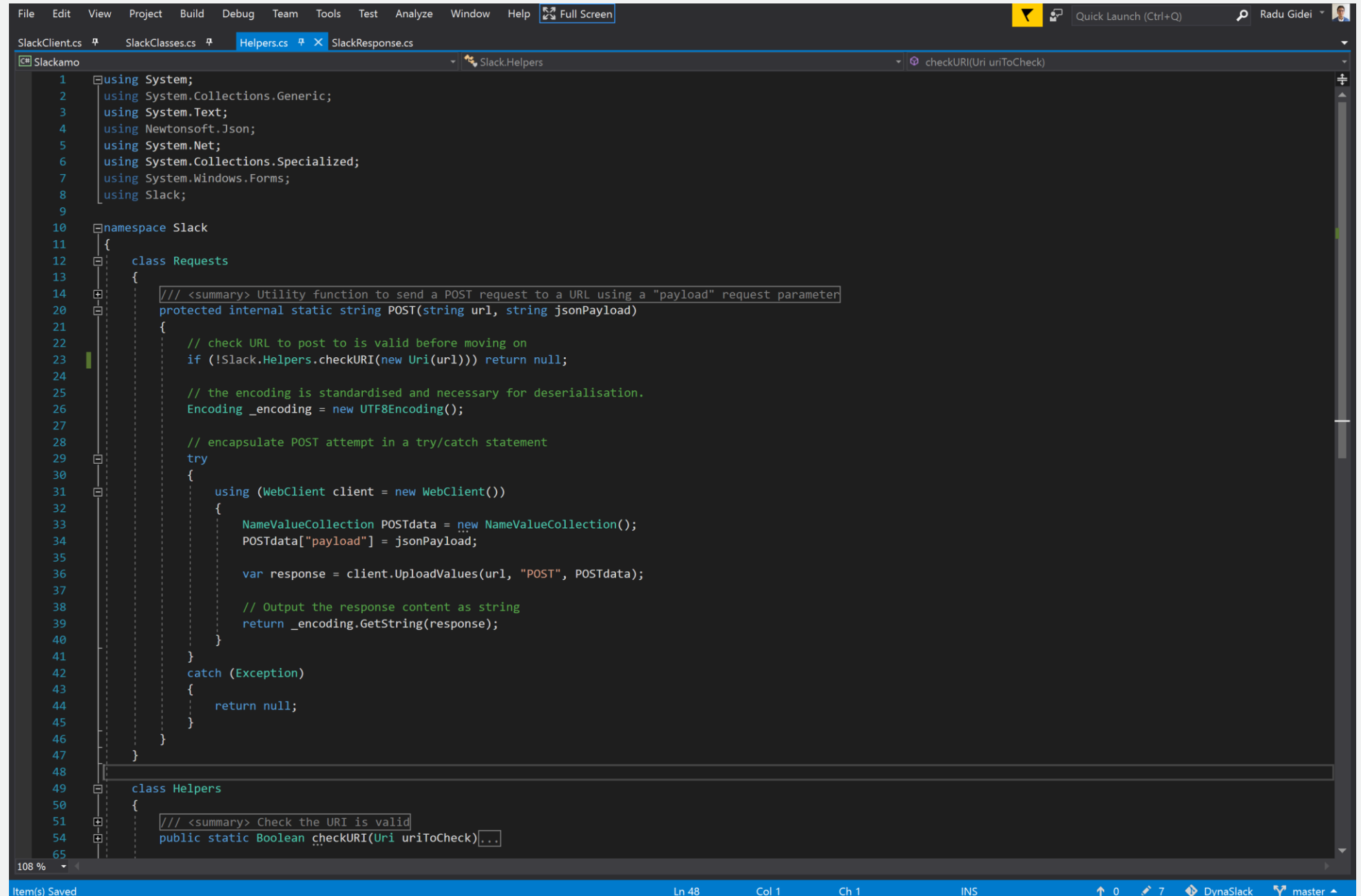


# Today's definition

demo

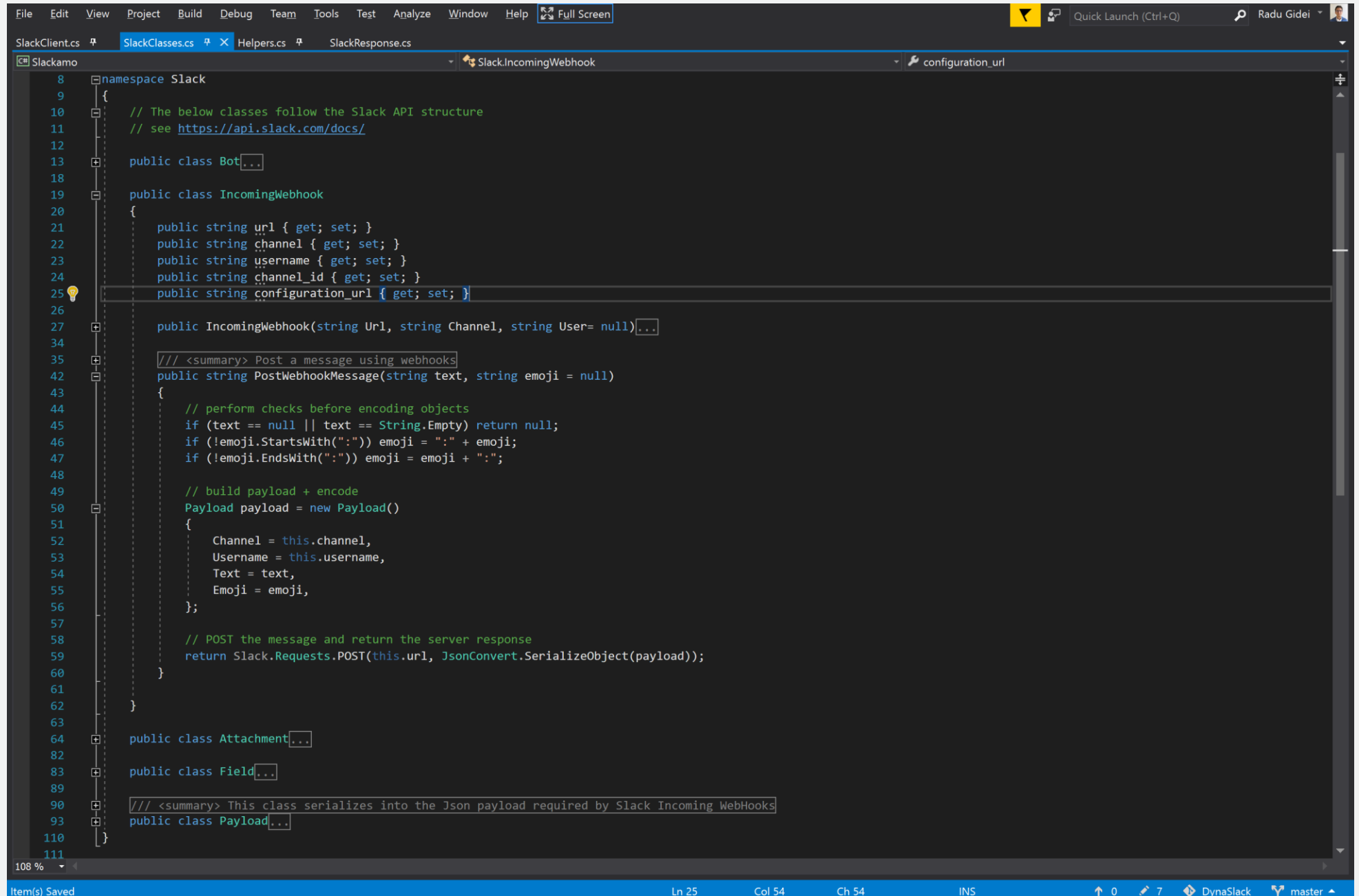
Ok...but how much code was that ?

# Post message



```
File Edit View Project Build Debug Team Tools Test Analyze Window Help Full Screen
SlackClient.cs SlackClasses.cs Helpers.cs SlackResponse.cs
Slackamo Slack.Helpers checkURI(Uri uriToCheck)
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using Newtonsoft.Json;
5 using System.Net;
6 using System.Collections.Specialized;
7 using System.Windows.Forms;
8 using Slack;
9
10 namespace Slack
11 {
12     class Requests
13     {
14         /// <summary> Utility function to send a POST request to a URL using a "payload" request parameter
15         protected internal static string POST(string url, string jsonPayload)
16         {
17             // check URL to post to is valid before moving on
18             if (!Slack.Helpers.checkURI(new Uri(url))) return null;
19
20             // the encoding is standardised and necessary for deserialisation.
21             Encoding _encoding = new UTF8Encoding();
22
23             // encapsulate POST attempt in a try/catch statement
24             try
25             {
26                 using (WebClient client = new WebClient())
27                 {
28                     NameValueCollection POSTdata = new NameValueCollection();
29                     POSTdata["payload"] = jsonPayload;
30
31                     var response = client.UploadValues(url, "POST", POSTdata);
32
33                     // Output the response content as string
34                     return _encoding.GetString(response);
35                 }
36             }
37             catch (Exception)
38             {
39                 return null;
40             }
41         }
42     }
43
44     class Helpers
45     {
46         /// <summary> Check the URI is valid
47         public static Boolean checkURI(Uri uriToCheck)...
```

