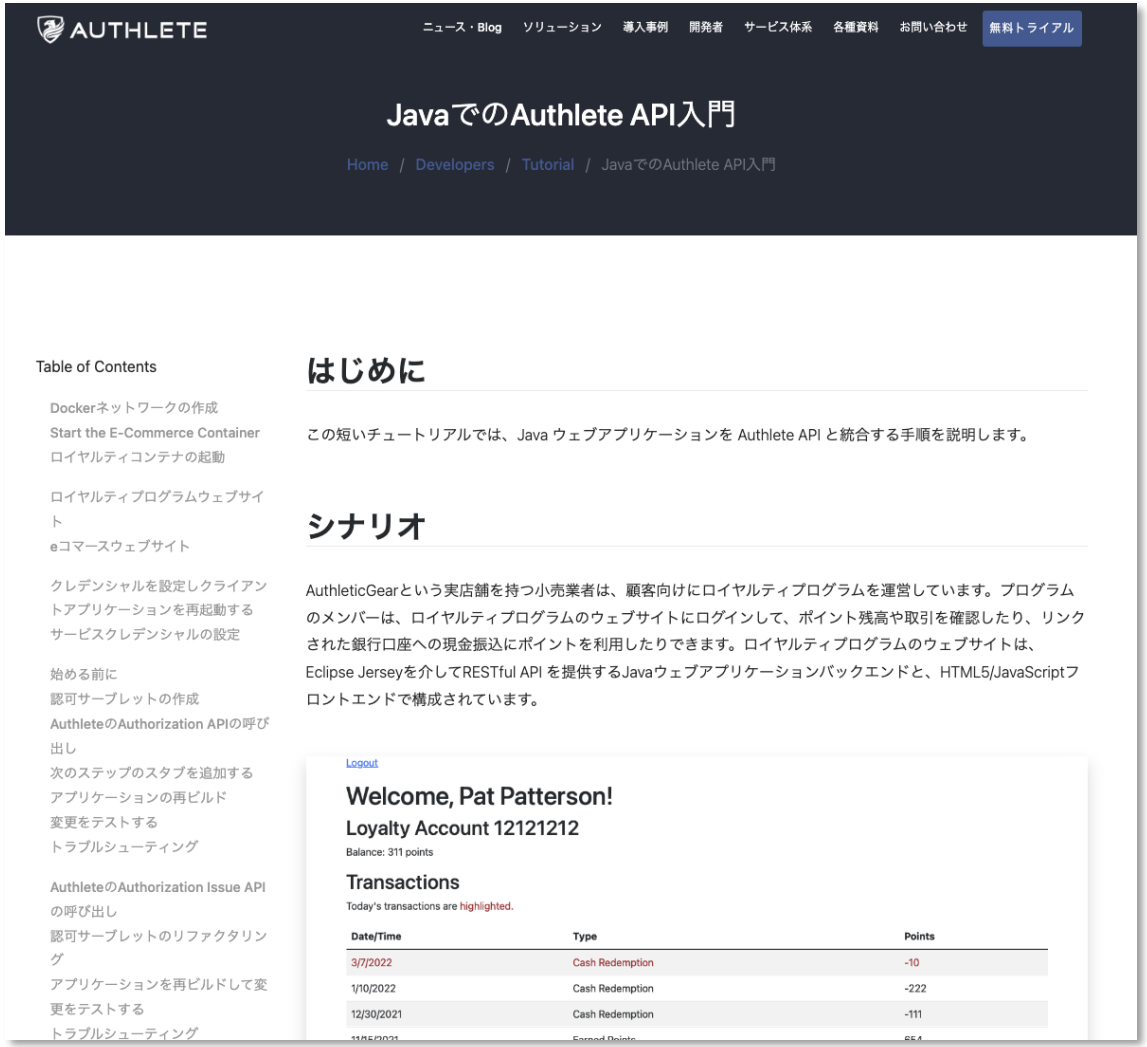
OAuth & OpenID Connect 勉強会 by Authlete - OAuth/OIDC 機能の組み込みかた

