



Building and generating a .NET client for Space

Maarten Balliauw
@maartenballiauw

**JET
BRAINS**



Agenda

JetBrains Space

JetBrains Space HTTP API

Building a .NET SDK for the Space HTTP API

Code generation

Developer experience

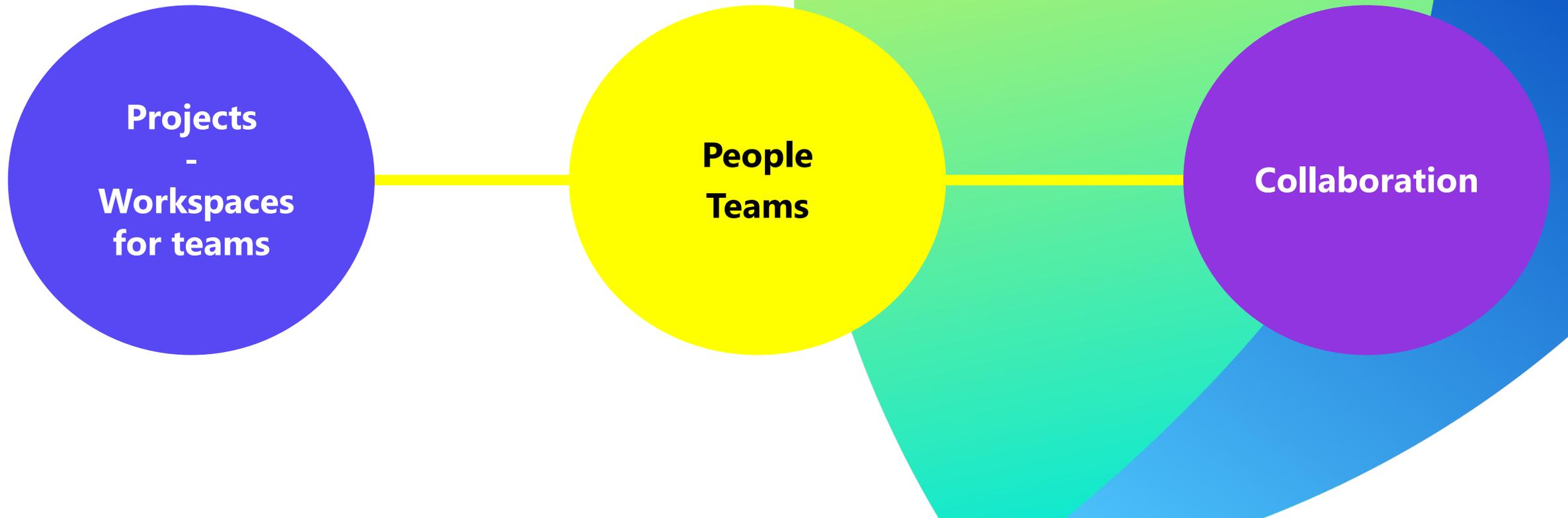
A wide-angle view of Earth from space, showing the curvature of the planet and the atmosphere. The sun is rising over the horizon on the left, creating a bright orange and yellow glow. The Earth's surface is visible, showing landmasses and clouds. The sky is dark with many stars.

