

Writing REST APIs with OpenAPI and Swagger Ada

Stéphane Carrez

FOSDEM 2018

OpenAPI and Swagger Ada

- Introduction to OpenAPI and Swagger
- Writing a REST Ada client
- Writing a REST Ada server
- Handling security with OAuth2
- Demo

30 years of RPC

- Sun RPC (RFC 1057) in 1988
- CORBA IDL in 1991
- Ada95 Distributed Annex E in 1995
- Java RMI in 2000
- WSDL and SOAP in 2000
- Google gRPC with Protocol Buffers since 2001

30 years but same goals

- Simplify the developer's job
- Describe the protocol between client & server
- Generate client stubs and server skeleton
- Handle and hide communication details
- Document the client & server interaction

Why REST and OpenAPI?

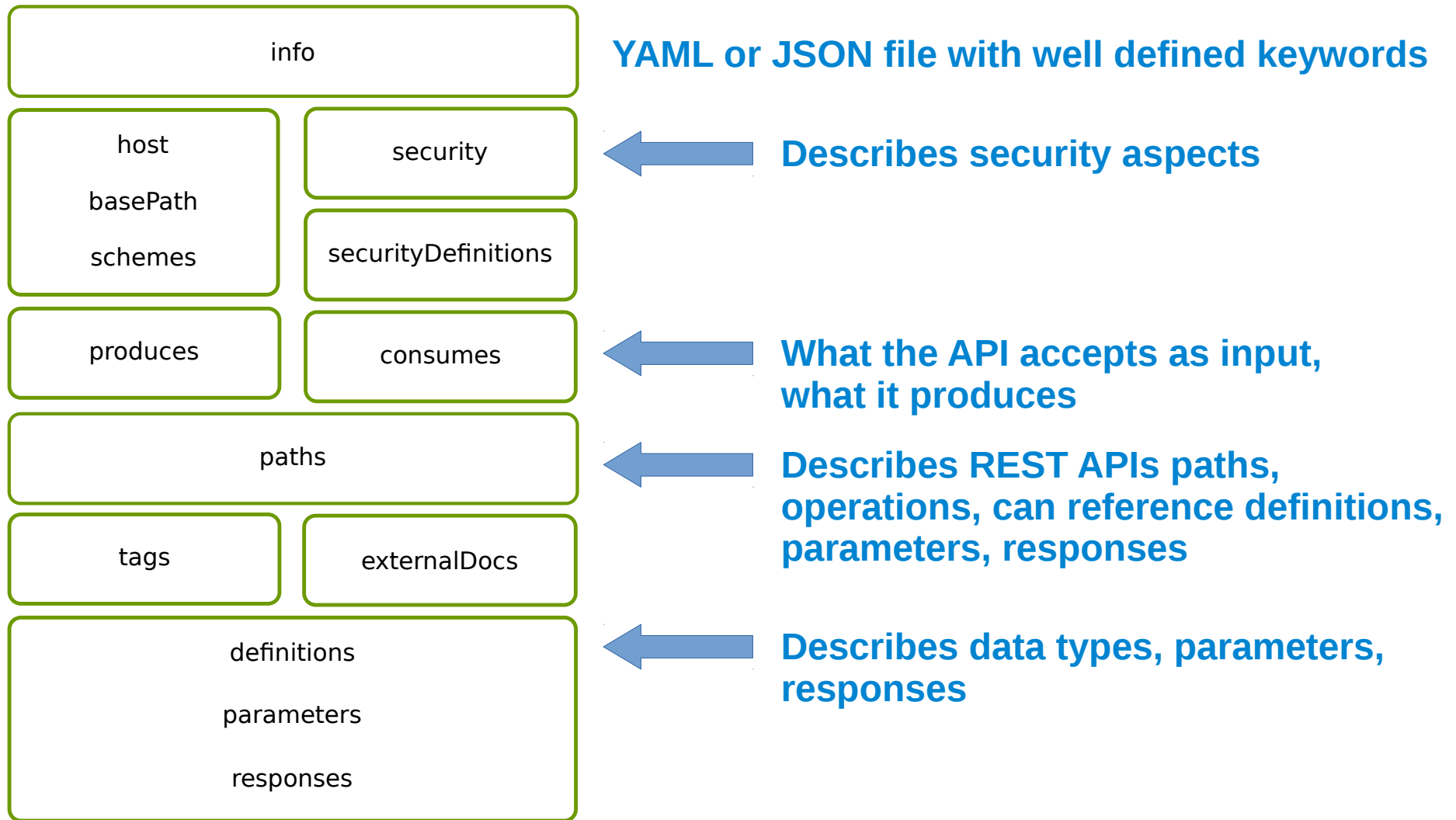
- REST as an alternative to SOAP since 2000 (Roy Thomas Fielding)
- Easier to use, write, implement, debug
- Can easily be used from browsers
- Increasing usage of REST with mobile applications
- Need for description, documentation
- Need for client language bindings