Authlete API を用いた実装例

森川将聖
Authlete

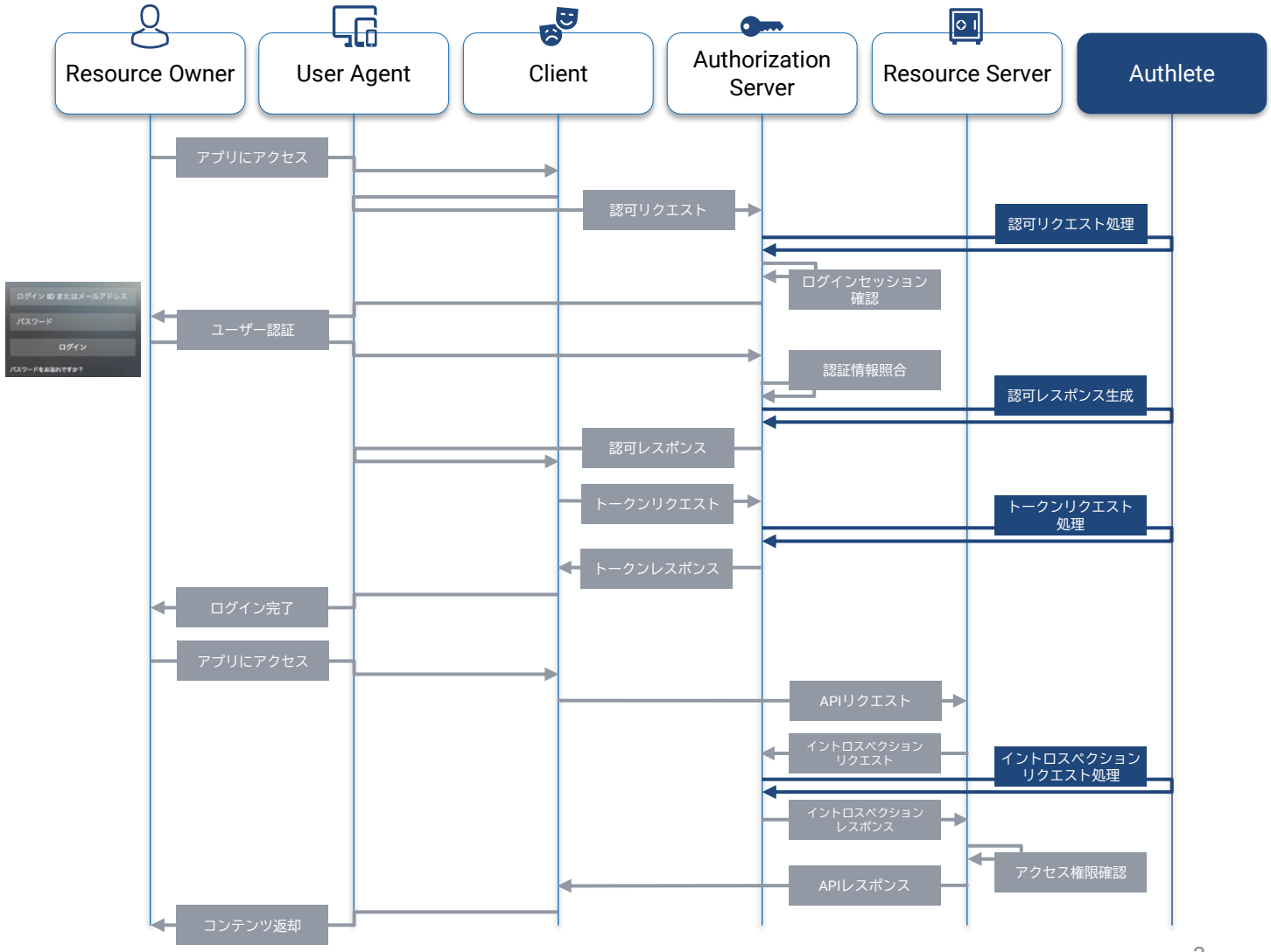
はじめに

- 弊社の Web サイトで公開しているチュートリアルの内容を紹介します
  - Java のサンプル Web アプリケーションに OAuth の実装を段階的に追加する流れを実際のコードの一部を確認しながら追っていきます
- Authlete を用いることにより、短時間で効率的に OAuth の実装を Web アプリケーションに追加できることを確認します



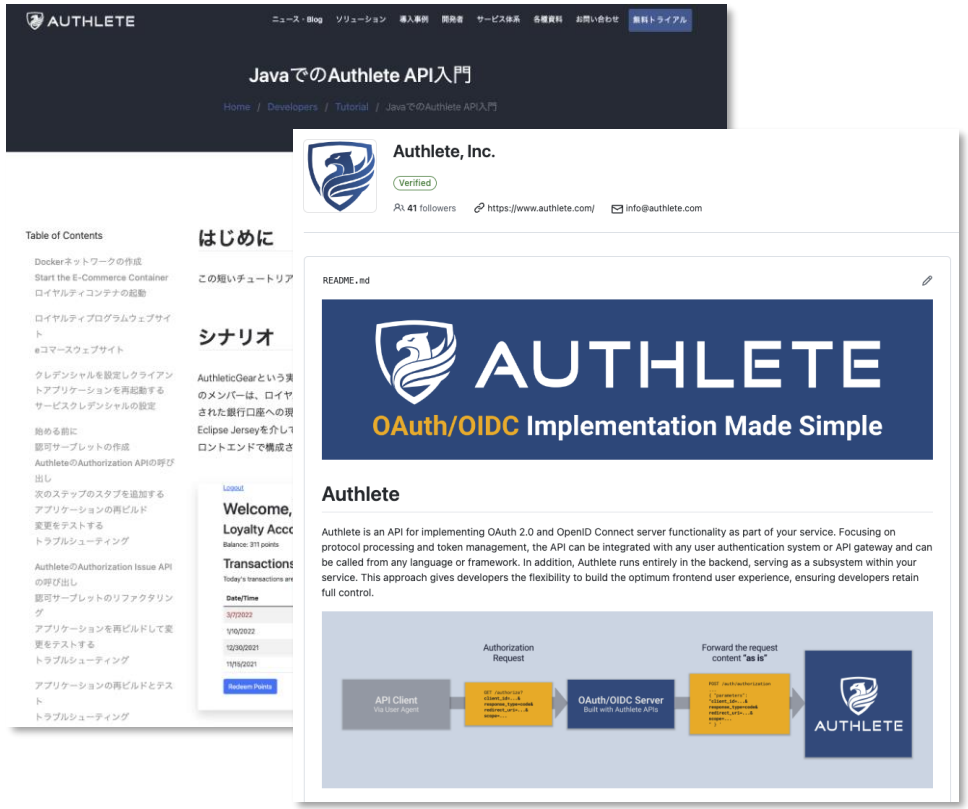
Authlete: OAuth/OIDC化に必要な実装を提供

- OAuth/OIDCリクエスト処理を代行するAPI
- トークン管理のための永続的ストレージ
- 鍵管理やクライアント設定の一元化
- ユーザーのセルフサービスや運用を支援する管理API



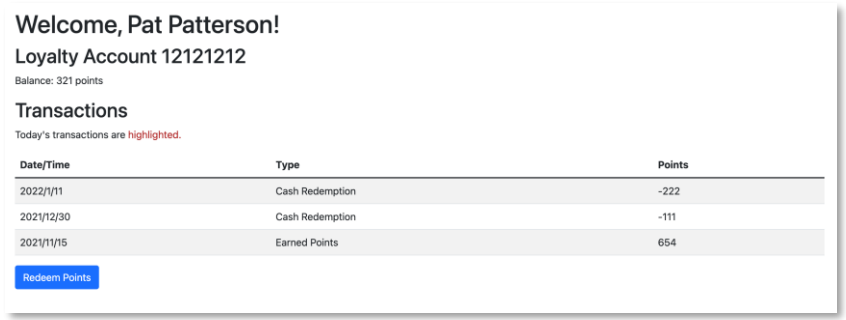
Authlete を用いた OAuth/OIDC 化のチュートリアル

- Java での Authlete API 入門（本日紹介するチュートリアル）
  - https://www.authlete.com/ja/developers/tutorial/getting_started_java/
- GitHub リポジトリ
  - https://github.com/authlete/java-getting-started
- 必要なもの
  - Docker を使える環境
  - お好みのエディタ
  - Java EE の基本的な知識