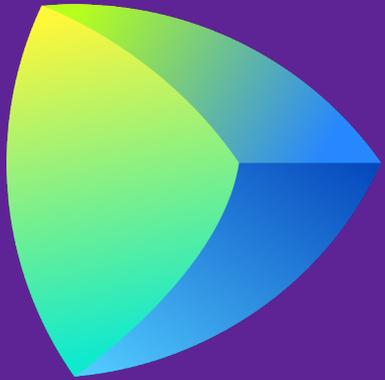
JetBrains Space

JetBrains Space

Integrated Team Environment

<https://jetbrains.space>

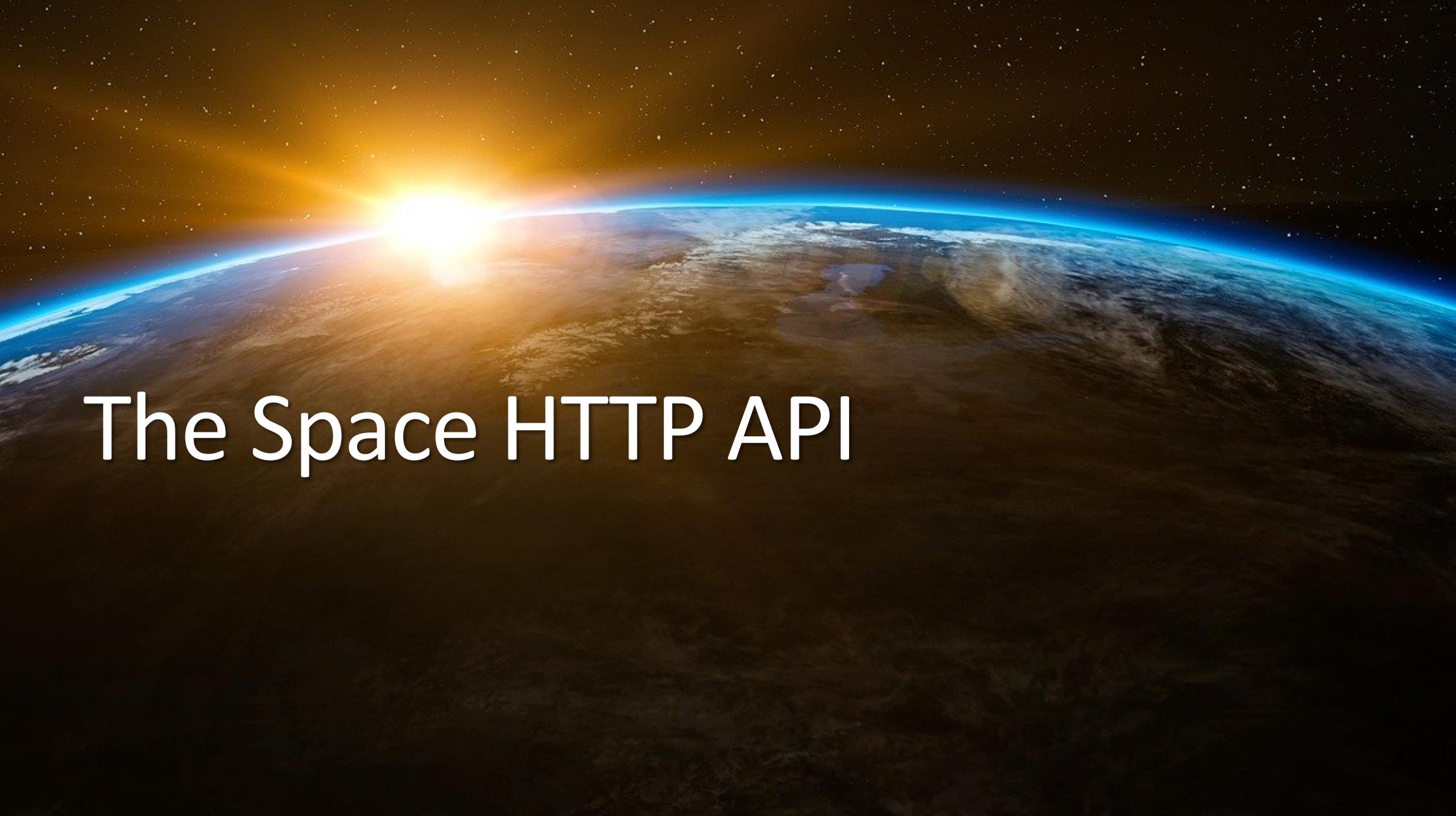




JetBrains Space

demo



A wide-angle view of Earth from space, showing the curvature of the planet and the atmosphere. The sun is rising over the horizon on the left, creating a bright orange and yellow glow. The Earth's surface is visible, showing continents and oceans. The sky is dark with many stars.

The Space HTTP API

Integrated Team Environment

More information in Space == more value from Space

Internal integrations out of the box

- Vacation shown in profile

- Blog post targeted to team/location

- Calendar integration in blog post

- Everything can be a ToDo item

- ...

