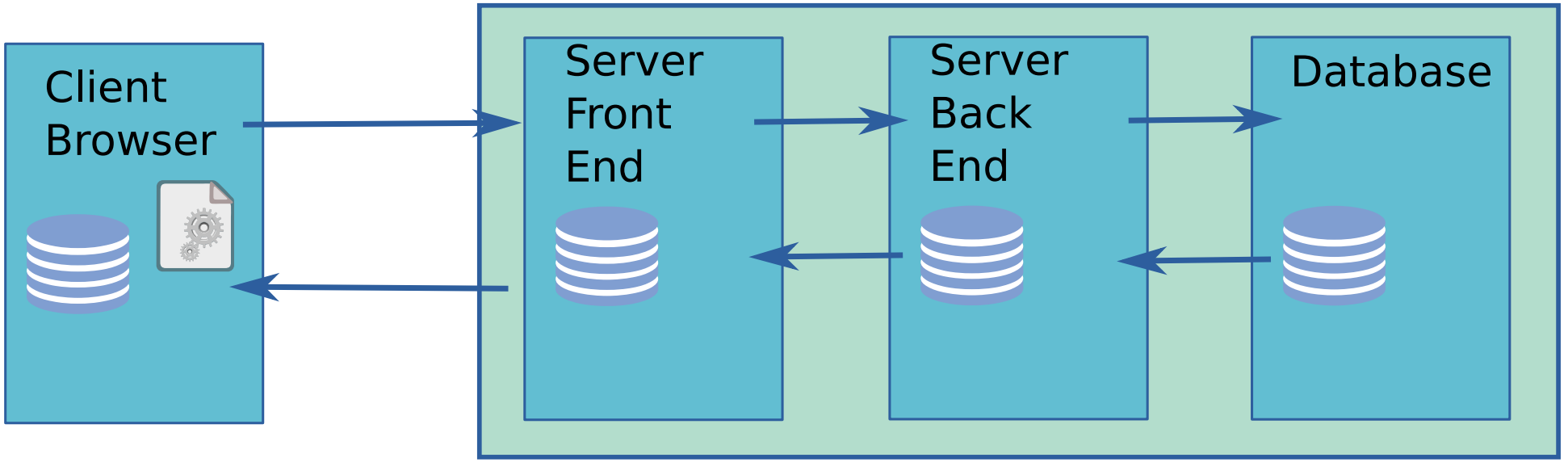
Secure Web Applications with AWA

Stéphane Carrez

FOSDEM 2019

What is a Web Application

- Client server program with browser as client
- Examples: Gmail, Dropbox, Netflix, Zoho,...