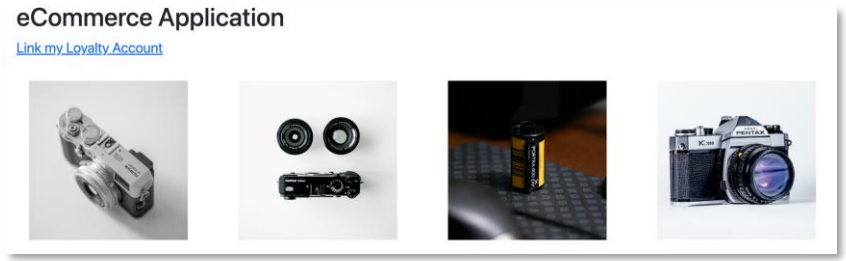


チュートリアルのシナリオ

- 2つのサンプル Web サイト
  - ポイントサイト
    - ログイン/ログアウト機能
    - ポイントの管理



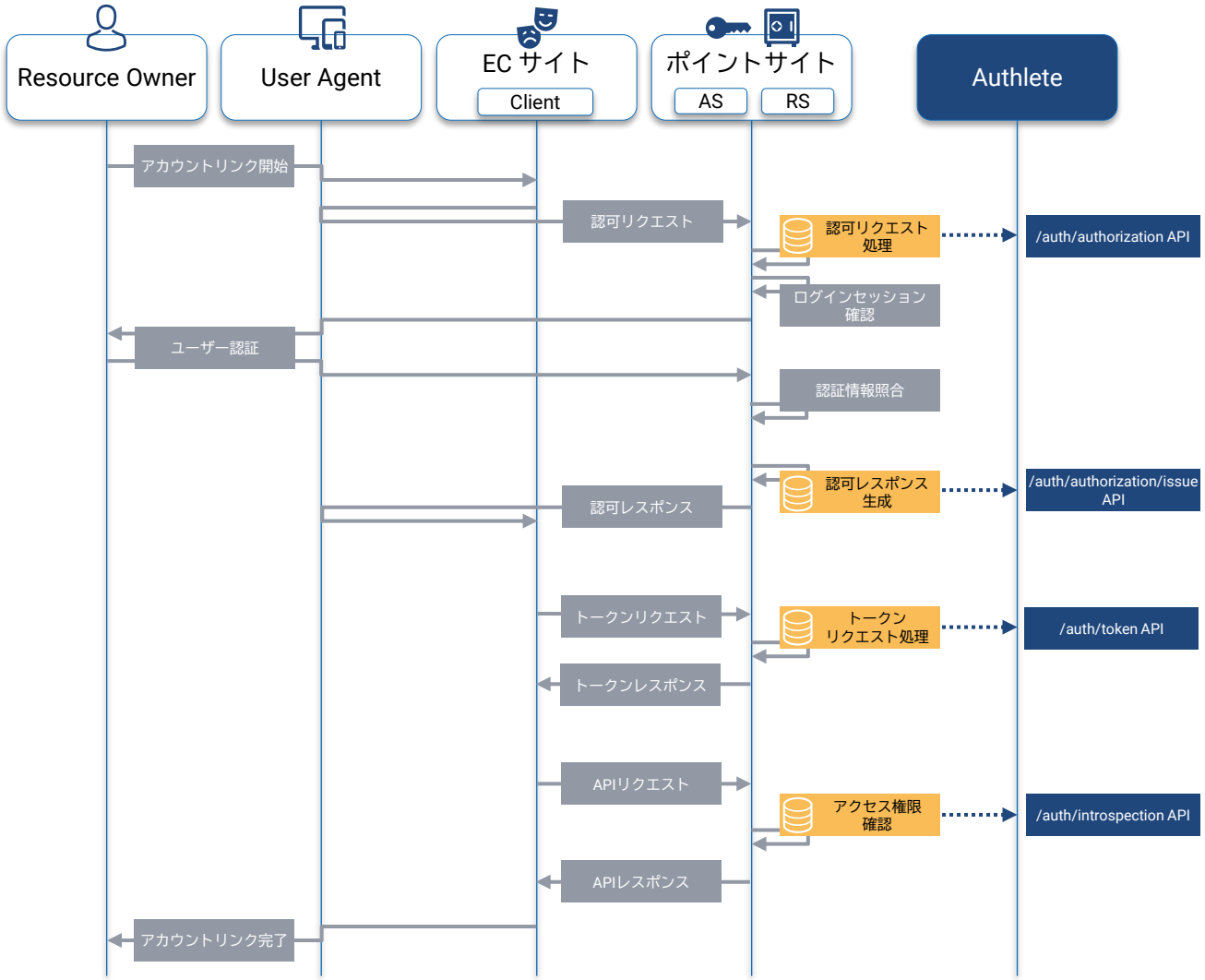
  - EC サイト
    - OAuth クライアント機能



- 目標
  - ユーザーが保有するポイントの残高を EC サイト上で表示する
    - ポイントサイトに認可サーバー/リソースサーバー実装を追加

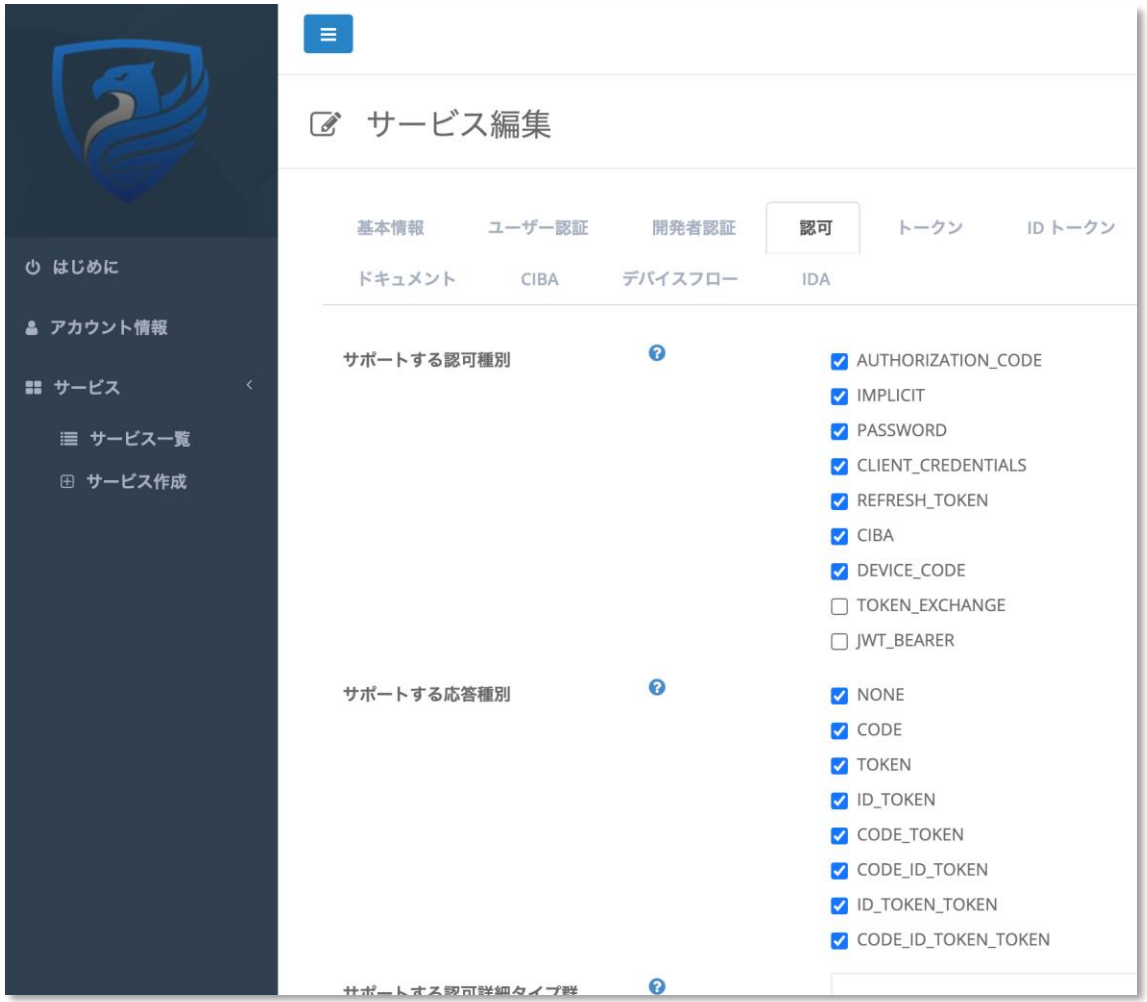
チュートリアルのシナリオ

- ポイントサイトに OAuth の実装を追加する
  - 認可サーバー実装