What with external integrations? Extensibility?

Everything as an HTTP API!

What do you want to build? Which integration do you need?

- Import data

- Post to chat

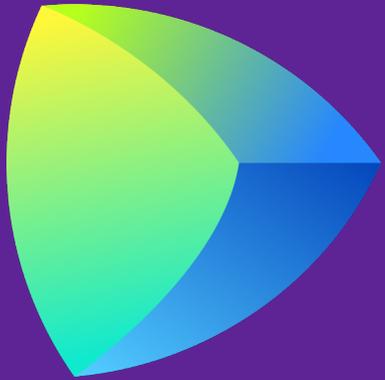
- Manage users, teams, and locations programmatically

- Build dashboards

No good answer, so opening up everything

- Large HTTP API surface...

- HTTP API Playground to explore!



HTTP API Playground

demo



\$fields

Top-level fields returned by default

Unless \$fields query parameter used

Use \$fields to be specific, or get nested fields

\$fields=id,icon,name

\$fields=id,name,boards(id,description)

\$fields=id,name,boards(id,info(columns))

\$fields=*,boards(id,description)

```
PR_Project {  
   id: string  
   adminProfiles: [TD_MemberProfile {  +  }]  
   adminTeams: [TD_Team {  +  }]  
   archived: boolean  
   boards: [BoardRecord {  -  }  
     id: string  
     archived: boolean  
     info: BoardInfo {  -  }  
       columns: BoardColumns {  -  }  
         columns: [BoardColumn {  -  }  
           default: boolean?  
           name: string  
           statuses: [IssueStatus {  +  }]  
        }  
      }  
     description: string?  
     owners: BoardOwners {  +  }  
  }  
   name: string  
}  
}]  
 description: string?
```



Building a .NET SDK for the Space HTTP API

Hello, .NET! 🙌

Goal: make our users successful at using Space in their team/organization

- Provide a ready-to-use SDK

- Minimal configuration needed

- Discoverability

- Use latest C# language features (where they make sense)

TL;DR: Lead integration developers on the shortest path to success

How? Handcrafted

No.

Over 150 API endpoints

Over 700 object types

Space EAP / Space Beta means churn!

How? OpenAPI?

Use [NSwag](#), generate based on OpenAPI / Swagger document

Some incompatible object types needed custom code generation

The image shows a split-screen view. On the left is the Swagger Editor interface, displaying a large JSON OpenAPI document. The document defines various components like 'UnfurAttachment', 'UnfurDetailsArticle', 'UnfurDetailsCodeSnippet', etc. On the right is a REST client interface for a service named 'Space'. It shows a dropdown menu for servers (currently set to '/') and an 'Authorize' button. Below that, the 'default' server is selected, and a list of endpoints is shown:

- POST /absences Create Absence
- GET /absences Get All Absences
- POST /absences/{id}/approve Approve Absence
- GET /absences/member:{member} Get All Absences by Member

How? OpenAPI?

Not very developer friendly...

```
public class Body
{
    public string Member { get; set; }
    public string Reason { get; set; }
    public string Description { get; set; }
    public string Location { get; set; }
    public string Since { get; set; }
    public string Till { get; set; }
    public bool Available { get; set; } = false;
    public string Icon { get; set; }
}

public async Task<AbsenceRecord> GetAbsencesAsync(
    string fields, Body body, CancellationToken cancellationToken)
{
}
}
```

How? Custom code generation

[https://your.jetbrains.space/api/http/http-api-model?\\$fields=dto,enums,urlParams,resources\(*,nestedResources!\)](https://your.jetbrains.space/api/http/http-api-model?$fields=dto,enums,urlParams,resources(*,nestedResources!))