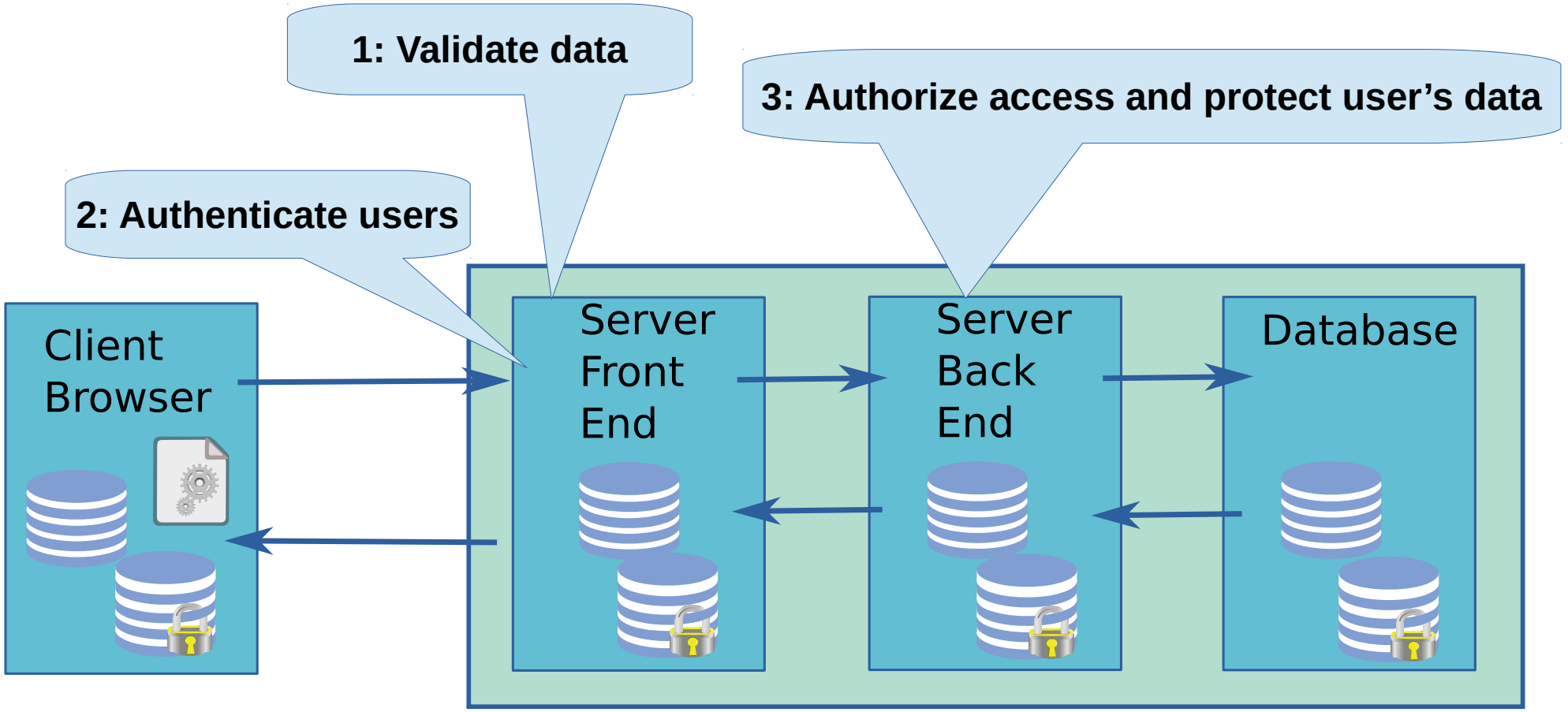
Javascript
HTML, CSS

PHP, Javascript,
Ruby, Java, ...

SQL, NOSQL, ...

Problems with Web Applications

- Must protect data



Project history

- Started in 2011 with already 6 releases
- Based on experience building SaaS application (J2EE, Java Server Faces, Hibernate, OAuth)
- Benefit from several J2EE features but in Ada
- Build SaaS applications in Ada

Applications using AWA

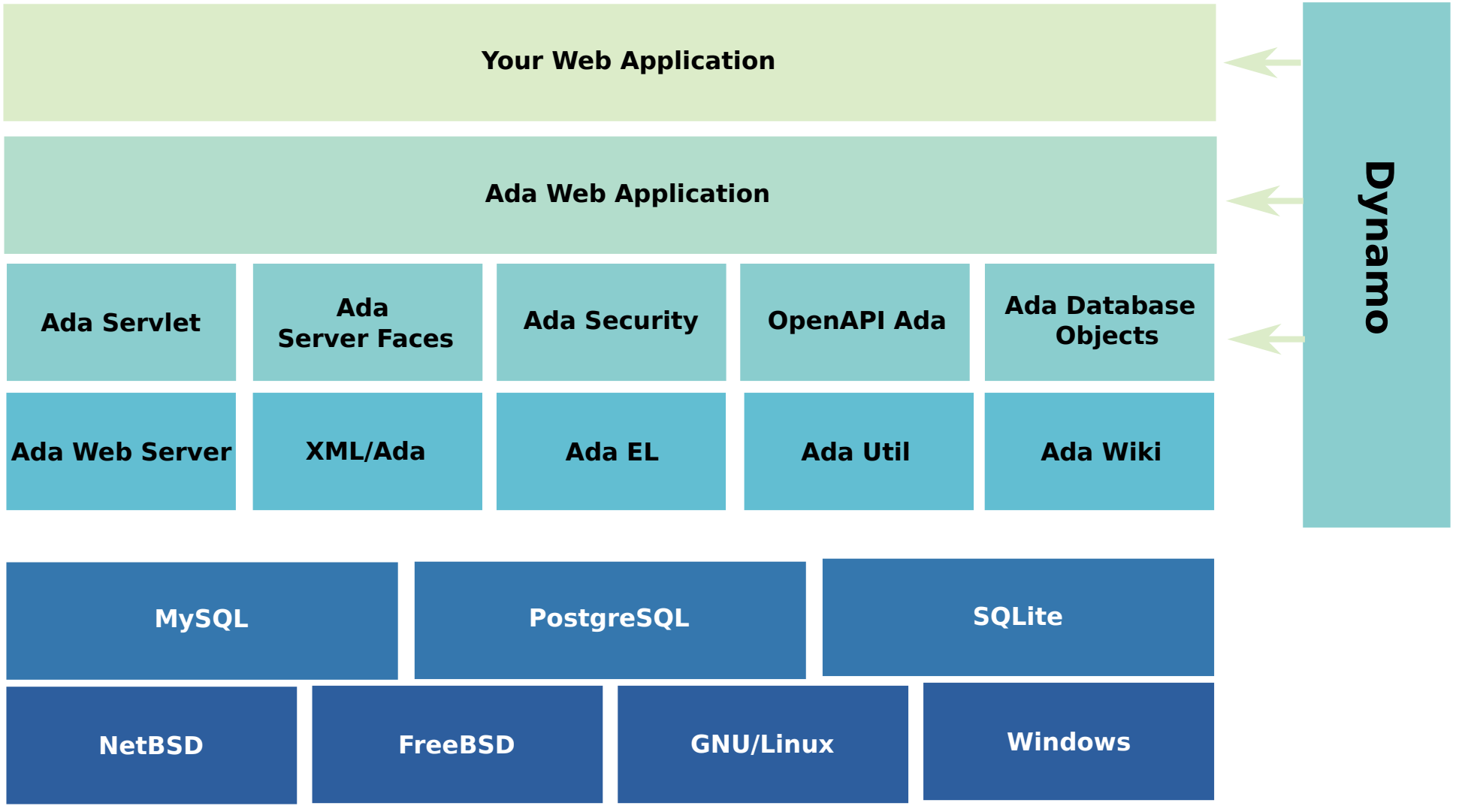
- Personal blog: https://blog.vacs.fr
- Ada France: https://www.ada-france.org