  - リソースサーバー実装
- 各実装箇所で処理に対応する Authlete API を使う



Authlete のサービス設定

- サービスオーナーコンソール
  - 認可サーバーに関する設定を行うためのコンソール画面
  - サービスオーナーアカウントでログイン

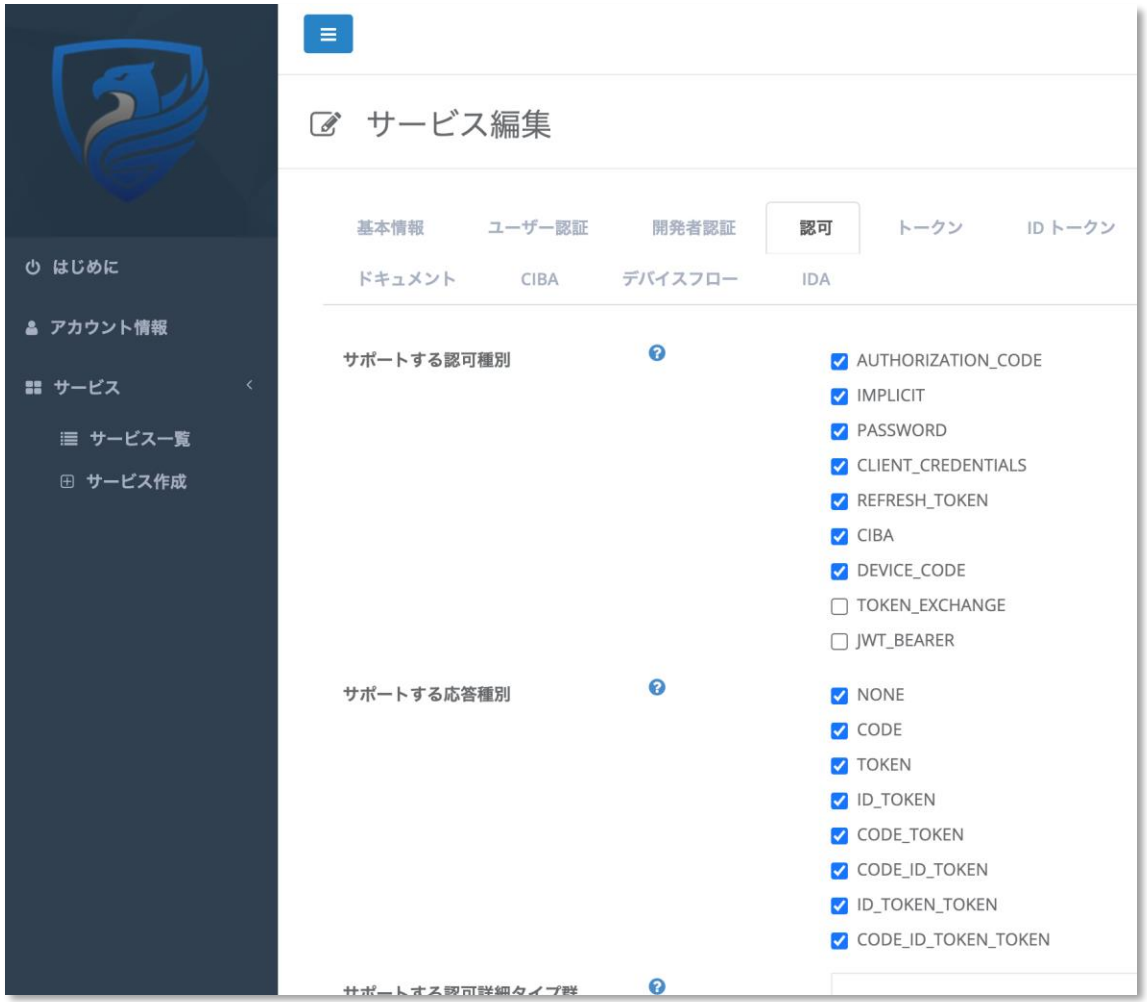


https://so.authlete.com/

Authlete のサービス設定

- API Key/Secret
  - ポイントサイトのソースディレクトリ（/loyalty/src/main/webapp/WEB-INF/）に authleteCredential.json というファイルを作成し API Key/Secret を記載する

