pac4j : la librairie de sécurité pour Java

par Jérôme LELEU

Au programme…

- Bio
- Contexte
- Ecosystème pac4j
- pac4j v1.7 & implémentations
- pac4j v1.8 & implémentations
- Battle versus Spring Security
- Conclusion

Qui suis-je ?

- Jérôme LELEU
- Leader technique @ SFR
- Chairman SSO CAS
- Committer Shiro
- Créateur pac4j
- @leleuj / https://github.com/leleuj

Contexte

Contexte

- Sécurité = problématique #1
- Multi-authentifications (= multi-protocoles)
- Librairies « trop compliquées »
- Librairies disparates (protocoles, frameworks)
- « Ne réinventons plus la roue »

Ecosystème pac4j

Généralités

- Démarré en 2012, Shiro / CAS
- Organization pac4j sous Github, 5 développeurs, mailings lists : pac4j-users / pac4j-dev
- Sûreté
- Simplicité & cohérence (frameworks / protocoles)
- Version actuelle : 1.7, à venir : 1.8

Les composants

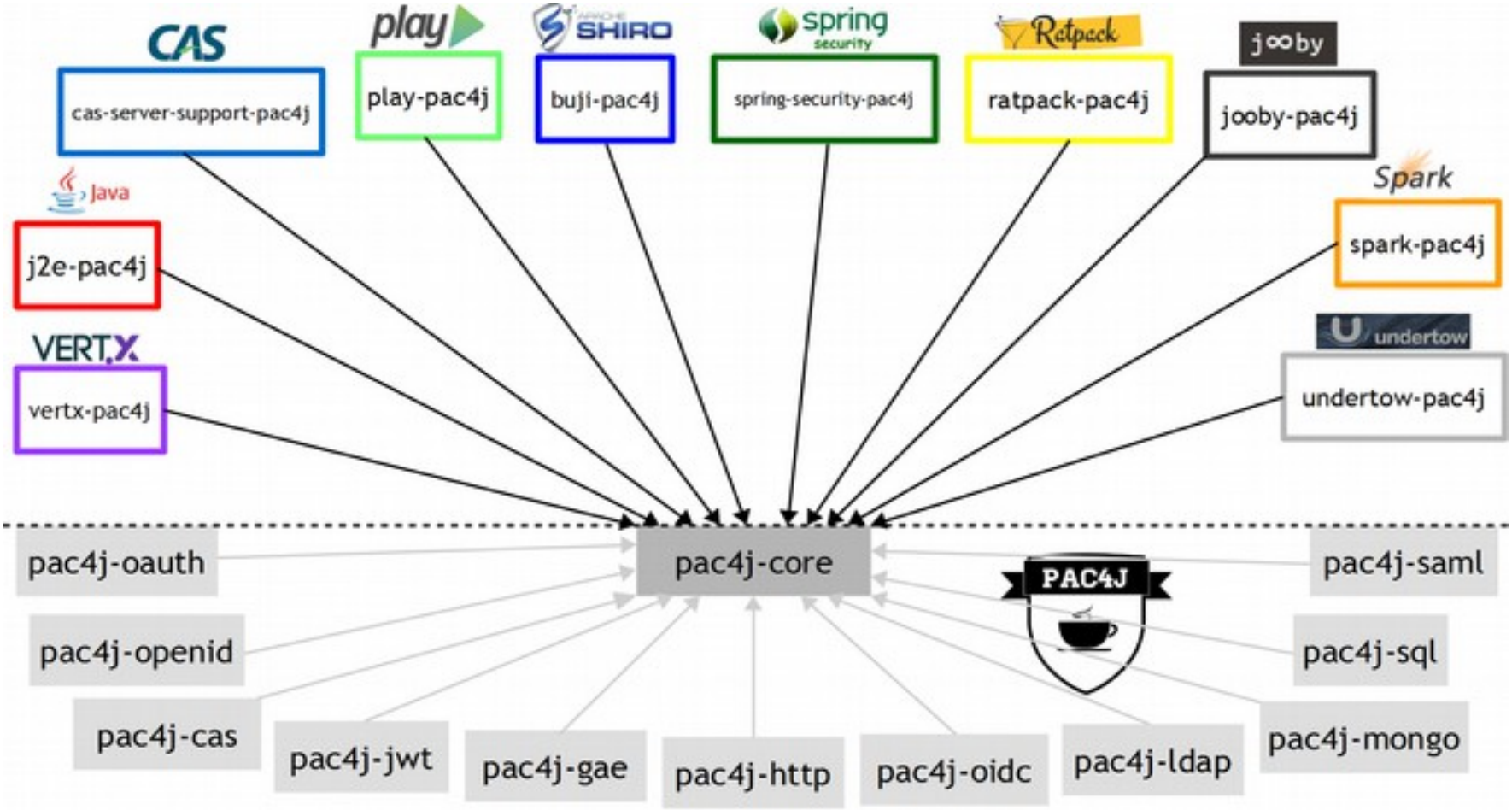
- Coeur = moteur : pac4j / pac4j-*

  Authentification, récupération du profil, autorisations, multi-protocoles

- Implémentations multi frameworks : *-pac4j basé sur pac4j-core

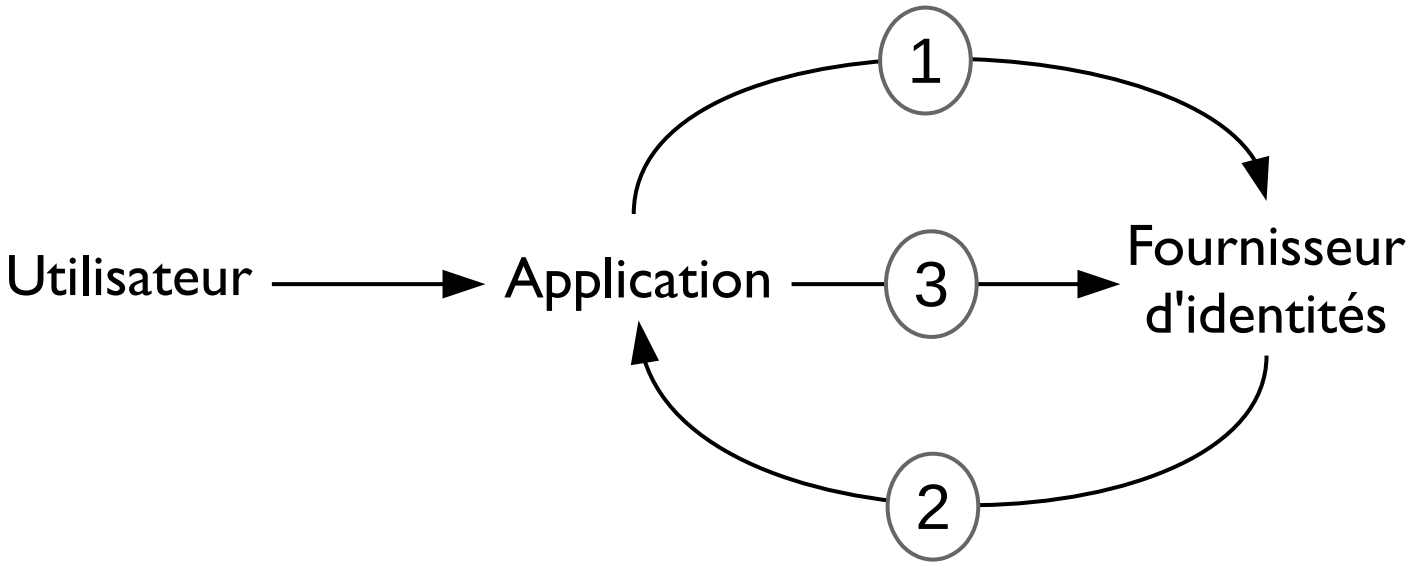
- Démos : *-pac4j-demo

Les librairies



pac4j v1.7 & implémentations

Tous les mêmes protocoles



http://

Les clients

- Interface *Client* et hiérarchie de clients

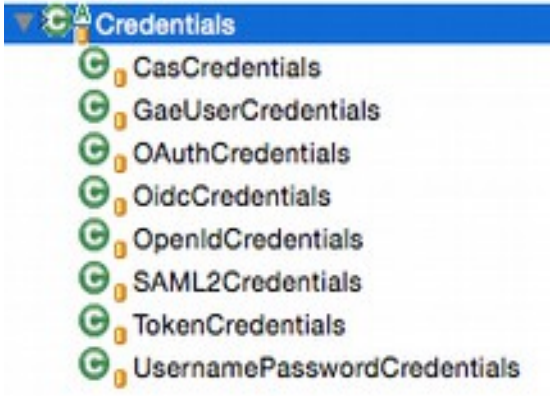

```
public interface Client<C extends Credentials, U extends UserProfile> {

    String getName();

    void redirect(WebContext context, …) throws RequiresHttpAction;

    C getCredentials(WebContext context) throws RequiresHttpAction;

    U getUserProfile(C credentials, WebContext context);
}
```