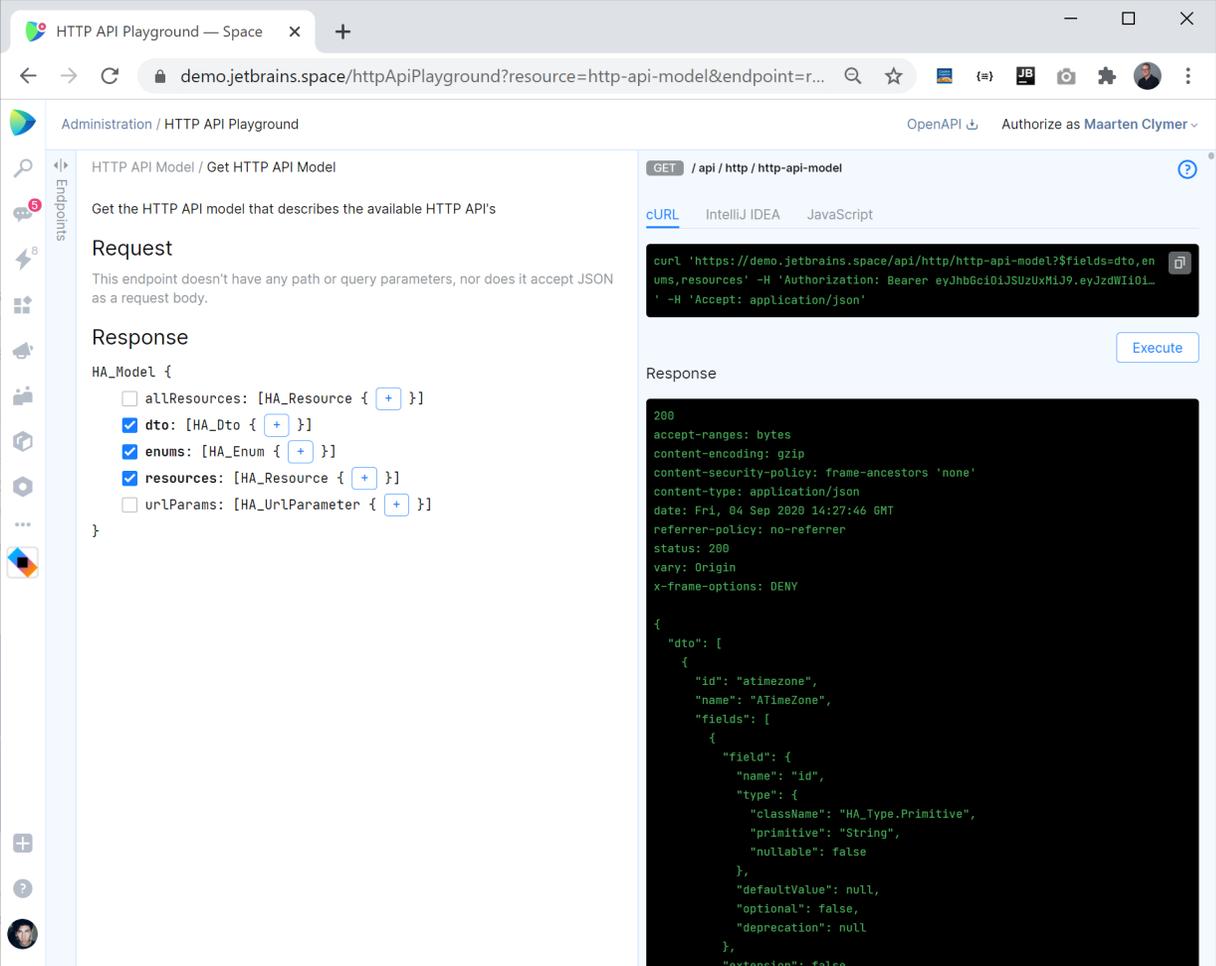
Space exposes HTTP API Model

Enumerations

DTOs (objects passed around)

Resources

+ data types, nullability,
default values, ...



The screenshot shows the HTTP API Playground interface. The browser address bar displays the URL: `demo.jetbrains.space/httpApiPlayground?resource=http-api-model&endpoint=r...`. The page title is "Administration / HTTP API Playground". The main content area shows the details for the endpoint "GET /api/http/http-api-model".

Request
This endpoint doesn't have any path or query parameters, nor does it accept JSON as a request body.