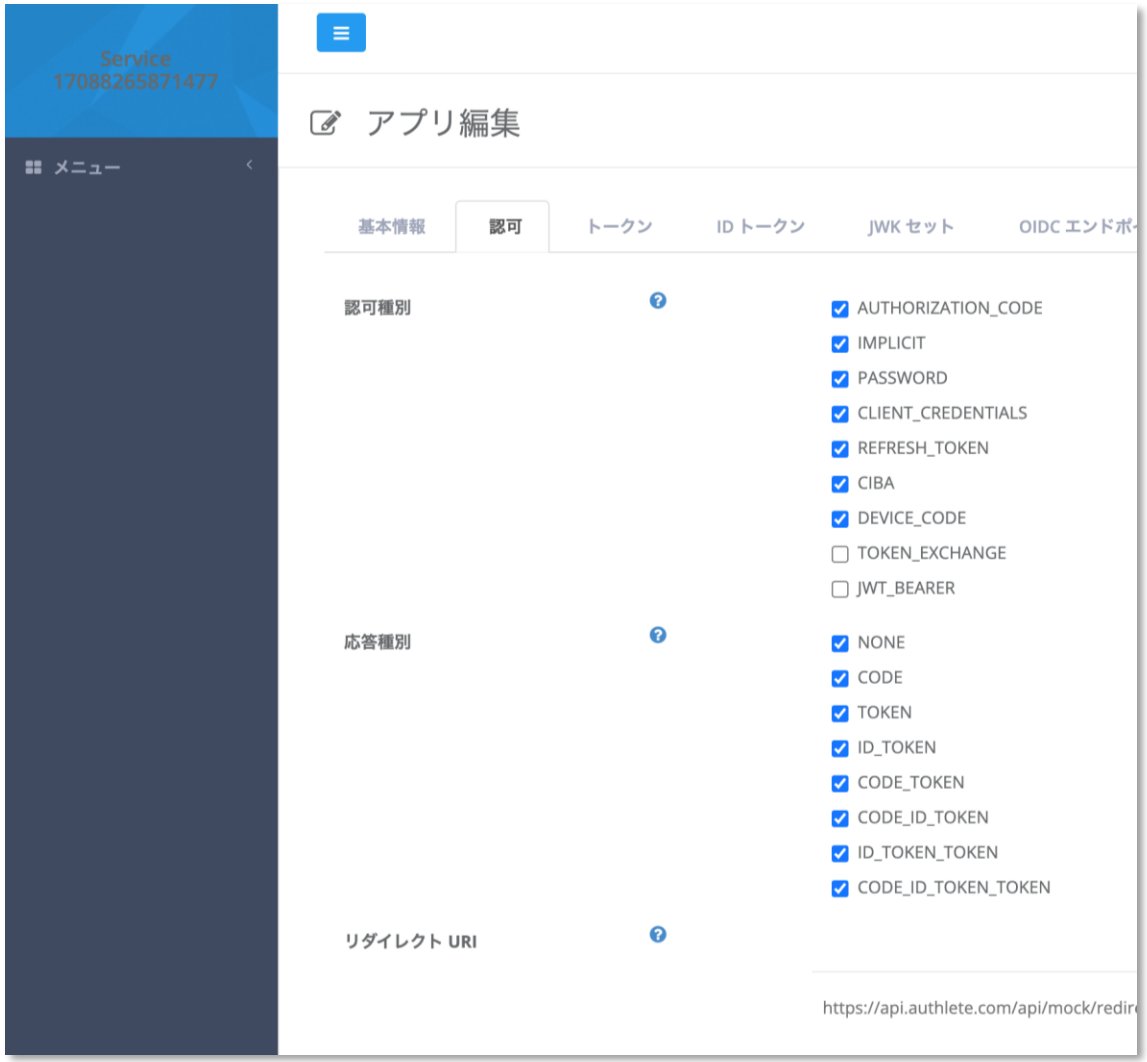
    ```
    {
      "api_key" : "<your api key>",
      "api_secret" : "<your api secret>"
    }
    ```
- サポートする認可種別
  - `AUTHORIZATION_CODE` が含まれていることを確認する
- サポートする応答種別
  - `CODE` が含まれていることを確認する
- サポートするクライアント認証方式
  - `CLIENT_SECRET_POST` が含まれていることを確認する