- Objet *Clients*

Contexte web & credentials

- Interface *WebContext*
- Classe abstraite *Credentials* et hiérarchie de classes

```
public interface WebContext {

    String getRequestParameter(String name);

    Map<String, String[]> getRequestParameters();

    String getRequestHeader(String name);

    void setSessionAttribute(String name, Object value);

    Object getSessionAttribute(String name);

    String getRequestMethod();

    ...
```



Profil utilisateur

- *UserProfile* = id + attributs + roles / permissions
- *CommonProfile* et hiérarchie de **Profile*
- *AttributesDefinition*, *AttributeConverter*
- *AuthorizationGenerator*

Les implémentations

- J2E, Play, Ratpack, Vert.x, Sparkjava, Undertow, Jooby + CAS + Spring Security, Shiro
- **WebContext*
- *RequiresAuthentication* + Callback**

Exemple : j2e-pac4j-demo

```
<filter>
    <filter-name>FacebookFilter</filter-name>
    <filter-class>org.pac4j.j2e.filter.RequiresAuthenticationFilter</filter-class>
    <init-param>
      <param-name>clientsFactory</param-name>
      <param-value>org.leleuj.config.MyClientsFactory</param-value>
    </init-param>
    <init-param>
      <param-name>clientName</param-name>
      <param-value>FacebookClient</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>FacebookFilter</filter-name>
    <url-pattern>/facebook/*</url-pattern>
</filter-mapping>
<filter>
    <filter-name>CallbackFilter</filter-name>
    <filter-class>org.pac4j.j2e.filter.CallbackFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>CallbackFilter</filter-name>
    <url-pattern>/callback</url-pattern>
</filter-mapping>
```

```