  https://github.com/Ada-France/ada-france
- Atlas demo: https://demo.vacs.fr/atlas

  https://github.com/stcarrez/atlas
- Jason: https://vdo.vacs.fr

  https://github.com/stcarrez/jason

AWA Architecture



https://github.com/stcarrez/ada-awa

AWA Features

Functional components

Wikis | Blogs | Questions | Storages | Images

System components

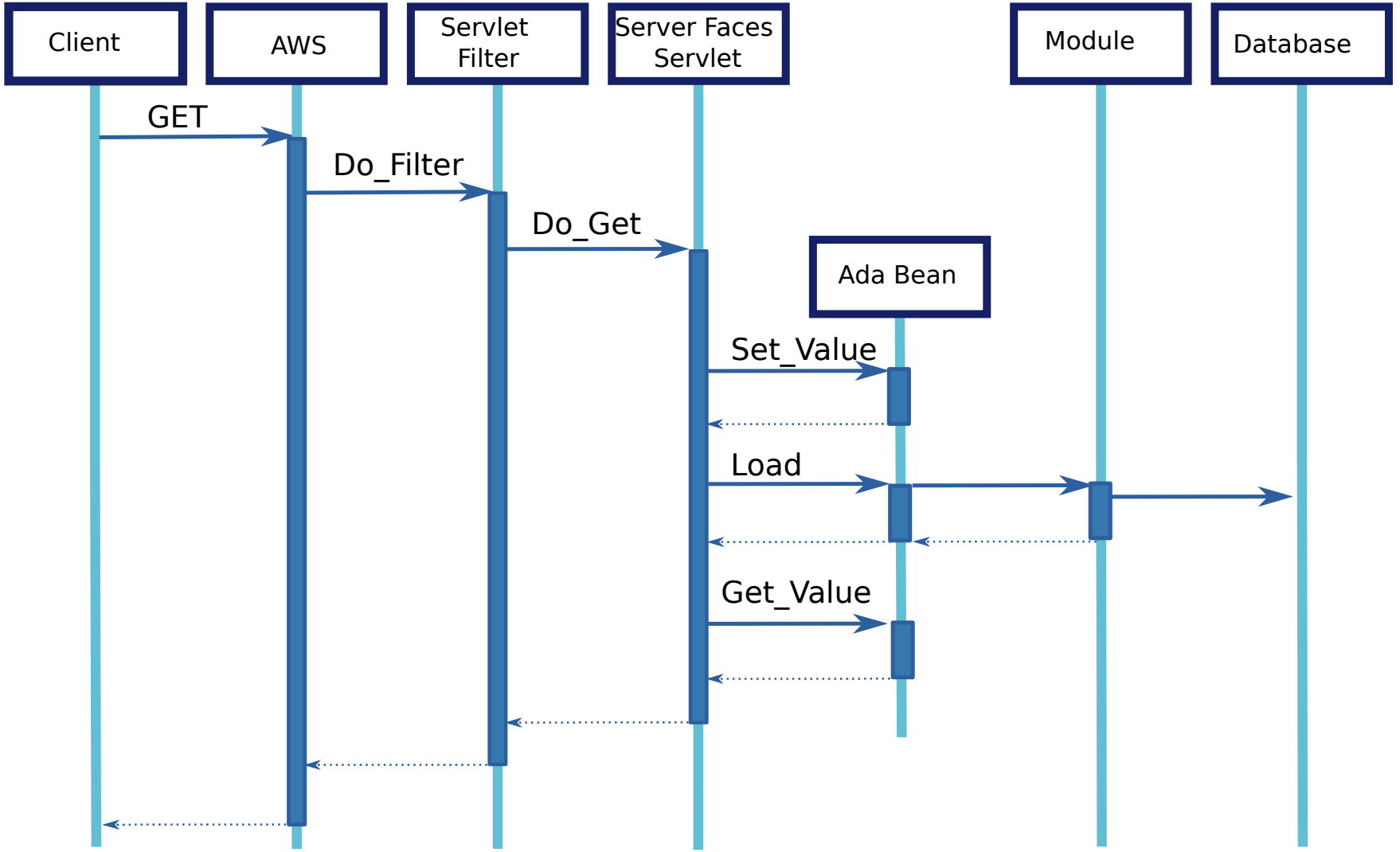
Users | Mails | Workspaces | Permissions | Jobs | Events | Setup

General purpose components

Comments | Counters | Votes | Tags | Settings | Changelogs | Flotcharts | Trumbowyg