OpenAPI Specification

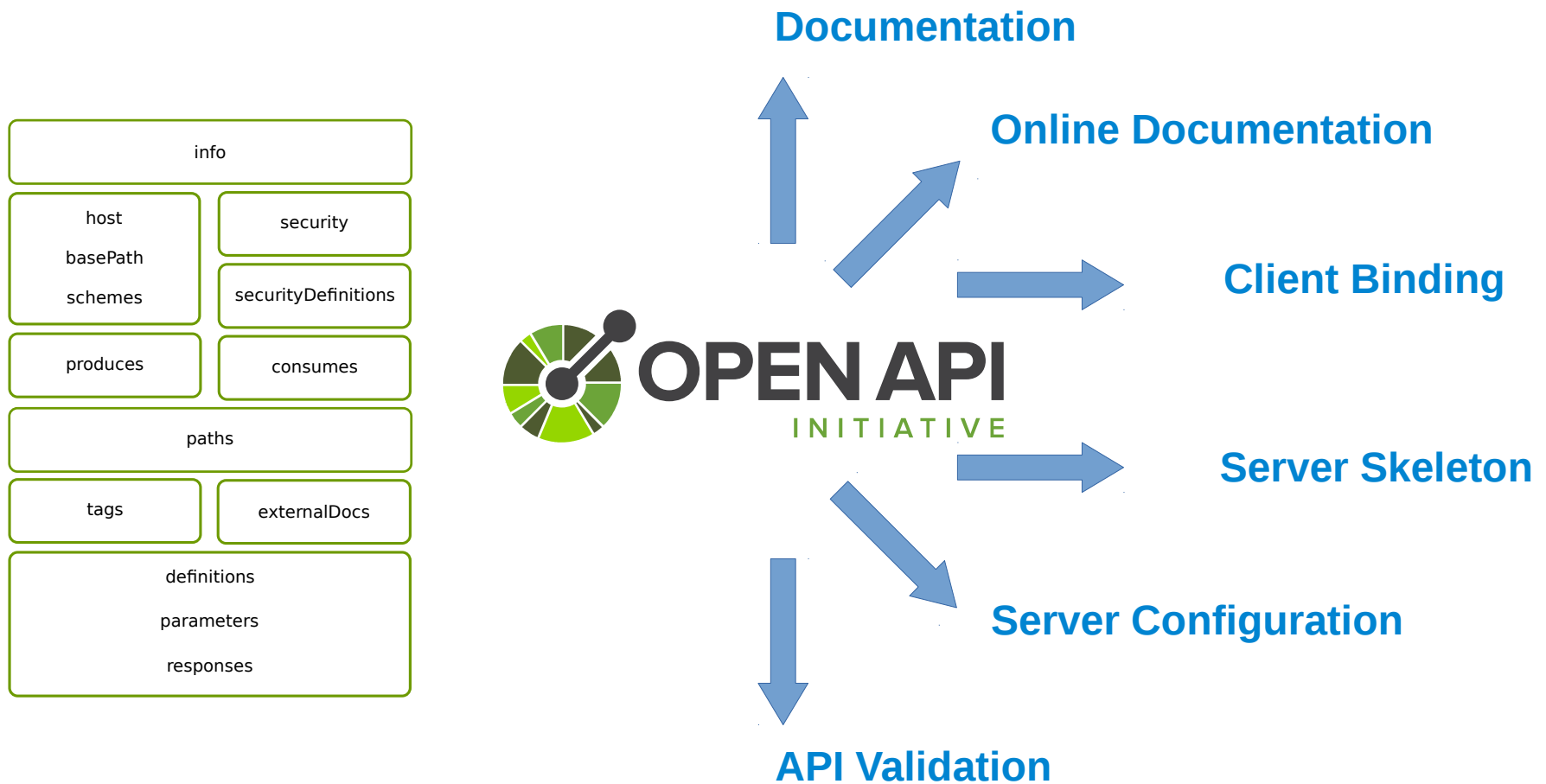
- Started in 2010 to describe REST APIs
- OpenAPI Initiative created in Nov 5, 2015 (Google, Microsoft, IBM, Paypal, ...)
- OpenAPI 3.0 released July 26, 2017
- <https://github.com/OAI/OpenAPI-Specification>



OpenAPI 2.0 Document Structure



OpenAPI benefits



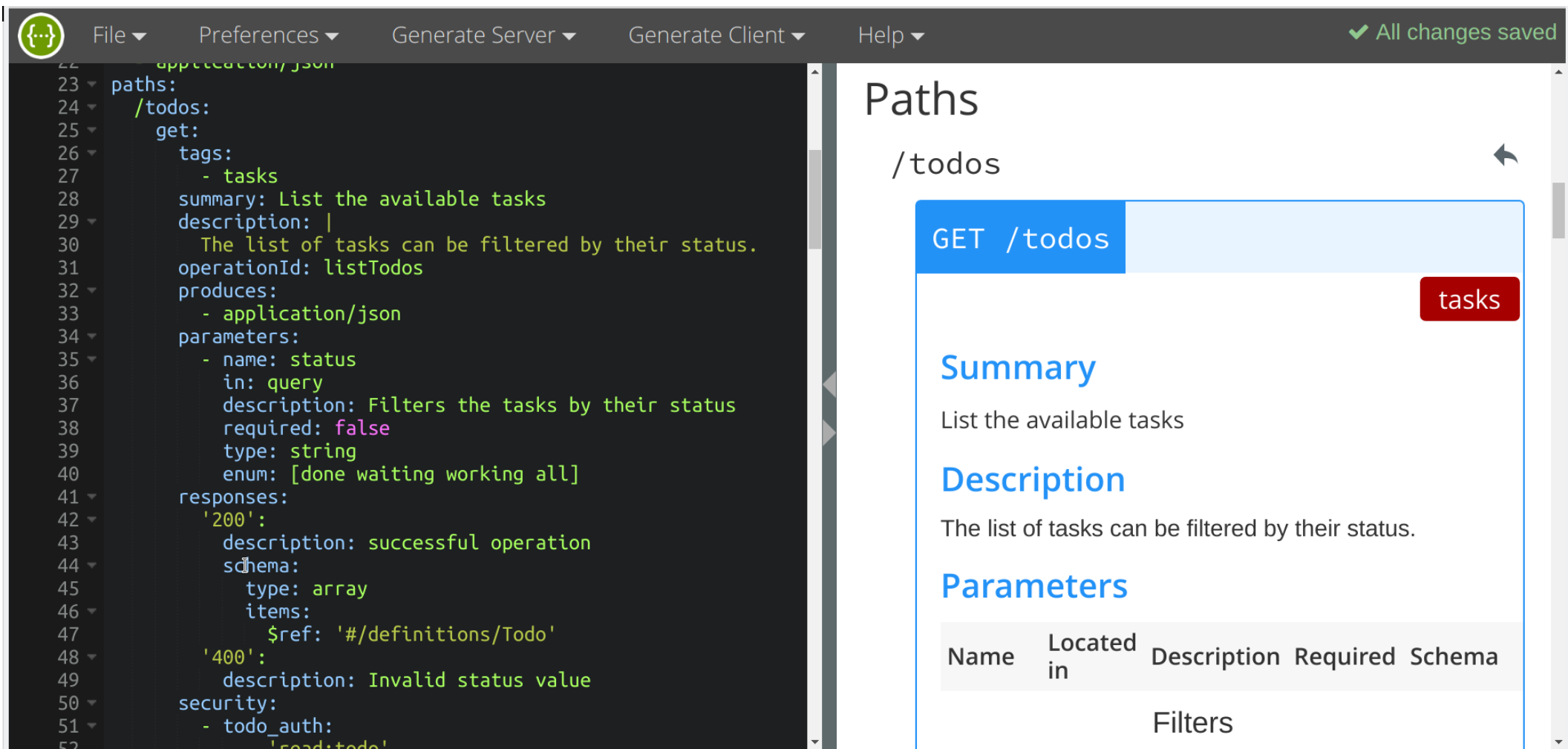
Swagger: Tools for OpenAPI

- Online Editor: Swagger Editor
- Generator: Swagger Codegen
- Documentation: Swagger UI
- Sources: <https://github.com/swagger-api>