public class MyClientsFactory implements ClientsFactory {

  @Override
  public Clients build(final Object env) {
    ...

    FacebookClient facebookClient = new FacebookClient("fi", "fs");
    TwitterClient twitterClient = new TwitterClient("ti", "ts");

    CasClient casClient = new CasClient();
    casClient.setCasLoginUrl("http://mycasserver/login");

    ...
    return new Clients("http://localhost:8080/callback", oidcClient,
          saml2Client, facebookClient, twitterClient, formClient,
          basicAuthClient, casClient, stravaClient);
  }
}
```

Code source de j2e-pac4j

```
public class RequiresAuthenticationFilter extends ClientsConfigFilter {

    protected void internalFilter(final HttpServletRequest request, final HttpServletResponse response,
            final HttpSession session, final FilterChain chain) throws IOException, ServletException {

        final CommonProfile profile = UserUtils.getProfile(request);

        if (profile != null) {
            chain.doFilter(request, response);
        } else {

            String requestedUrl = getRequestedUrl(request);
            session.setAttribute(ORIGINAL_REQUESTED_URL, requestedUrl);

            final WebContext context = new J2EContext(request, response);
            Client<Credentials, CommonProfile> client =
                        ClientsConfiguration.getClients().findClient(this.clientName);
            try {
                client.redirect(context, true, false);
            } catch (RequiresHttpAction e) { }
        }
    }
}
```

pac4j v1.8 & implémentations

Objectifs

- Web services = REST = HTTP (header, paramètre, IP)