https://so.authlete.com/

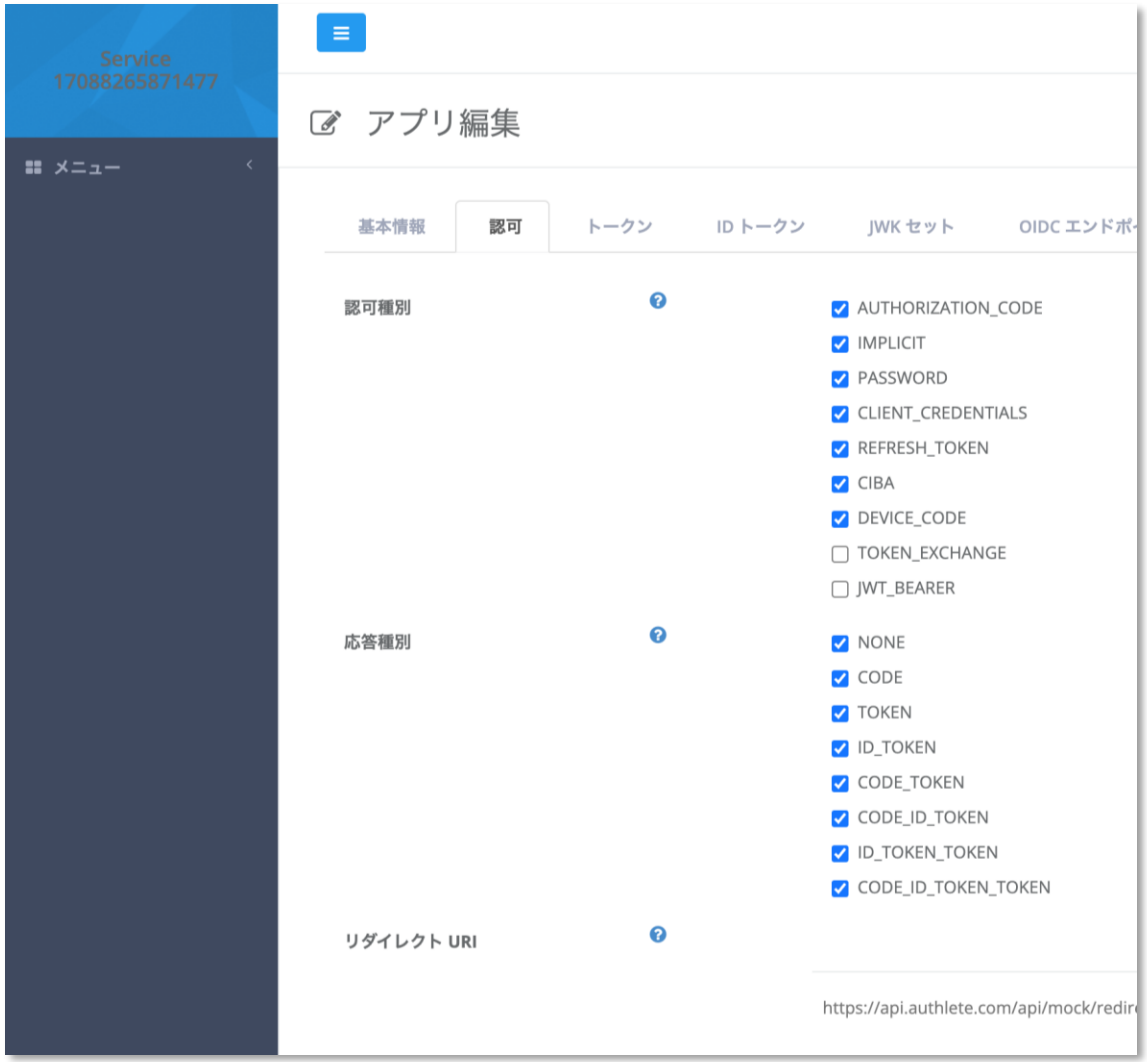
Authlete のクライアント設定

- クライアントデベロッパーコンソール
  - クライアントの設定を行うためのコンソール画面
  - サービスの API Key/Secret でログイン

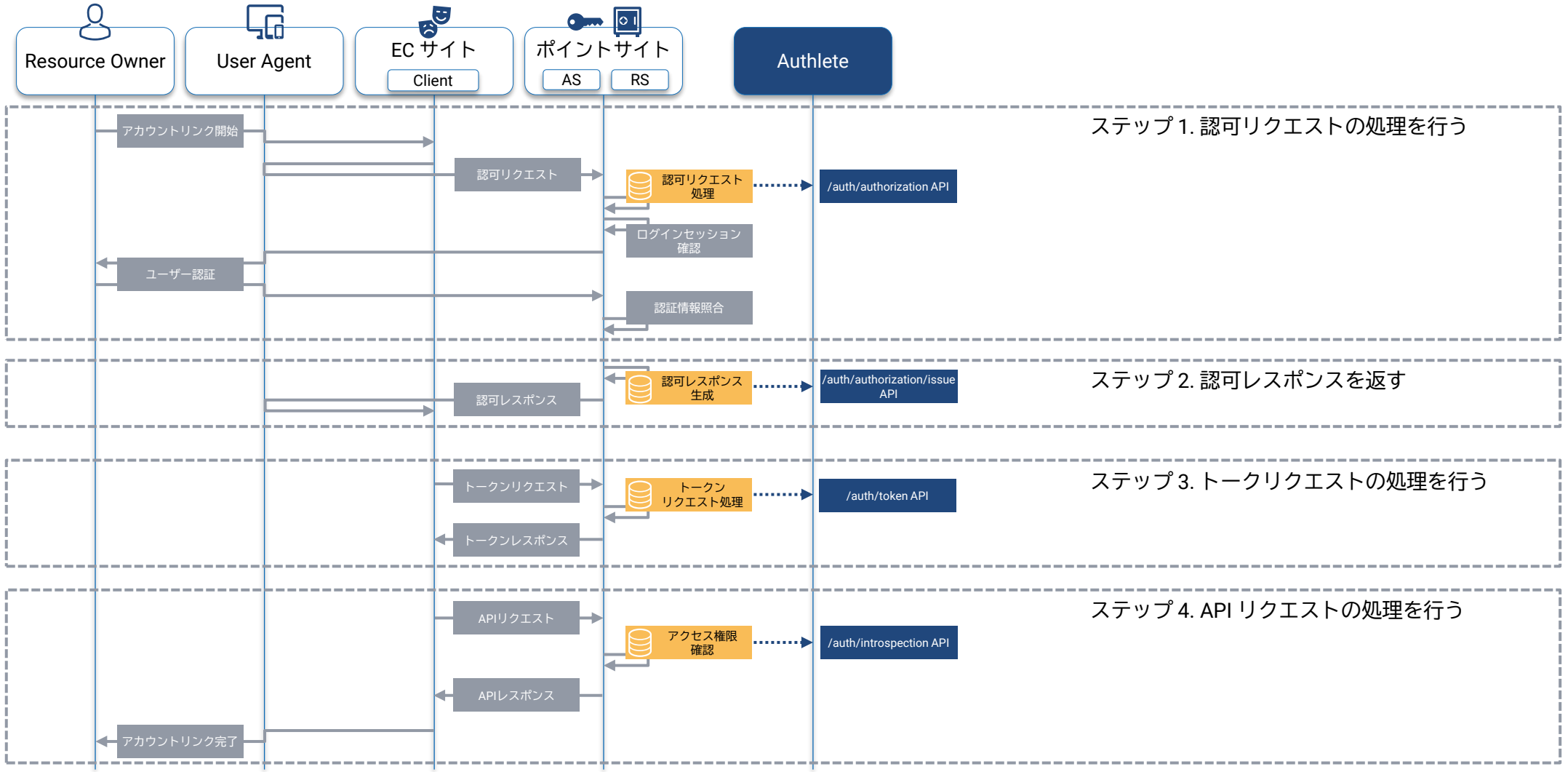


Authlete のクライアント設定

- client_id/client_secret
  - EC サイトのコンテナ起動時に環境変数 `CLIENT_ID` と `CLIENT_SECRET` として設定する
- クライアントタイプ（Client Type）
  - `CONFIDENTIAL` を選択する
- 認可種別（Grant Types）
  - `AUTHORIZATION_CODE` が含まれていることを確認する
- 応答種別（Response Types）
  - `CODE` が含まれていることを確認する
- リダイレクト URI（Redirect URI）
  - ` http://localhost:8080/ecommerce/oauth` を追加する
- クライアント認証方式（Client Authentication Method）
  - ` CLIENT_SECRET_POST` を選択する