https://github.com/stcarrez/ada-awa

AWA Request Flow

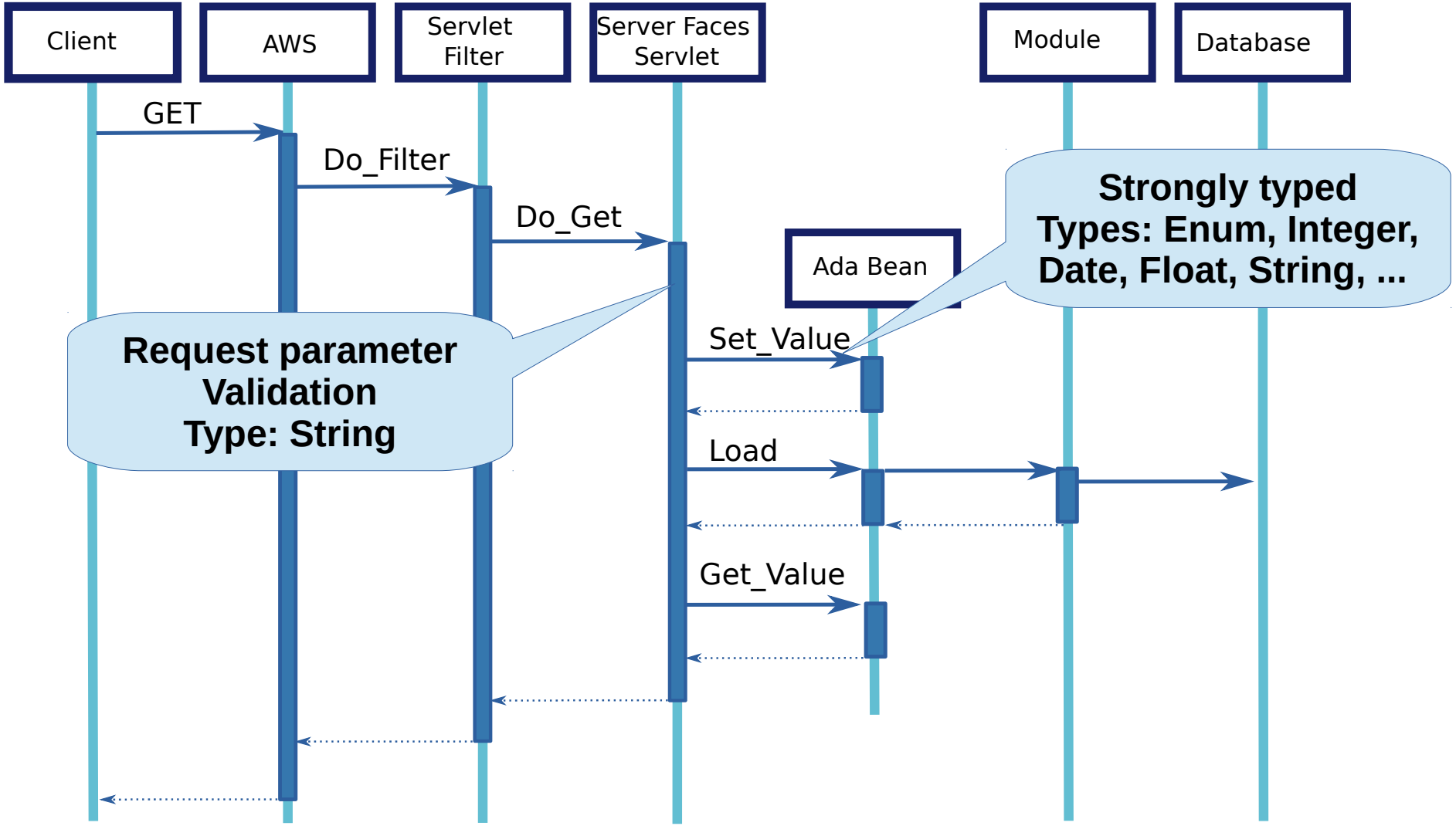


https://github.com/stcarrez/ada-awa

Problem 1: Validate Data

- HTTP parameters are passed as String
- Must be validated, verified before being used
- Ada strong typing helps to enforce the validation

Validation in Request Flow



https://github.com/stcarrez/ada-awa

Ada Server Faces (Java JSR 344)

- MVC web framework
- Render HTML, XML, JSON, Text,…, Ada
- Validate inputs
- Uses XML to describe views

Ada Server Faces

- Facelets: XHTML files with templating
- Component based interface

```
<f:metadata>
  <f:viewParam id='page' value='#{wikiView.name}'/>
  <f:viewAction action='#{wikiView.load}'/>
</f:metadata>
<div>
  <awa:wiki value="#{wikiView.content}"/>
</div>
<div class="wiki-page-footer">
  <h:outputFormat styleClass="wiki-page-date"
                  value="#{wikiMsg.wiki_page_info_date}">
    <f:param value="#{wikiView.date}"/>
    <f:converter converterId="smartDateConverter"/>
  </h:outputFormat>
</div>
```

Operation called before rendering

Custom UI component: render wiki text

Standard UI component with custom format\

Ada EL (Java JSR 245)

- The presentation layer need values from Ada objects
- EL is a simple but powerful expression language
- Java implements EL using introspection
  → security issue

EL expression

```
#{wikiView.title}
```

Ada

```
type Wiki_View_Bean is ...
  Title : Unbounded_String;
  ...
end record;
```

Ada Beans: get and set values

- Get values for the presentation layer (Ada EL)
- Explicit definition: implement the `Bean` interface
- Values represented by `Object` type
  (can hold most Ada types, including Ada Beans)

```
type Object is private;

type Readonly_Bean is limited interface;
function Get_Value (From : in Readonly_Bean;
                    Name : in String) return Object is abstract;

type Bean is limited interface and Readonly_Bean;
procedure Set_Value (From : in out Bean;
                     Name : in String;
                     Value : in Object) is abstract;
```