Response
The response structure is defined as follows:

```
HA_Model {  
   allResources: [HA_Resource { + }]  
   dto: [HA_Dto { + }]  
   enums: [HA_Enum { + }]  
   resources: [HA_Resource { + }]  
   urlParams: [HA_UrlParameter { + }]  
}
```

The response body is shown in a dark-themed code editor. It starts with a 200 status and various headers, followed by a JSON object:

```
200  
accept-ranges: bytes  
content-encoding: gzip  
content-security-policy: frame-ancestors 'none'  
content-type: application/json  
date: Fri, 04 Sep 2020 14:27:46 GMT  
referrer-policy: no-referrer  
status: 200  
vary: Origin  
x-frame-options: DENY  
  
{  
  "dto": [  
    {  
      "id": "atimezone",  
      "name": "ATimeZone",  
      "fields": [  
        {  
          "field": {  
            "name": "id",  
            "type": {  
              "className": "HA_Type.Primitive",  
              "primitive": "String",  
              "nullable": false  
            },  
            "defaultValue": null,  
            "optional": false,  
            "deprecation": null  
          },  
          "extension": false  
        }  
      ]  
    }  
  ]  
}
```

Enumeration model

```
public class ApiEnum
{
    public string Id { get; set; }

    public ApiDeprecation? Deprecation { get; set; }

    public string Name { get; set; }

    public List<string> Values { get; set; }
}
```

```
public class ApiDeprecation
{
    public string? Message { get; set; }

    public string? Since { get; set; }

    public bool ForRemoval { get; set; }
}
```

Enumeration code generator

```
public class EnumerationVisitor
{
    public string Visit(ApiEnum apiEnum)
    {
        var builder = new StringBuilder();

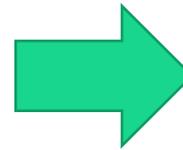
        if (apiEnum.Deprecation != null)
            builder.AppendLine(Visit(apiEnum.Deprecation));

        builder.AppendLine("public enum " + apiEnum.Name);
        builder.AppendLine("{}");

        foreach (var apiEnumValue in apiEnum.Values)
        {
            builder.AppendLine($" {apiEnumValue},");
        }

        builder.AppendLine("{}");
        return builder.ToString();
    }

    public string Visit(ApiDeprecation apiDeprecation)
        => $"[Obsolete(\"{apiDeprecation.Message}\")]";
}
```



```
public enum AbsenceListMode
{
    All,
    WithAccessibleReasonUnapproved,
    WithAccessibleReasonAll,
}
```

DTO model

```
public class ApiDto
{
    public string Id { get; set; }
    public ApiDeprecation? Deprecation { get; set; }
    public string Name { get; set; } = default!;
    public List<ApiDtoField> Fields { get; set; }
    public Reference<ApiDto>? Extends { get; set; }
    public List<Reference<ApiDto>> Implements { get; set; }
    public List<Reference<ApiDto>> Inheritors { get; set; }
    public HierarchyRole HierarchyRole { get; set; }
    public bool Record { get; set; }
}
```

```
public class ApiField
{
    public string Name { get; set; }
    public ApiFieldType Type { get; set; }
    public bool Optional { get; set; }
    public ApiDefaultValue? DefaultValue { get; set; }
}
```


Custom code generation

demo



Roslyn/StringBuilder/...?

Generally, a few ways to generate code:

IL-based

IL Generation (Reflection.Emit)

Model-based

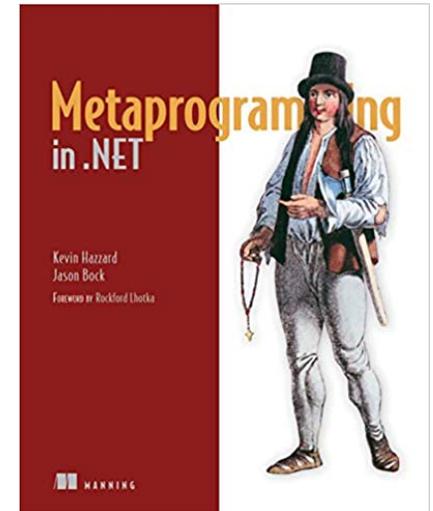
CodeDOM

Expression trees

Roslyn syntax generation

Template-based (T4, Razor, ...)