  + LDAP, JWT, RDBMS, MongoDB

  + autorisations
- Sûr, simple et facilement extensible
- Mêmes capacités / algorithmes (frameworks) → guidelines d'implémentation
- Version majeure / ETA : fin septembre

Stateful / indirect vs stateless / direct

	IndirectClient	*DirectClient*
Cas d'usage	Interface web	Service web
Sauvegarde et restitution de l'url originale	Oui	Non
Authentification exécutée	1 fois par session	À chaque requête
Récupération des credentials	Sur l'url de callback	Avec la requête HTTP courante
Sauvegarde du profil utilisateur	En session web	Dans la requête courante

DirectClient

```
public abstract class DirectClient<C extends Credentials, U extends CommonProfile>
                                    extends BaseClient<C, U> {

    @Override
    public final void redirect(final WebContext context, ...) {
        throw new TechnicalException("direct clients do not support redirections");
    }
}
```

- *getCredentials(...)* ↔ *Authenticator*
- *getUserProfile(...)* ↔ *ProfileCreator*

Gestion du profil & configuration

- *ProfileManager*

- *Config, ConfigFactory, ConfigBuilder, ConfigSingleton*

Autorisations

- Interface *Authorizer*

```
public interface Authorizer<U extends UserProfile> {

    boolean isAuthorized(WebContext context, U profile);
}
```


```
public class IsAuthenticatedAuthorizer<U extends UserProfile>
        implements Authorizer<U> {

    public boolean isAuthorized(WebContext context, U profile) {
        return profile != null;
    }
}
```

Code source de j2e-pac4j

```
boolean useSession = useSession(context, client);
ProfileManager manager = new ProfileManager(context);
UserProfile profile = manager.get(useSession);

if (profile == null && client instanceof DirectClient) {
  Credentials credentials;
  try {
    credentials = client.getCredentials(context);
  } catch (RequiresHttpAction e) { … }
  profile = client.getUserProfile(credentials, context);
  if (profile != null) {
    manager.save(useSession, profile);
  }
}
if (profile != null) {
  if (authorizer.isAuthorized(context, profile)) {
    chain.doFilter(request, response);
  } else {
    context.setResponseStatus(HttpConstants.FORBIDDEN);
  }
} else {
  if (client instanceof IndirectClient) {
    saveRequestedUrl(context);
    redirectToIdentityProvider(client, context);
  } else {
    context.setResponseStatus(HttpConstants.UNAUTHORIZED);
  }
}
```

Battle versus Spring Security

```xml
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<http pattern="/webservices/**" create-session="stateless" entry-point-
ref="basicAuthEntryPoint">
  <intercept-url pattern="/**" access="isAuthenticated()" />
  <custom-filter position="BASIC_AUTH_FILTER" ref="basicAuthFilter" />
  <custom-filter after="BASIC_AUTH_FILTER" ref="tokenFilter" />
</http>

<bean id="basicAuthEntryPoint" ... />

<authentication-manager alias="restAuthentManager">
  <authentication-provider ref="basicAuthProvider" />
  <authentication-provider ref="tokenProvider" />
</authentication-manager>

<bean id="basicAuthFilter" ...>
        <property name="authenticationManager" ref="restAuthentManager"/>
</bean>

<bean id="tokenFilter" ...>
        <property name="authenticationManager" ref="restAuthentManager"/>
</bean>

<bean id="basicAuthProvider" ... />

<bean id="tokenProvider" ... />

<http pattern="/ihm/**" entry-point-ref="casEntryPoint">
  <intercept-url access="isAuthenticated()" />
  <form-login />
  <custom-filter position="CAS_FILTER" ref="casFilter" />
  <logout />
</http>

<authentication-manager alias="authenticationManager">
  <authentication-provider ref="formProvider" />
  <authentication-provider ref="casProvider" />
</authentication-manager>

<bean id="serviceProperties" ... />

<bean id="casFilter" ... />

<bean id="casEntryPoint" ... />
<bean id="casProvider" ... />
```