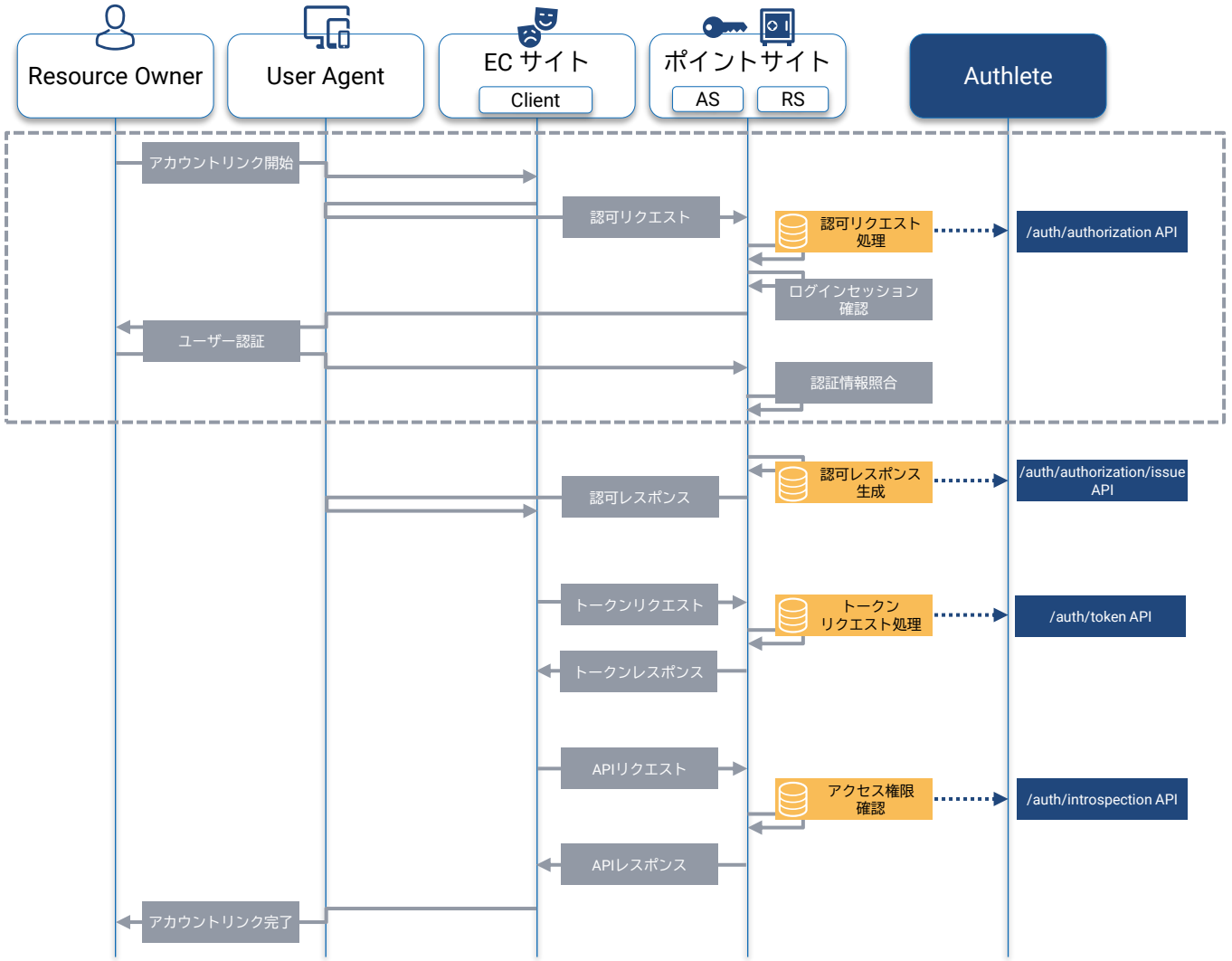


ステップ 1 ~ 4


Resource Owner
User Agent
EC サイト
Client
ポイントサイト
AS
RS
Authlete
アカウントリンク開始
認可リクエスト
認可リクエスト処理
/auth/authorization API
ログインセッション確認
ユーザー認証
認証情報照合
ステップ 1. 認可リクエストの処理を行う
認可レスポンス
認可レスポンス生成
/auth/authorization/issue API
ステップ 2. 認可レスポンスを返す
トークンリクエスト
トークンリクエスト処理
/auth/token API
トークンレスポンス
ステップ 3. トークリクエストの処理を行う
APIリクエスト
アクセス権限確認
/auth/introspection API
APIレスポンス
アカウントリンク完了
ステップ 4. API リクエストの処理を行う

ステップ 1. 認可リクエストの処理を行う

- クライアントから送られてきた認可リクエストのパラメーターを取り出す
- Authlte の /auth/authorization API をコールする
  - 認可リクエストのパラメーターを「そのまま」送信
- API レスポンスを保存する
  - `ticket` レスポンスパラメーター



ステップ 1. 認可リクエストの処理を行う

/java-getting-started/loyalty/src/main/java/com/authlete/simpleauth/oauth/OAuthAuthorizationServlet.java

```
private void initiateAuthleteAuthorization(HttpServletRequest request,
        HttpServletResponse response) throws IOException {
    // 1. Get a Jersey HTTP client
    Client client = OAuthUtils.getClient(getServletContext());

    // 2. We will wrap the incoming query string in a JSON object
    Map<String, Object> requestMap =
            Collections.singletonMap("parameters", request.getQueryString());

    // 3. Call the Authlete Authorization endpoint
    String url = "https://api.authlete.com/api/auth/authorization";

    logger.info("Sending API request to {}:\n{}", url, OAuthUtils.prettyPrint(requestMap));

    // 4. Make the API call, parsing the JSON response into a map
    Map<String, Object> authApiResponse = client.target(url).request().post(
            Entity.entity(requestMap, MediaType.APPLICATION_JSON_TYPE), new GenericType<>() {});

    logger.info("Received API response:\n{}", OAuthUtils.prettyPrint(authApiResponse));

    // 5. 'action' tells us what to do next, 'responseContent' is the payload we'll
    // return
    String action = (String) authApiResponse.get("action");
    String responseContent = (String) authApiResponse.get("responseContent");

    // 6. Perform the action
    switch (action) {
        case "INTERACTION":
            List<Object> prompts = (List<Object>) authApiResponse.get("prompts");
            for (Object prompt : prompts) {
                if (prompt.equals("LOGIN")) {
                    // 7. Prompt the user to login
                    request.getSession().setAttribute("authApiResponse", authApiResponse);
                    LoginUtils.redirectForLogin(request, response);
                    return;
                }
            }
            break;
        // 8. Handle errors
        case "INTERNAL_SERVER_ERROR":
            OAuthUtils.setResponseBody(response, HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    responseContent);
            return;

        case "BAD_REQUEST":
            OAuthUtils.setResponseBody(response, HttpServletResponse.SC_BAD_REQUEST,
                    responseContent);
            return;
    }

    // 9. We should never get here!
    Map<String, String> errorResponse = Map.of("error", "unexpected_error", "error_description",
            "Contact the service owner for details");
    OAuthUtils.setResponseBody(response, HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
            mapper.writeValueAsString(errorResponse));
}
```