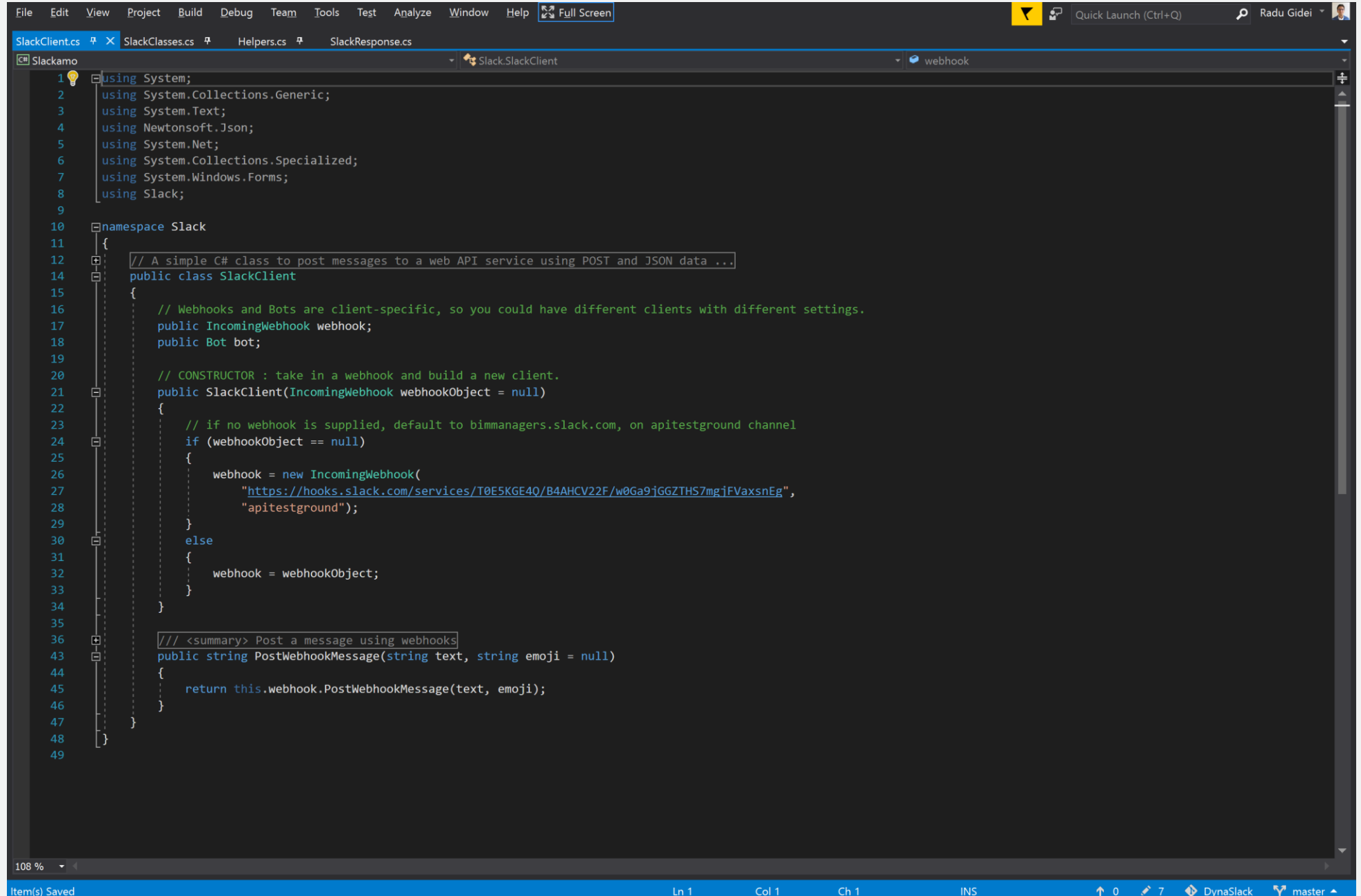
# Webhook



The image shows a screenshot of the Visual Studio Code editor with a C# file named `SlackClasses.cs` open. The code is organized into a `namespace Slack` and contains several public classes: `Bot`, `IncomingWebhook`, `Attachment`, `Field`, and `Payload`. The `IncomingWebhook` class has properties for `url`, `channel`, `username`, `channel_id`, and `configuration_url`. It also has a constructor and a `PostWebhookMessage` method that handles message posting, including emoji handling and payload building. The `Payload` class is intended to be serialized into a JSON payload for the webhook request.

```
8 namespace Slack
9 {
10     // The below classes follow the Slack API structure
11     // see https://api.slack.com/docs/
12
13     public class Bot{...}
14
15     public class IncomingWebhook
16     {
17         public string url { get; set; }
18         public string channel { get; set; }
19         public string username { get; set; }
20         public string channel_id { get; set; }
21         public string configuration_url { get; set; }
22
23         public IncomingWebhook(string Url, string Channel, string User= null){...}
24
25         /// <summary> Post a message using webhooks
26         public string PostWebhookMessage(string text, string emoji = null)
27         {
28             // perform checks before encoding objects
29             if (text == null || text == String.Empty) return null;
30             if (!emoji.StartsWith(":")) emoji = ":" + emoji;
31             if (!emoji.EndsWith(":")) emoji = emoji + ":";
32
33             // build payload + encode
34             Payload payload = new Payload()
35             {
36                 Channel = this.channel,
37                 Username = this.username,
38                 Text = text,
39                 Emoji = emoji,
40             };
41
42             // POST the message and return the server response
43             return Slack.Requests.POST(this.url, JsonConvert.SerializeObject(payload));
44         }
45     }
46
47     public class Attachment{...}
48
49     public class Field{...}
50
51     /// <summary> This class serializes into the Json payload required by Slack Incoming WebHooks
52     public class Payload{...}
53 }
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
```