Ada Beans: method calls

- Declare a table of supported operations
- Implement the `Method_Bean` interface

```
type Method_Bean is limited interface;
function Get_Methods (From : in Method_Bean)
   return Method_Binding_Array_Access is abstract;
```

- Let Dynamo generate the code

```
procedure Op_Load (Bean    : in out Wiki_Page_Bean;
                   Outcome : in out Unbounded_String);

package Binding_Wiki_Page_Bean_3 is
  new ASF.Events.Faces.Actions.Action_Method.Bind
     (Bean => Wiki_Page_Bean, Method => Op_Load, Name => "load");
```

Ada Beans: factory

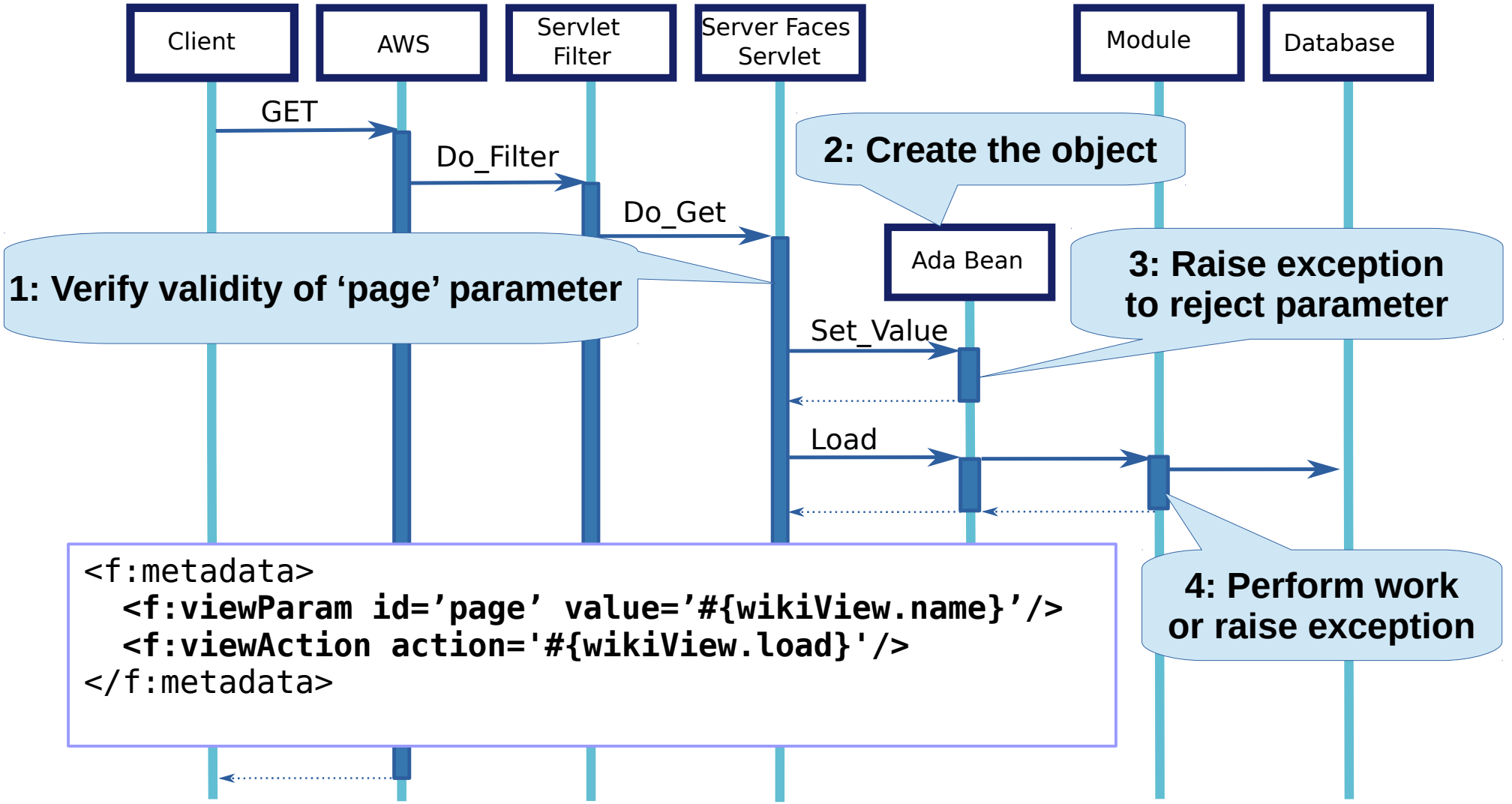
- Need creation of Ada Beans for a Web request
- Write function to create the Ada bean instance
- Register the function under a name
- Use XML configuration to declare bean names

```
function Create_Wiki_View_Bean
 return Util.Beans.Basic.Readonly_Bean_Access;

Register.Register
 (Plugin => Plugin,
  Name   => "AWA.Wikis.Beans.Wiki_View_Bean",
  Handler => Create_Wiki_View_Bean'Access);
```

```
<managed-bean>
 <description>...</description>
 <managed-bean-name>wikiView</managed-bean-name>
 <managed-bean-class>AWA.Wikis.Beans.Wiki_View_Bean<
 <managed-bean-scope>request</managed-bean-scope>
 <managed-property>
  <property-name>image_prefix</property-name>
  <property-class>String</property-class>
  <value>#{contextPath}/images/</value>
 </managed-property>
</managed-bean>
```