Vs

```
setPort(8080);

DirectBasicAuthClient basicAuthClient = new
DirectBasicAuthClient(new BasicAuthAuthenticator());
ParameterClient parameterClient = new
ParameterClient("token", new TokenAuthenticator());

FormClient formClient = new
FormClient("http://localhost:8080/theForm", new
FormAuthenticator());
CasClient casClient = new CasClient();
casClient.setCasLoginUrl("http://mycasserverurl/login");

Clients clients = new
Clients("http://localhost:8080/callback", formClient,
basicAuthClient, parameterClient, casClient);

Route callback = new CallbackRoute(clients);
get("/callback", callback);
post("/callback", callback);
before("/webservices", new
RequiresAuthenticationFilter(clients,
"DirectBasicAuthClient,ParameterClient"));
before("/ihm", new RequiresAuthenticationFilter(clients,
"CasClient"));

get("/theForm", (rq, rs) -> form(rq, clients)
templateEngine);
get("/logout", (rq, rs) -> new ApplicationLogoutFilter());
```

Conclusion

Conclusion

- LA librairie de sécurité pour Java
- Utilisez-la !
- Envie de contribuer ?

Question ?

Merci !

Sponsors


arolla
Mastering Software Development


carbon IT


zenika


oxiane


valtech_


[sfeir]


PALO IT
Innovation accelerator


Xebia


MIRAKL


IPPON
Digital . Technologies . Hosting


SOAT

Annexes

Les versions de pac4j

- v1.0 → v1.3 : OAuth
- v1.4 : + CAS, HTTP (basic auth, form), OpenID
- v1.5 : + SAML
- v1.6 : + Google App Engine
- v1.7 : + OpenID Connect
- v1.8 : + JWT, LDAP, RDBMS, MongoDB + REST + autorisations

Tous les clients

Exemple : play-pac4j-java-demo

```
public class Application extends JavaController {

    private static Result protectedIndex() {
        // profile
        final CommonProfile profile = getUserProfile();
        return ok(views.html.protectedIndex.render(profile));
    }

    @RequiresAuthentication(clientName = "FacebookClient")
    public static Result facebookIndex() {
        return protectedIndex();
    }
```

```
GET   /                       controllers.Application.index()
GET   /facebook/index.html    controllers.Application.facebookIndex()
GET   /callback               org.pac4j.play.CallbackController.callback()
POST  /callback               org.pac4j.play.CallbackController.callback()
```

Code source de play-pac4j

```
public class CallbackController extends Controller {

    public static Promise<Result> callback() {
        final Clients clientsGroup = Config.getClients();
        final BaseClient client = (BaseClient) clientsGroup.findClient(context);
        final JavaWebContext context = new JavaWebContext(request(), response(), session());

        Promise<Result> promise = Promise.promise(new Function0<Result>() {
            public Result apply() {
                Credentials credentials = null;
                try {
                    credentials = client.getCredentials(context);
                } catch (final RequiresHttpAction e) {
                    return handleHttpAction(e) ;
                }

                final CommonProfile profile = client.getUserProfile(credentials, context);
                final String sessionId = StorageHelper.getOrCreationSessionId(session());
                if (profile != null) {
                    StorageHelper.saveProfile(sessionId, profile);
                }

                final String requestedUrl = StorageHelper.getRequestedUrl(sessionId, client.getName());
                return redirect(defaultUrl(requestedUrl, Config.getDefaultSuccessUrl()));
            }
        });
        return promise;
    }
```

Futur

- *-pac4j → modules officiels
- Supporter plus de frameworks
- Supporter plus de mécanismes d'authentification
- Bâtir une vraie communauté
- Déconnexion / Enregistrement ?