Swagger Editor:

<https://editor.swagger.io>



The image shows the Swagger Editor interface. On the left, a dark-themed code editor displays a YAML definition for a REST API. The definition includes a path `/todos` with a `GET` method. The `tags` array contains `tasks`. The `summary` is `List the available tasks`, and the `description` is `The list of tasks can be filtered by their status.` The `operationId` is `listTodos`. The `produces` array contains `application/json`. The `parameters` array contains a query parameter `status` with a description `Filters the tasks by their status`, `required: false`, `type: string`, and an `enum` of `[done waiting working all]`. The `responses` object has two entries: `200` with a description `successful operation` and a schema of `array` of `Todo` objects, and `400` with a description `Invalid status value`. The `security` array contains `todo_auth`.

On the right, the visual representation of the API is shown. It displays the `GET /todos` endpoint with a `tasks` tag. The `Summary` section shows `List the available tasks`. The `Description` section shows `The list of tasks can be filtered by their status.` The `Parameters` section shows a table with columns `Name`, `Located in`, `Description`, `Required`, and `Schema`. The table contains one row for the `status` parameter, which is a query parameter, not required, and has a schema of `string` with an enum of `[done waiting working all]`.

```
22 application/json
23 paths:
24   /todos:
25     get:
26       tags:
27         - tasks
28       summary: List the available tasks
29       description: |
30         The list of tasks can be filtered by their status.
31       operationId: listTodos
32       produces:
33         - application/json
34       parameters:
35         - name: status
36           in: query
37           description: Filters the tasks by their status
38           required: false
39           type: string
40           enum: [done waiting working all]
41       responses:
42         '200':
43           description: successful operation
44           schema:
45             type: array
46             items:
47               $ref: '#/definitions/Todo'
48         '400':
49           description: Invalid status value
50       security:
51         - todo_auth:
52           'read:todo'
```

Paths

/todos

GET /todos

tasks

Summary

List the available tasks

Description

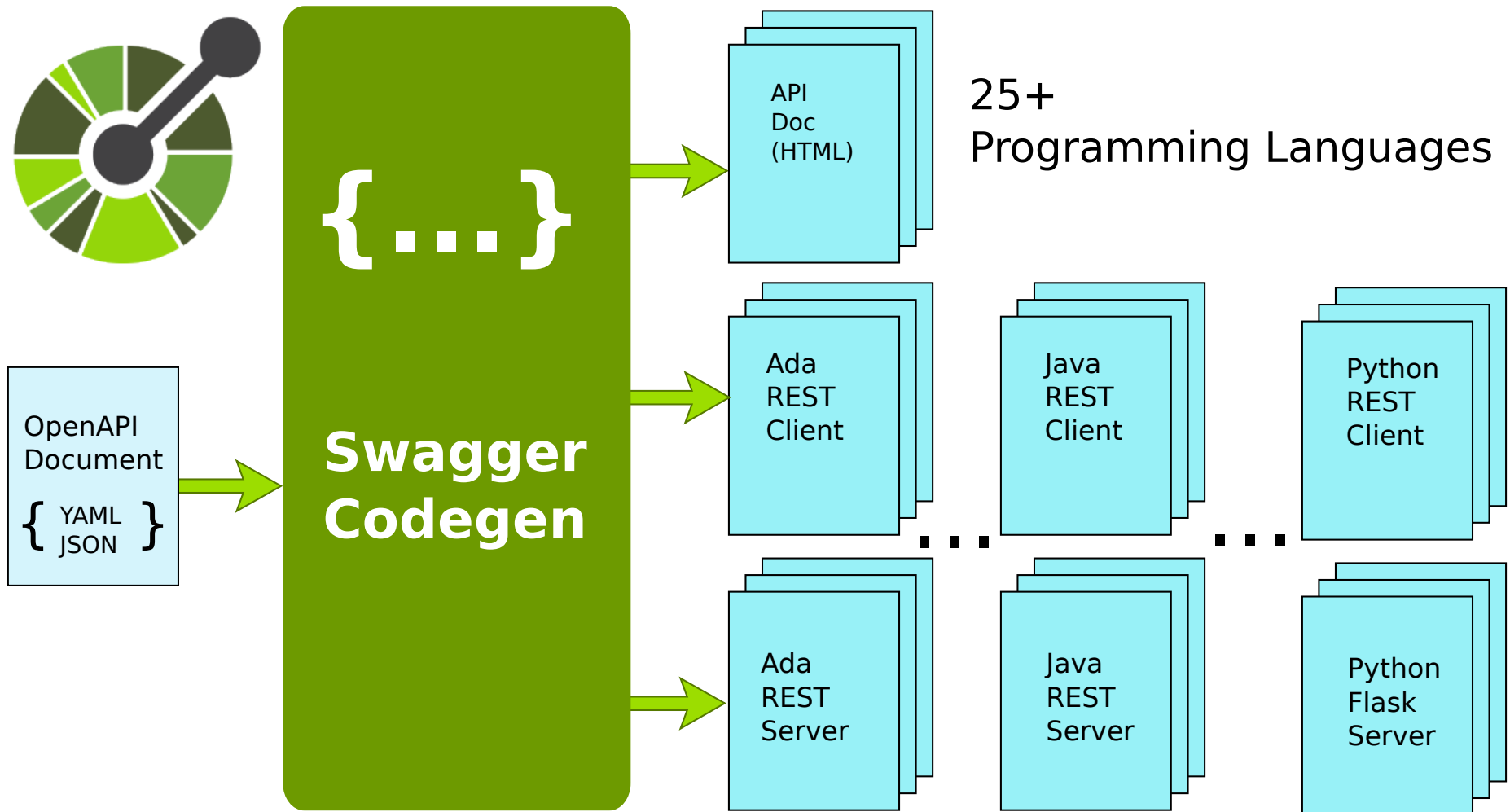
The list of tasks can be filtered by their status.