Validation in Request Flow




```
<f:metadata>
  <f:viewParam id='page' value='#{wikiView.name}'/>
  <f:viewAction action='#{wikiView.load}'/>
</f:metadata>
```

https://github.com/stcarrez/ada-awa

Solution 1: Validate Data

- Ada Server Faces takes care of data validation:
  - By providing controls before conversion,
  - By converting input to Ada final types
- Ada beans are explicitly declared
- Ada bean's `Set_Value` called after validation
- Data is stored and represented using Ada types

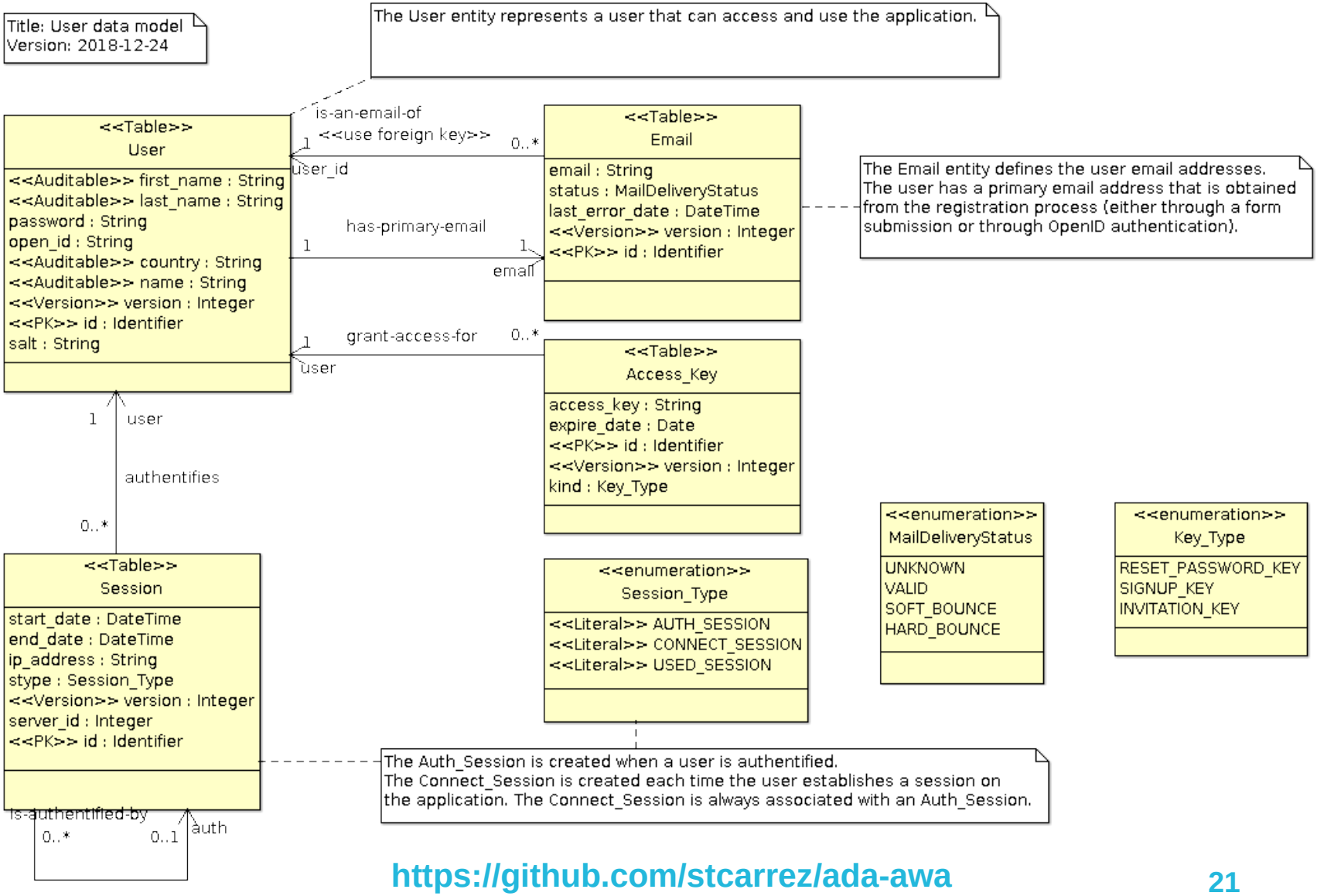
Problem 2: Authenticate Users

- Identify known users
- Get credentials for these users
- Registration process for unknown users

AWA Users Module

- Authenticate users
  - with OpenID Connect
  - with email & password
- Provide full registration and invitation process
- Email validation through access key validation

AWA User, Email and Session



https://github.com/stcarrez/ada-awa

Ada Security: OpenID Connect

- Authentication framework built on top of OAuth2
- Authenticate users with OpenID Connect
  - → Google, Facebook, Twitter, ...