# Slack Client



```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using Newtonsoft.Json;
5 using System.Net;
6 using System.Collections.Specialized;
7 using System.Windows.Forms;
8 using Slack;
9
10 namespace Slack
11 {
12     // A simple C# class to post messages to a web API service using POST and JSON data ...
13     public class SlackClient
14     {
15         // Webhooks and Bots are client-specific, so you could have different clients with different settings.
16         public IncomingWebhook webhook;
17         public Bot bot;
18
19         // CONSTRUCTOR : take in a webhook and build a new client.
20         public SlackClient(IncomingWebhook webhookObject = null)
21         {
22             // if no webhook is supplied, default to bimmanagers.slack.com, on apitestground channel
23             if (webhookObject == null)
24             {
25                 webhook = new IncomingWebhook(
26                     "https://hooks.slack.com/services/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",
27                     "apitestground");
28             }
29             else
30             {
31                 webhook = webhookObject;
32             }
33         }
34
35         /// <summary> Post a message using webhooks
36         public string PostWebhookMessage(string text, string emoji = null)
37         {
38             return this.webhook.PostWebhookMessage(text, emoji);
39         }
40     }
41 }
42
43
44
45
46
47
48
49
```







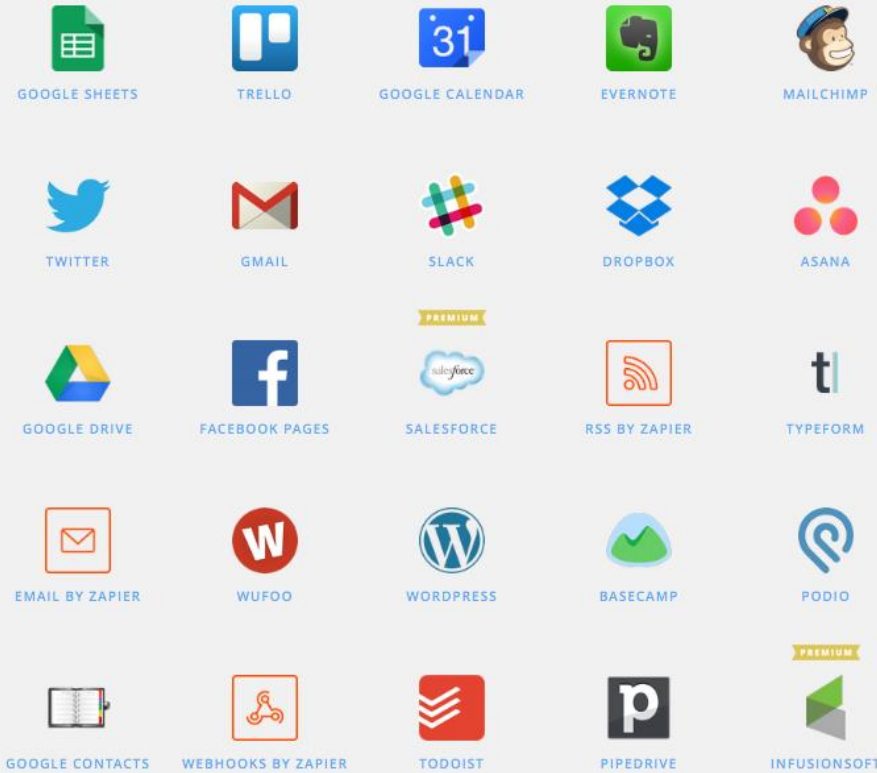
# Closing thoughts

Any questions ?



## Build a Custom Integration

-  **Incoming WebHooks** >  
Send data into Slack in real-time.
-  **Bots** >  
Connect a bot to the Slack Real Time Messaging API.
-  **Slash Commands** >  
Customized Slack commands for your team.
-  **Outgoing WebHooks** >  
Get data out of Slack in real-time.



search the Dynamo Package Manager for :

**DynaSlack**