Parameters

Name	Located in	Description	Required	Schema
status	query	Filters the tasks by their status	False	string [done waiting working all]

Filters

Swagger Codegen



Writing a REST Ada client

- Get the OpenAPI/Swagger description file
 - From SwaggerHub: <https://swaggerhub.com/>
 - From APIs.guru: <https://apis.guru/openapi-directory/>
- Generate the Ada client code
- Use the generated code to make API calls

OpenAPI: Info description (1/3)

- YAML or JSON file
- General purpose description of the API
- Describe the service entry point

```
swagger: "2.0"
info:
  version: "1.0"
  title: "Todo API"
  contact:
    email: Stephane.Carrez@gmail.com
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
host: localhost:8080
basePath: /v1
tags:
  - name: tasks
    description: Operations to manage tasks
schemes:
  - https
  - http
```

OpenAPI: REST operation (2/3)

- Describe the REST operations

```
paths:
  /todos:
    get:
      tags:
        - tasks
      summary: List the available tasks
      description: List the available tasks
      operationId: listTodos
      produces:
        - application/json
      parameters:
        - name: status
          in: query
          description: Filters the task by their status
          required: false
          type: string
          enum:
            - done
            - waiting
            - working
            - all
      responses:
        '200':
          description: successful operation
          schema:
            type: array
            items:
              $ref: '#/definitions/ToDo'
        '400':
          description: Invalid status value
```

OpenAPI: Model definitions (3/3)

definitions:

 Todo:

 type: object

 properties:

 id:

 type: integer

 format: int64

 description: The todo identifier

 title:

 type: string

 description: The todo title

 create_date:

 type: string

 format: date-time

 description: The todo creation date

 done_date:

 type: string

 format: date-time

 description: The todo resolution date

 status:

 type: string

 description: The todo state

 enum:

 - waiting

 - done

 required:

 - id

 - title

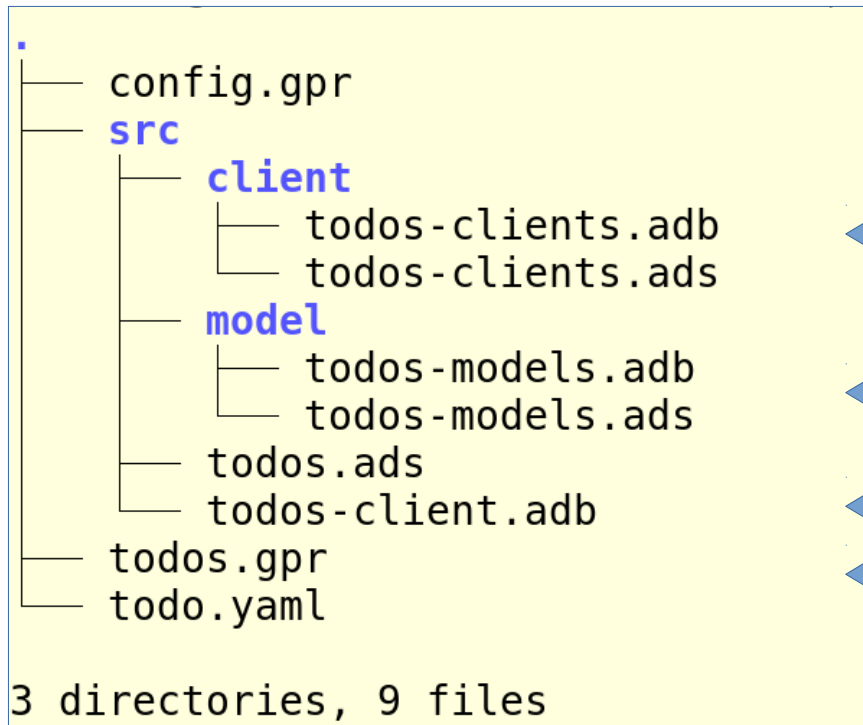
 - status

 - create_date

Client: let's generate the code!

- Generate the client code with Swagger Codegen

```
$ java -jar swagger-codegen-cli.jar generate -l ada -i todo.yaml \
-DprojectName=Todos --model-package Todos
```