Strings all the way



<https://amzn.to/2EWsA44>

Strings all the way!

See string, recognize the output easily

IL-based – Too complex

Model-based – Less readable (very verbose)

Template-based – Less readable (many small files to include)

```
builder.AppendLine(  
    $"public {subject.Type.ToCSharpType()} {subject.ToCSharpPropertyName()} {{ get; set; }}");
```

```
var propertyDeclaration = SyntaxFactory.PropertyDeclaration(  
    SyntaxFactory.ParseTypeName(subject.Type.ToCSharpType()), subject.ToCSharpPropertyName())  
    .AddModifiers(SyntaxFactory.Token(SyntaxKind.PublicKeyword))  
    .AddAccessorListAccessors(  
        SyntaxFactory.AccessorDeclaration(SyntaxKind.GetAccessorDeclaration)  
            .WithSemicolonToken(SyntaxFactory.Token(SyntaxKind.SemicolonToken)),  
        SyntaxFactory.AccessorDeclaration(SyntaxKind.SetAccessorDeclaration)  
            .WithSemicolonToken(SyntaxFactory.Token(SyntaxKind.SemicolonToken)));
```

Extension methods everywhere

Many places where we need:

C# class name

C# variable name

C# type from Space model type

...

```
public static class ApiDocumentationExtensions
{
    public static string ToCSharpDocumentationComment(
        this string subject)
    {
        var builder = new StringBuilder();
        builder.AppendLine("/// <summary>");
        builder.AppendLine("/// " + subject);
        builder.AppendLine("/// </summary>");
        return builder.ToString();
    }
}

public static class ApiDtoExtensions
{
    public static string ToCSharpClassName(this ApiDto subject)
        => CSharpIdentifier.ForClassOrNamespace(subject.Name);
}
```

System.Text.Json

"System.Text.Json is a built-in (de)serializer for .NET Core 3 and beyond."

Developer experience!

- No extra dependencies needed

- No dependency on a Newtonsoft.Json that you are not using

System.Text.Json

Not a 100% replacement for Newtonsoft.Json...

<https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-migrate-from-newtonsoft-how-to>

Polymorphic serialization

 Not supported, workaround, sample

Polymorphic deserialization

 Not supported, workaround, sample

Deserialize JSON `null` literal to non-nullable value types

 Not supported, workaround, sample

Deserialize to immutable classes and structs

 Not supported, workaround, sample

System.Text.Json is extensible enough

Custom `JsonConverter<T>`

JsonConverter<T>

demo

<https://blog.maartenballiauw.be/post/2020/01/29/deserializing-json-into-polymorphic-classes-with-systemtextjson.html>





Code generation & developer experience

The ideal “Getting started”

1. [Register application](#) in Space
2. Install SpaceDotNet.Client package

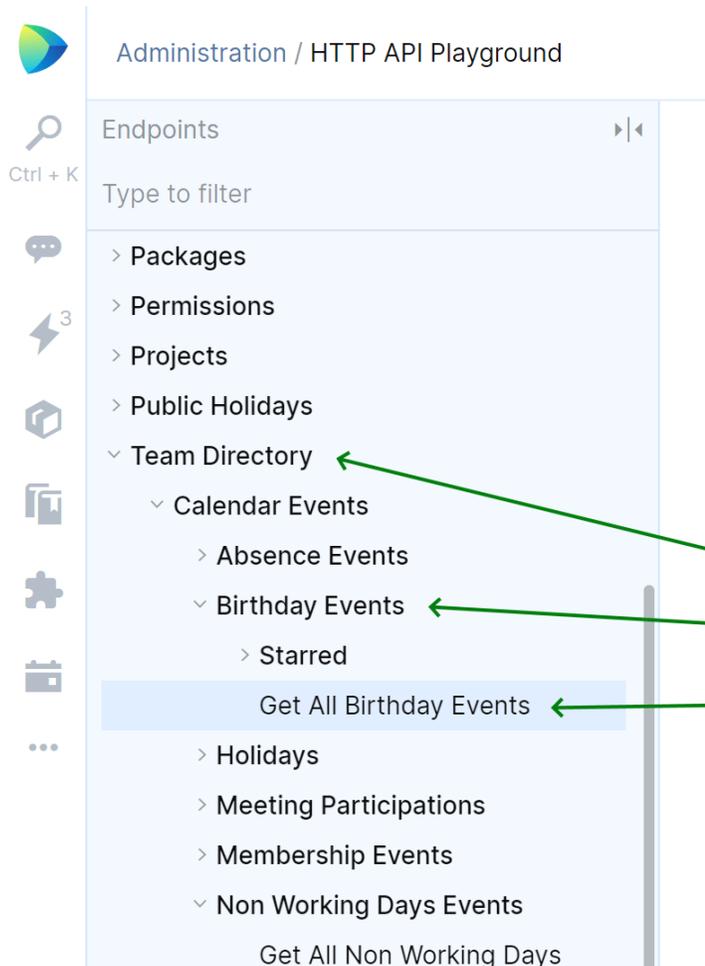
```
dotnet add package SpaceDotNet.Client
```