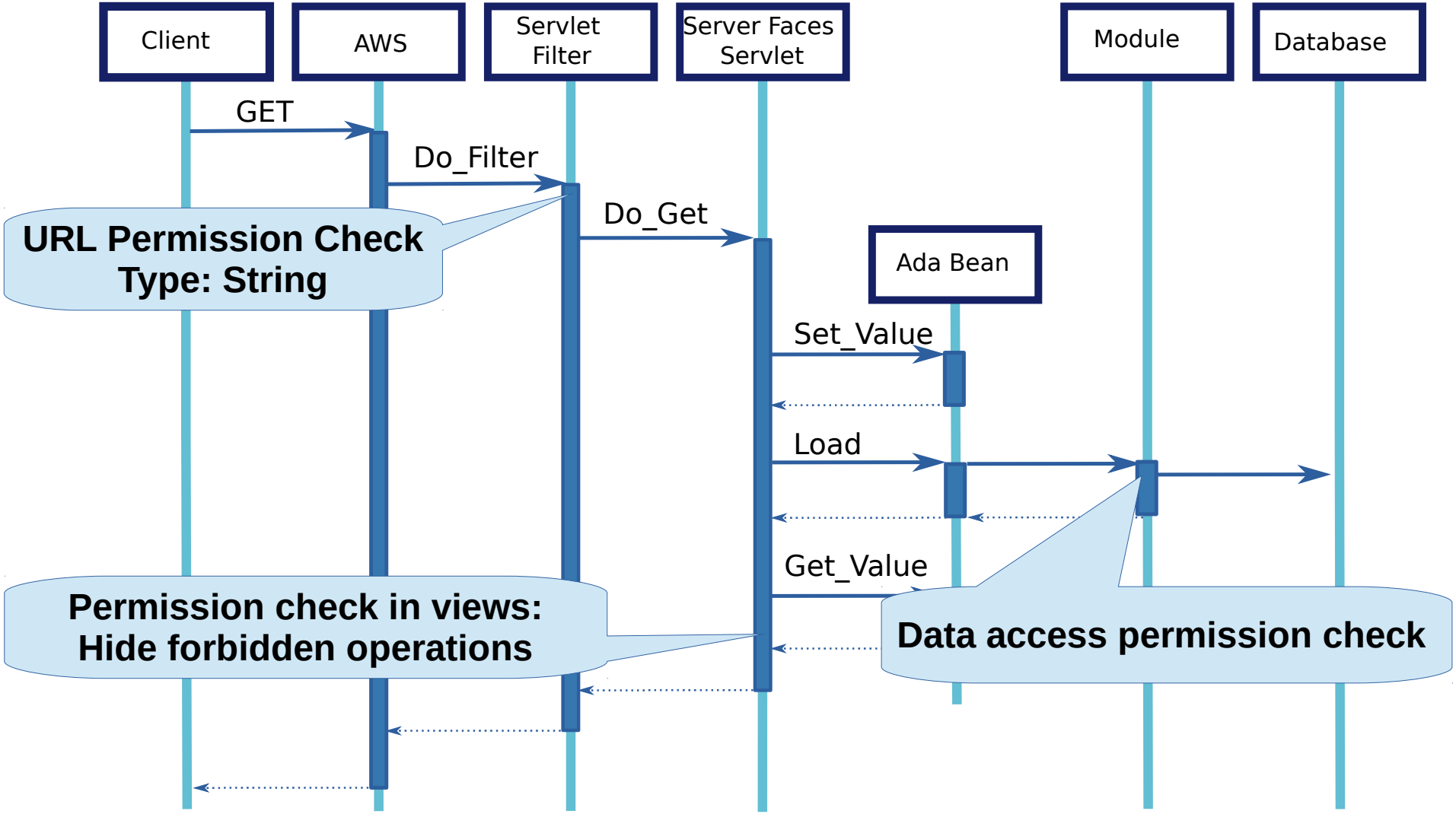
Solution 2: Authenticate Users

- Ada Security provides support for OpenID
- AWA provides some support for user enrollment
  - Online registration
  - Invitation of users through secure key

Problem 3: Authorize Access

- Grant access to authorized users
- Verify before the resource is accessed
- Deny access to unauthorized users