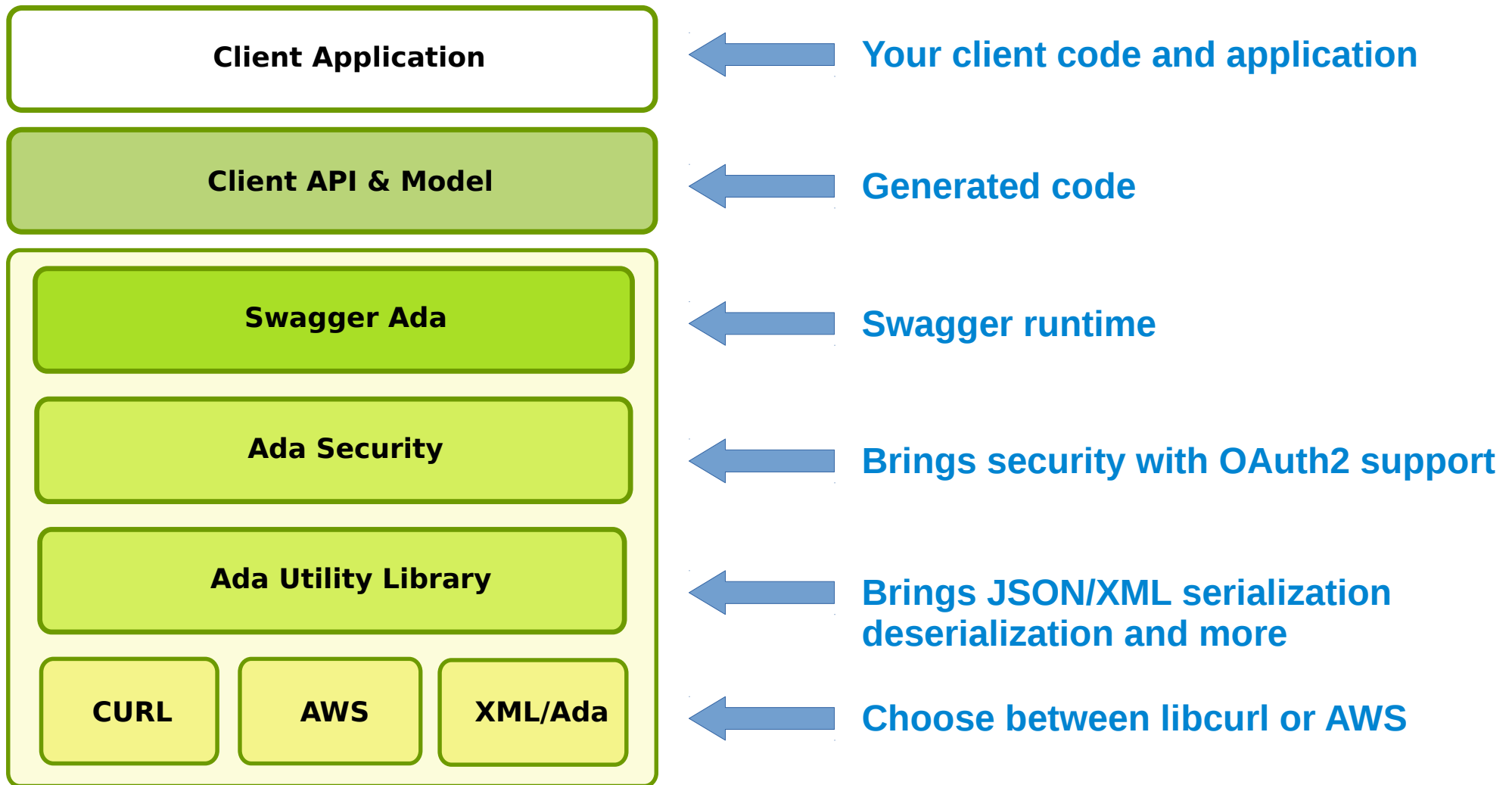
Client API: **package** Todos.Clients

Model API: **package** Todos.Models

Sample: **procedure** Todos.Client

GNAT project

Ada REST Client



Client and Server Data Model

- Data types described in the `Models` package
- Same `Models` Ada package for client and server
- Operations to serialize and deserialize (JSON/XML)

```
package Todos.Models is
  type Todo_Type is record
    Id           : Swagger.Long;
    Title        : Swagger.UString;
    Create_Date  : Swagger.Datetime;
    Done_Date    : Swagger.Nullable_Date;
    Status       : Swagger.UString;
  end record;
  package Todo_Type_Vectors is
    new Ada.Containers.Vectors
      (Positive, Todo_Type);
end Todos.Models;
```

```
Todo:
  type: object
  properties:
    id:
      type: integer
      format: int64
      description: The todo identifier
    title:
      type: string
      description: The todo title
    create_date:
      type: string
      format: date-time
      description: The todo creation date
    done_date:
      type: string
      format: date-time
      description: The todo resolution date
    status:
      type: string
      description: The todo state
    enum:
      - waiting
      - done
```

Client API

- Represented by the `Client_Type` tagged record
- Provides operations described by the OpenAPI
- Allows to control the API call (headers, security)

```
package Todos.Clients is
  type Client_Type is
    new Swagger.Clients.Client_Type with null record;

  procedure Create_Todo (Client : in out Client_Type;
                        Title   : in Swagger.Ustring;
                        Result  : out Todos.Models.TODO_Type);
  procedure List_Todos (Client : in out Client_Type;
                       Status   : in out Swagger.Nullable_UString;
                       Result   : out Todos.Models.TODO_Vector);
end Todos.Clients;
```

Calling REST in Ada

- Declare a `Client_Type` instance
- Configure it (server URL, credentials)
- Call the operation with its parameters

```
with Todos.Clients;  
with Todos.Models;  
...  
Client : Todos.Clients.Client_Type;  
List    : Todos.Models.TODO_Type_Vectors.Vector;  
Empty   : Swagger.Nullable_String := (Is_Null => True, Value => <>);  
...  
Client.Set_Server ("http://localhost:8080/v1");  
Client.List_Todos (Empty, List);
```

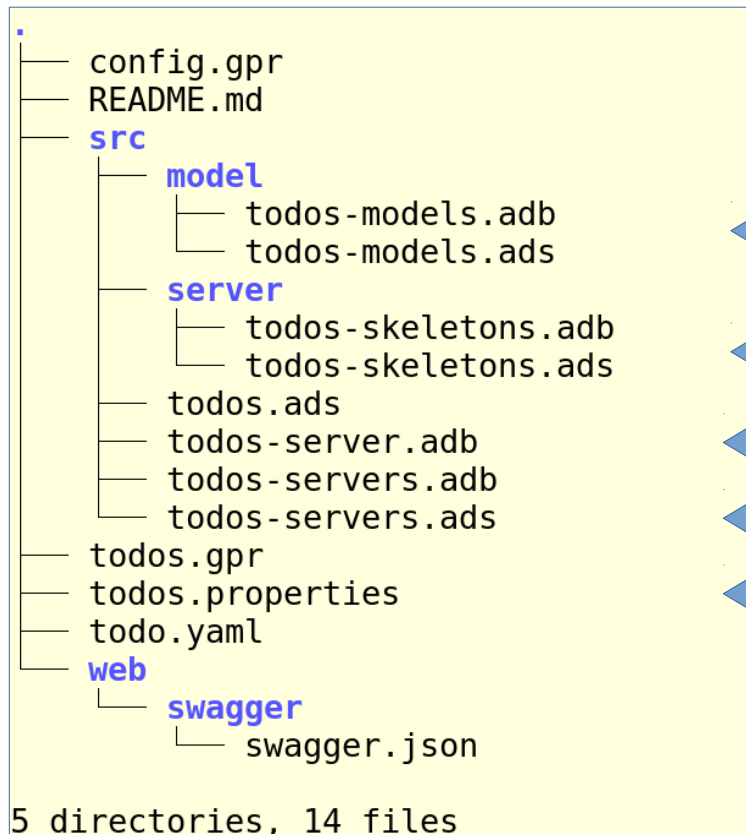
Writing a REST Ada server

- Write the OpenAPI/Swagger description file
- Generate the Ada server code
- Implement the server operations
- Share the OpenAPI description on SwaggerHub!