3. Create connection instance

```
var connection = new ClientCredentialsConnection(  
    "https://your.jetbrains.space/",  
    "client-id",  
    "client-secret",  
    new HttpClient());
```

4. Profit!

Large API surface



```
var teamDirectoryClient = new TeamDirectoryClient(connection);  
var birthdays = await teamDirectoryClient  
    .CalendarEvents  
    .BirthdayEvents  
    .GetAllBirthdayEventsAsync(...);
```

Top level - <...>Client

Property on service client

Method on service client

Remember \$fields ?

Top-level fields returned by default

Unless \$fields query parameter used

Use \$fields to be specific, or get nested fields

```
PR_Project {  
   id: string  
   adminProfiles: [TD_MemberProfile {  +  
   adminTeams: [TD_Team {  +  
   archived: boolean  
   boards: [BoardRecord {  -  
     id: string  
     archived: boolean  
     info: BoardInfo {  -  
       columns: BoardColumns {  -  
         columns: [BoardColumn {  -  
           default: boolean?  
           name: string  
           statuses: [IssueStatus {  +  
          
        
      
  }  
   description: string?  
   owners: BoardOwners {  +  
}  
 name: string  
}]  
 description: string?
```

\$fields

Top-level fields returned by default

Use partial to be specific, or get nested fields

```
var memberProfile = await teamDirectoryClient.Profiles
    .GetProfileAsync(ProfileIdentifier.Username("Heather.Stewart"));
```

```
var memberProfile = await teamDirectoryClient.Profiles
    .GetProfileAsync(ProfileIdentifier.Username("Heather.Stewart"), _ => _
        .WithAllFieldsWildcard()           // with all top level fields
        .WithManagers(managers => managers // include managers
            .WithId()                       // with their Id
            .WithUsername()                 // and their Username
            .WithName(name => name          // and their Name
                .WithFirstName()           // with FirstName
                .WithLastName())));        // and LastName
```

Developer experience

demo



Batches (e.g. Get all ToDo items)

```
public class Batch<T>
{
    List<T>? Data { get; }
    string? Next { get; }
    int? TotalCount { get; }

    bool HasNext();
}
```

```
var batch = await _todoClient.GetAllToDoItemsAsync(
    from: weekStart.AsSpaceDate(),
    partial: _ => _
        .WithData(data => data
            // ...
        .WithTotalCount()
        .WithNext());
```

```
do
{
    foreach (var todo in batch.Data)
    {
        // ...
    }
}
```

```
batch = await _todoClient.GetAllToDoItemsAsync(
    from: weekStart.AsSpaceDate(),
    skip: batch.Next,
    partial: _ => _ /* ... */);
}
while (batch.HasNext());
```

IAsyncEnumerable to the rescue!

```
await foreach (var todo in _todoClient
    .GetAllToDoItemsAsyncEnumerable(from: weekStart.AsSpaceDate(), partial: _ => _
    .WithId()
    .WithContent(content => content
    .ForInherited<ToDoItemContentMdText>(md => md
    .WithAllFieldsWildcard()))
    .WithStatus()))
{
    // ...
}
```

 Don't use it for .Count()

```
var batch = await _todoClient.GetAllToDoItemsAsync(
    from: weekStart.AsSpaceDate(), partial: _ => _.WithTotalCount());
var numberOfResults = batch.TotalCount;
```

Batches and IEnumerable

demo



Request bodies are flattened

~~GetAbsences(AbsencesRequest request)~~

GetAbsences(string member, string reason, ...)