Authorization in Request Flow



https://github.com/stcarrez/ada-awa

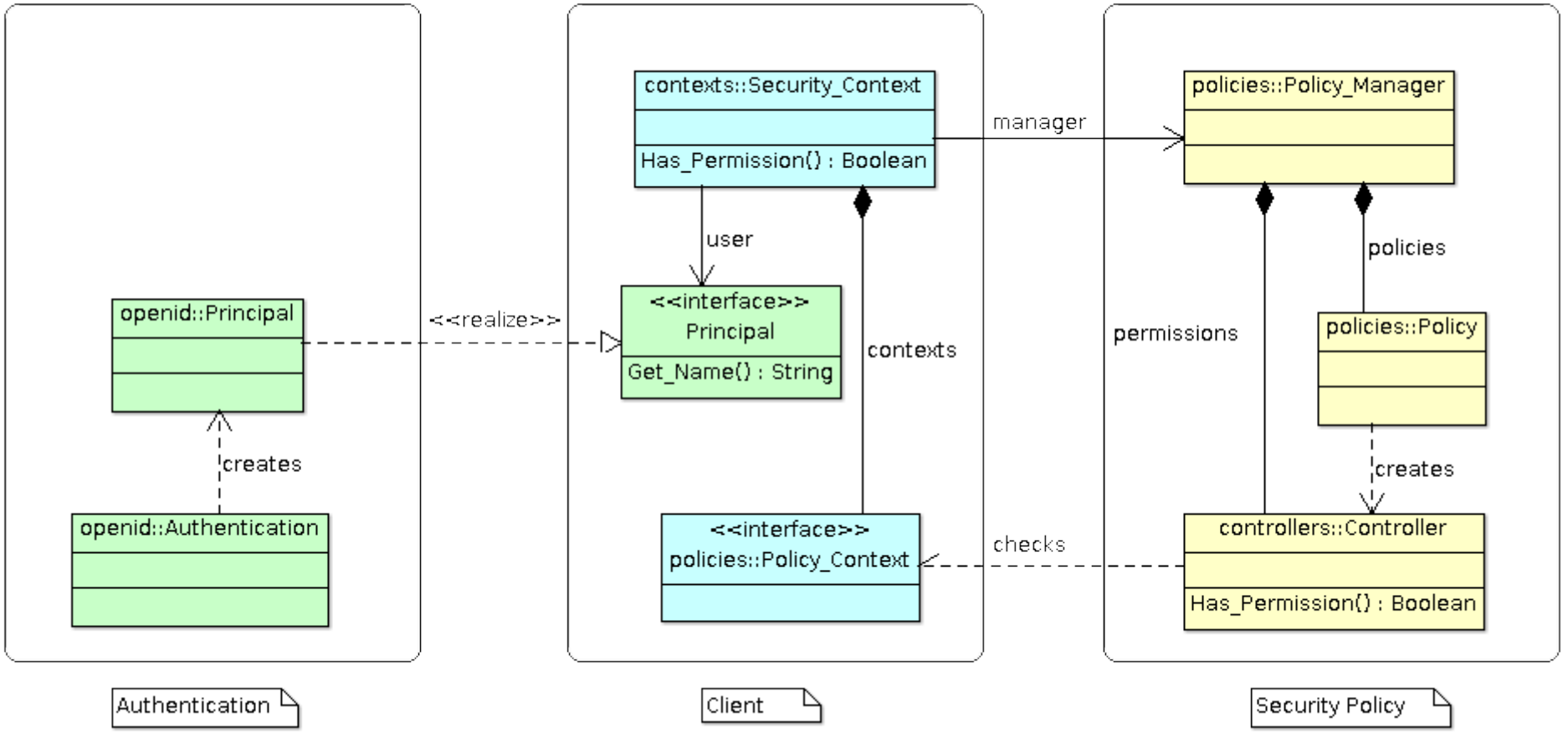
Some Security Concepts

- Policy and policy manager:
  - security rules to protect the system or resources
- Principal:
  - the entity that can be authenticated (credentials)
- Permission:
  - Access to a system or resource

Ada Security

- Security framework to enforce security policies
- Describe security policies
- Authorize access to resources based on security policy and security context

Ada Security Model



https://github.com/stcarrez/ada-awa

Security Policies

- Security policies are checked by a controller
- Use existing policies or write your own

```
type Entity_Controller (Len : Positive) is
limited new Security.Controllers.Controller with record
  Entities : Entity_Type_Array;
  SQL      : String (1 .. Len);
end record;

overriding
function Has_Permission
  (Handler : in Entity_Controller;
   Context : in Security.Contexts.Security_Context'Class;
   Permission : in Security.Permissions.Permission'Class)
      return Boolean;
```

Declaring permissions

- Instantiate `Security.Permissions.Definition`

```
with Security.Permissions;
...
  package ACL_Create is
    new Security.Permissions.Definition (“create”);
```

- Bind the permission to a security controller (XML)

```
<role-permission>
 <name>create</name>
 <role>admin</role>
</role-permission>
```

```
<entity-permission>
 <name>create</name>
 <entity-type>awa_workspace</entity-type>
 <sql>
SELECT acl.id FROM awa_acl AS acl
WHERE acl.entity_type = :entity_type
AND acl.user_id = :user_id
AND acl.entity_id = :entity_id
AND acl.permission = $permission[create]
 </sql>
</entity-permission>
```

Checking permissions

- Checking a permission acts as a barrier
- Raises the `NO_PERMISSION` exception

```
with AWA.Permissions;
...
  AWA.Permissions.Check (Permission => ACL_Create.Permission);
  --  can proceed if permission is granted
```

- Checking a permission in views hides the content

```
<h:panelGroup
   rendered="#{auth:hasPermission('create',wikiSpaceId)}">
  <!-- rendered if permission is granted →
  ...
</h:panelGroup>
```

Solution 3: Authorize Access

- Declare a permission in Ada and configure it
- Check for a permission to block unauthorized users
- Hide content when permission is denied

Getting started with Dynamo

- Creating a project

```
dynamo create-project myproject
./configure
make generate build
./bin/myproject-server
```

- Adding a new page

```
dynamo add-page newpage
```

- Adding a new Ada module

```
dynamo add-module mymodule
```

Conclusion

- AWA takes care of application security
  - By validating user input
  - By enforcing strong typing in the model
  - By authenticating users
  - By authorizing access to resources
- AWA Programmer's Guide
  - https://ada-awa.readthedocs.io/en/latest/