Server: let's generate the code!

- Generate the server code with Swagger Codegen

```
$ java -jar swagger-codegen-cli.jar generate -l ada-server -i todo.yaml \
-DprojectName=Todos --model-package Todos
```



Model API: **package** Todos.Models

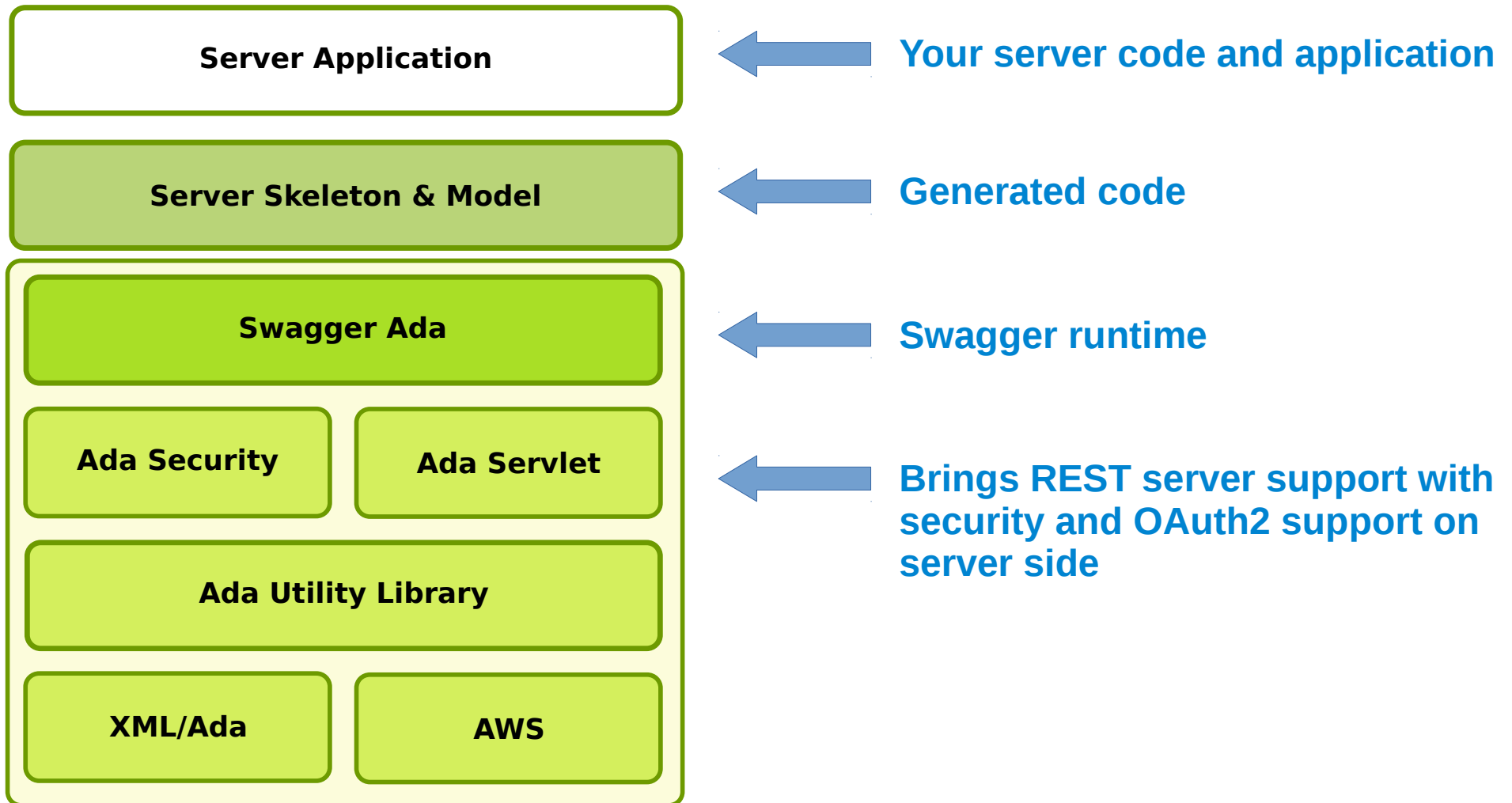
Server skeleton: **package** Todos.Skeletons

Server: **procedure** Todos.Server

Server code: **package** Todos.Servers

GNAT project, server configuration file

Ada REST Server



Server Skeleton

- Declares the `Server_Type` limited interface to describe the operations
- Additional `Context_Type` object gives access to request, response
- Two generic packages for server skeleton provide two server models:
 - Instance per request
 - Global shared instance within a protected object

```
package Todos.Skeletons is
  type Server_Type is limited interface;

  procedure Create_Todo
    (Server  : in out Server_Type;
     Title   : in  Swagger.Ustring;
     Result  : out  Todos.Models.TODO_Type;
     Context : in out Swagger.Servers.Context_Type) is abstract;
  ...
end Todos.Skeletons;
```

Server Implementation (1/2)

- Implement the `Server_Type` interface with its operations
- Populate `Result` or use the `Context` to send an error
- Serialization/Deserialization handled by the skeleton

```
package Todos.Servers is
  type Server_Type is limited new Todos.Skeletons.Server_Type ...

  overriding procedure Create_Todo
    (Server   : in out Server_Type;
     Title    : in  Swagger.Ustring;
     Result   : out  Todos.Models.TODO_Type;
     Context  : in out Swagger.Servers.Context_Type);
  ...
end Todos.Servers;
```

Server Implementation (2/2)

- Instantiate one of the two server skeletons (per-request model or shared model)