How? OpenAPI?

Not very developer friendly...

```
public class Body
{
    public string Member { get; set; }
    public string Reason { get; set; }
    public string Description { get; set; }
    public string Location { get; set; }
    public string Since { get; set; }
    public string Till { get; set; }
    public bool Available { get; set; } = false;
    public string Icon { get; set; }
}

public async Task<AbsenceRecord> GetAbsencesAsync(
    string fields, Body body, CancellationToken cancellationToken)
{
}
}
```

Factory methods for inheritors

Which option would you prefer?

```
await chatClient.Messages.SendMessageAsync(  
    recipient: new MessageRecipientChannel()  
    {  
        Channel = new ChatChannelFromName()  
        {  
            Name = chatChannelName  
        }  
    },  
);
```

```
await chatClient.Messages.SendMessageAsync(  
    recipient: MessageRecipient.Channel(ChatChannel.FromName(chatChannelName)),  
);
```

Default values

Unfortunately not possible in C#...

```
public async IEnumerable<AbsenceRecord> GetAllAbsencesAsyncEnumerable(  
    AbsenceListMode viewMode = AbsenceListMode.All, string? member = null)
```

Workaround:

```
public IEnumerable<AbsenceRecord> GetAllAbsencesAsyncEnumerable(  
    AbsenceListMode? viewMode = null, string? member = null)  
    => BatchEnumerator.AllItems(  
        () => GetAllAbsencesAsync(viewMode: viewMode ?? AbsenceListMode.All, member: member);
```

Unfortunately not obvious what the default is when not specified.
Should we add XMLdoc instead?

Code generation

demo





Developer experience (extras)

Use SpaceDotNet in web apps

```
public void ConfigureServices(IServiceCollection services)
{
    //...

    // Space authentication
    services.AddAuthentication(options =>
    {
        options.DefaultAuthenticateScheme = CookieAuthenticationDefaults.AuthenticationScheme;
        options.DefaultSignInScheme = CookieAuthenticationDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = SpaceDefaults.AuthenticationScheme;
    })
    .AddCookie()
    .AddSpace(options => Configuration.Bind("Space", options))
    .AddSpaceTokenManagement(); // auto-refreshes tokens in the background and provides a Space connection

    // Space client API
    services.AddSpaceClientApi();

    // Space webhook receiver
    services.AddSpaceWebHookHandler<CateringWebHookHandler>(options => Configuration.Bind("Space", options));
}
```

Use SpaceDotNet in web apps

.AddSpace()

Use Space for authN

.AddSpaceClientApi()

Register ...Client with service collection

.AddSpaceTokenManagement() (experimental)

Auto-refreshes tokens in the background

Space client always auth'ed as currently logged in user

.AddSpaceWebHookHandler<T>() (experimental)

Webhook support

Use SpaceDotNet in web apps

demo



A wide-angle view of Earth from space, showing the curvature of the planet and the atmosphere. The sun is rising over the horizon on the left side, creating a bright orange and yellow glow that illuminates the scene. The Earth's surface is visible, showing landmasses and oceans. The atmosphere is a thin blue line along the horizon. The background is a dark, starry space.

Conclusion

Conclusion

JetBrains Space is an Integrated Team Environment

JetBrains Space HTTP API exposes *everything*

Building a .NET SDK for the Space HTTP API

Goal: make our users succesful at using Space

Provide a ready-to-use SDK

Minimal configuration needed

Discoverability

Code generation

Focus on developer experience



Thank you!

<https://blog.maartenballiau.be>

@maartenballiau