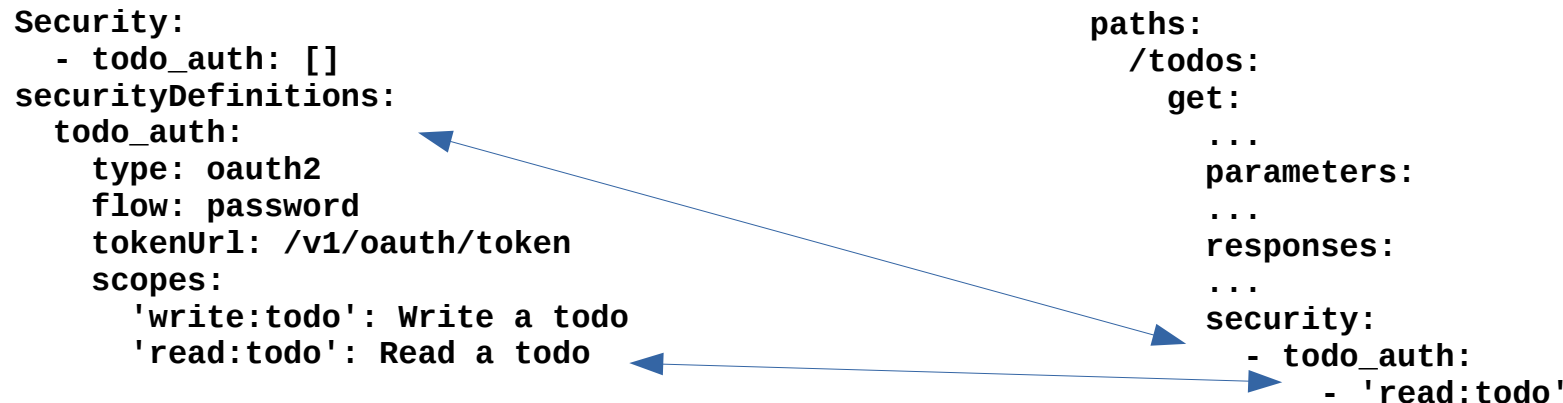
```
package Todos.Servers is
    ...
    package Server_Impl is
        new Todos.Skeletons.Shared_Instance (Server_Type);
    end Todos.Servers;
```

- Register the OpenAPI to the application

```
procedure Todos.Server is
    App : aliased Swagger.Servers.Applications.Application_Type;
begin
    . . .
    Todos.Servers.Server_Impl.Register (App);
    . . .
end Todos.Server;
```

OpenAPI: Describing security

- Describe security endpoints
- Describe security scopes
- Assign required security scopes to operations



Client Security with OAuth2 (1/2)

- Create and initialize a credentials object
- Obtain the access token and optional refresh token
- Configure the client to use the credentials

```
with Swagger.Credentials.OAuth;
```

```
Cred : aliased Swagger.Credentials.OAuth.OAuth2_Credential_Type;
```

```
...
```

```
  Cred.Set_Application_Identifier ("todo-app");
```

```
  Cred.Set_Application_Secret ("todo-app-secret");
```

```
  Cred.Set_Provider_URI ("http://localhost:8080/v1/oauth/token");
```

```
  Cred.Request_Token (Username, Password, "read:todo write:todo");
```

```
...
```

```
  Client.Set_Credentials (Cred'Access);
```

Client Security with OAuth2 (2/2)

- Make API calls: credentials are passed on the wire within the 'Authorization' header

```
List : Todos.Models.TODO_Type_Vectors.Vector;  
...  
Client.List_Todos (Empty, List);
```



```
GET /v1/todos HTTP/1.1  
Host: localhost:8080  
Authorization: Bearer 74rE0wU.d44CPA1l_kyyB2krd8bYdVYWqgmtloIR.9zyiBM  
Accept: application/json
```

Server security (1/2)

- Each OpenAPI scope represented by a permission definition (generated code):

```
package ACL_Read_Todo
  is new Security.Permissions.Definition ("read:todo");
```

- Authentication and permission check generated in the server skeleton (generated code):

```
if not Context.Is_Authenticated then
  Context.Set_Error (401, "Not authenticated");
  return;
end if;
if not Context.Has_Permission (ACL_Read_Todo.Permission) then
  Context.Set_Error (403, "Permission denied");
  return;
end if;
```

Server security (2/2)

- Configure the server key for OAuth2 tokens:

```
swagger.key=Y29naGk5SGkKUG9YaTdhaHgKYWlUaG1lM3UK
```

- Configure the server to register the client id and secret

```
app.list=1  
app.1.client_id=todo-app  
app.1.client_secret=todo-app-secret  
app.1.scope=none
```

- Configure the users allowed to authenticate

```
users.list=1,2  
users.1.username=admin  
users.1.password=admin  
users.2.username=test  
users.2.password=test
```

Demo: Todo client

```
GPS - todos-client.adb - /home/cicaron/work/vacs/ada-fosdem-2018/swagger-ada-todo/src/ - todos project
File Edit Navigate VCS Project Build Debug Tools Window Help

Project Outline Scenario
  AWS
  Config
  El
  Security
  Servlet
  Servlet_Aws
  Servlet_Core
  swagger
  swagger_server
  todos (root project)
    src
      todos-client.adb
      todos-server.adb
      todos-servers.adb
      todos-servers.ads
      todos.ads
      src/client
      src/model
      src/server
      bin
  Util
  Util_Http
  Util_Http_Aws
  Util_Http_Curl
  XmlAda_Dom
  XmlAda_Input
  XmlAda_Sax
  XmlAda_Schema
  XmlAda_Unicode

85   Cred      : aliased Swagger.Credentials.OAuth.OAuth2_Credential_Type;
86   C         : Todos.Clients.Client_Type;
87   Title     : Swagger.UString;
88   Todo      : Todos.Models.Todo_Type;
89   List      : Todos.Models.Todo_Type_Vectors.Vector;
90   begin
91     Configure ("client.properties", C, Cred);
92
93     C.Set_Credentials (Cred'Access);
94     if Arg_Count = 2 and Command = "add" then
95       Title := Swagger.To_UString (Ada.Command_Line.Argument (2));
96       C.Create_Todo (Title, Todo);
97       Put_Line ("Created todo " & Swagger.Long'Image (Todo.Id));
98       Print (Todo);
99
100    elsif Arg_Count = 1 and Command = "list" then
101      C.List_Todos ((Is_Null => True, Value => <>), List);
102      for T of List loop
103        Print (T);
104      end loop;
105
106    elsif Arg_Count = 2 and Command = "del" then
107      C.Delete_Todo (Todo_Id => Swagger.Long'Value (Ada.Command_Line.Argument (2)));
108
109    elsif Arg_Count = 2 and Command = "close" then
110      C.Update_Todo (Todo_Id => Swagger.Long'Value (Ada.Command_Line.Argument (2)),
111                    Title => (Is_Null => True, Value => <>),
112
Todos.Client
Messages Locations
Welcome to GPS 2016 (20160515) hosted on x86_64-pc-linux-gnu
the GNAT Programming Studio
```

Demo: Todo server (main)

```
GPS - todos-server.adb - /home/ciceron/work/vacs/ada-fosdem-2018/swagger-ada-todo/src/ - todos project
File Edit Navigate VCS Project Build Debug Tools Window Help

Project Outline Scenario
  AWS
  Config
  El
  Security
  Servlet
  Servlet_Aws
  Servlet_Core
  swagger
  swagger_server
  todos (root project)
    src
      todos-client.adb
      todos-server.adb
      todos-servers.adb
      todos-servers.ads
      todos.ads
      src/client
      src/model
      src/server
      bin
  Util
  Util_Http
  Util_Http_Aws
  Util_Http_Curl
  XmlAda_Dom
  XmlAda_Input
  XmlAda_Sax
  XmlAda_Schema
  XmlAda_Unicode

9  procedure Configure (Config : in out AWS.Config.Object);
10
11  CONFIG_PATH  : constant String := "todos.properties";
12
13  procedure Configure (Config : in out AWS.Config.Object) is
14  begin
15    AWS.Config.Set.Server_Port (Config, 8080);
16    AWS.Config.Set.Max_Connection (Config, 8);
17    AWS.Config.Set.Accept_Queue_Size (Config, 512);
18  end Configure;
19
20  App      : aliased Swagger.Servers.Applications.Application_Type;
21  WS       : Swagger.Servers.AWS.AWS_Container;
22  Log      : constant Util.Log.Loggers.Logger := Util.Log.Loggers.Create ("Todos.Server");
23  Props    : Util.Properties.Manager;
24  begin
25    Props.Load_Properties (CONFIG_PATH);
26    Util.Log.Loggers.Initialize (Props);
27
28    App.Configure (Props);
29    Todos.Servers.Server_Impl.Register (App);
30
31    WS.Configure (Configure'Access);
32    WS.Register_Application ("/v1", App'Unchecked_Access);
33    App.Dump_Routes (Util.Log.INFO_LEVEL);
34    Log.Info ("Connect you browser to: http://localhost:8080/v1/ui/index.html");
35
Todos.Server
Messages Locations
Welcome to GPS 2016 (20160515) hosted on x86_64-pc-linux-gnu
```

Demo: Todo server (impl)

GPS - todos-servers.adb - /home/cicéron/work/vacs/ada-fosdem-2018/swagger-ada-todo/src/ - todos project

File Edit Navigate VCS Project Build Debug Tools Window Help

Project Outline Scenario

filter

todos-servers.adb todos-client.adb todos-server.adb

AWS
Config
El
Security
Servlet
Servlet_Aws
Servlet_Core
swagger
swagger_server
todos (root project)
src
todos-client.adb
todos-server.adb
todos-servers.adb
todos-servers.ads
todos.ads
src/client
src/model
src/server
bin
Util
Util_Http
Util_Http_Aws
Util_Http_Curl
XmlAda_Dom
XmlAda_Input
XmlAda_Sax
XmlAda_Schema
XmlAda_Unicode

```
14
15 use type Swagger.UString;
16
17 -- Create a todo
18 overriding
19 procedure Create_Todo
20   (Server : in out Server_Type;
21    Title : in Swagger.UString;
22    Result : out Todos.Models.TODO_Type;
23    Context : in out Swagger.Servers.Context_Type) is
24   pragma Unreferenced (Context);
25 begin
26   Result.Id := Server.Next_Id;
27   Result.Create_Date := Ada.Calendar.Clock;
28   Result.Status := Swagger.To_UString ("waiting");
29   Result.Title := Title;
30   Server.Next_Id := Server.Next_Id + 1;
31   Server.Tasks.Insert (Result.Id, Result);
32 end Create_Todo;
33
34 -- Delete the todo
35 -- Supprime le test/travail soit avant son execution, soit après son execution.
36 overriding
37 procedure Delete_Todo
38   (Server : in out Server_Type;
39    Todo_Id : in Swagger.Long;
40    Context : in out Swagger.Servers.Context_Type) is
```

Todos.Models.TODO_Type
global record type declared at todos-models.ads:16

Todos.Servers

Messages Locations

Welcome to GPS 2016 (20160515) hosted on x86_64-pc-linux-gnu

Demo: Running the client

```
swagger-ada-todo : bash
Fichier Édition Affichage Signets Configuration Aide
gprbuild: "todos-client" up to date
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client
Usage: todos {list|add|del|close|update} {params}
  list                List the todos
  add <title>         Add a todo
  del <id>             Delete the todo with the given <id>
  close <id>          Change the todo status to 'close'
  update <id> <title> Update the todo title
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client list
ERROR: get request: Couldn't connect to server
Cannot connect to the server.
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client list
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client add 'FOSDEM 2018 presentation'
Created todo 1
 1 waiting 2018-02-03 08:13:25 - FOSDEM 2018 presentation
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client list
 1 waiting 2018-02-03 08:13:25 - FOSDEM 2018 presentation
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client close 1
 1 done 2018-02-03 08:13:25 2018-02-03 08:13:53 FOSDEM 2018 presentation
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client list
 1 done 2018-02-03 08:13:25 2018-02-03 08:13:53 FOSDEM 2018 presentation
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client close 2
The todo does not exist.
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client list
 1 done 2018-02-03 08:13:25 2018-02-03 08:13:53 FOSDEM 2018 presentation
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client del 2
The todo does not exist.
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client del 1
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client list
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$
```

Server not started

404 error received

Limitations and improvements

- Ada Strings
(need a lot of `To_String`/`To_Ustring` conversions)
- Enums are treated as strings
- Circular type dependencies not handled
- Error model needs some work
- Improvements:
 - Upload file support
 - Asynchronous client operation call

Conclusion

- OpenAPI describes REST APIs
- Swagger Codegen generates Ada code for you
- Swagger Ada is a runtime library for REST client and REST server implementation
- More than 500 APIs available, write your own!
- Next step: ... GraphQL?