- クライアントから送られてきた認可リクエストのパラメーターを Authlete の /auth/authorization API にそのまま渡す
- /auth/authorization API の `action` レスポンスパラメーターの値に応じて次の動作を決定
  - `INTERACTION` の場合は次の処理（ユーザー認証）へ（ /auth/authorization API のレスポンス内容を保存する）
  - `INTERACTION` 以外の場合は `responseContent` レスポンスパラメーターの内容をレスポンスボディとして返す

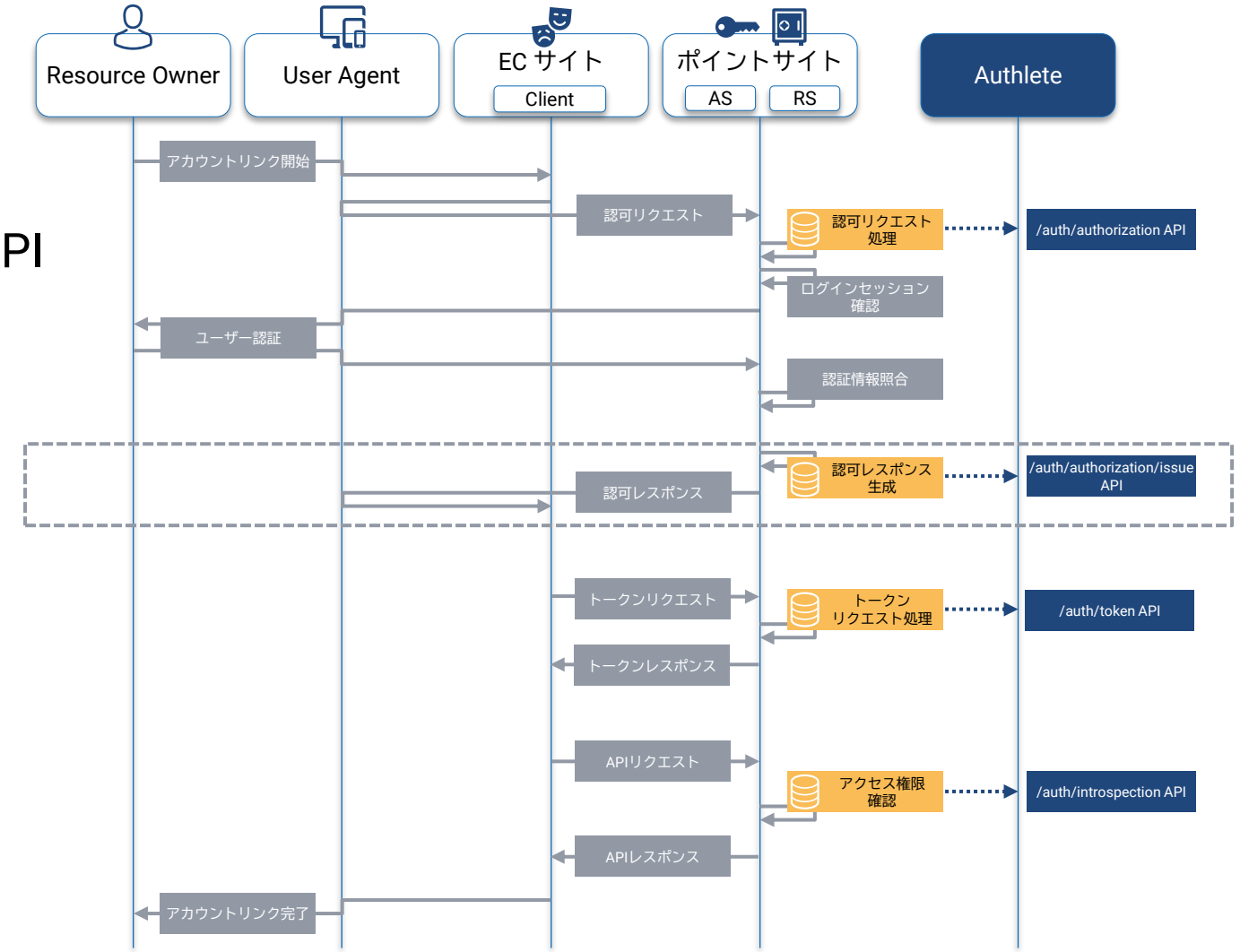
補足: Authlete API からのレスポンスの処理

- Authlete API のレスポンスに含まれるパラメーター
  - `action`
    - 認可サーバーが次に何をするべきかを表すパラメーター
  - `responseContent`
    - 認可サーバーが返すべきデータを表すパラメーター

https://www.authlete.com/ja/kb/getting-started/implementing-an-authorization-server/handling-responses-from-authlete-apis/

ステップ 2. 認可レスポンスを返す

- ユーザー認証の結果に応じた Authlete API をコールする
  - 成功:/auth/authorization/issue API
    - `ticket` パラメーター
    - `subject` パラメーター
  - 失敗:/auth/authorization/fail API
    - `ticket` パラメーター
- Authlete API から得たレスポンスの内容を返す



ステップ 2. 認可レスポンスを返す

/java-getting-started/loyalty/src/main/java/com/authlete/simpleauth/oauth/OAuthAuthorizationServlet.java

```java
private void processAuthleteAuthorization(HttpServletRequest request,
        HttpServletResponse response, Map<String, Object> authApiResponse) throws IOException {
    // 1. Create a Map to send in the Authlete API request
    Map<String, Object> requestMap = new HashMap<>();

    // 2. Copy the ticket from the last API response into the map
    requestMap.put("ticket", authApiResponse.get("ticket"));

    // 3. Verify that the user is actually logged in
    UserAccount authenticatedUser = LoginUtils.getAuthenticatedUser(request.getSession());
    if (authenticatedUser == null) {
        requestMap.put("reason", "NOT_LOGGED_IN");
        OAuthUtils.handleAuthleteApiCall(getServletContext(), response,
                "/auth/authorization/fail", requestMap);
        return;
    }

    // 4. Issue the code
    requestMap.put("subject", authenticatedUser.getUsername());
    OAuthUtils.handleAuthleteApiCall(getServletContext(), response, "/auth/authorization/issue",
            requestMap);
}
```

- Authlete の /auth/authorization API から得た `ticket` パラメーターを用意する
- 最終的にユーザー認証が成功しなかった場合は /auth/authorization/fail API をコールする
  - `ticket` パラメーター
- ユーザー認証が成功した場合は /auth/authorization/issue API をコールする
  - `ticket` パラメーター
  - `subject` パラメーター
    - 特定したユーザーの識別子を指定する
- コールした API から得た `action` に応じた処理を行う

AUTHLETE

ステップ 2. 認可レスポンスを返す

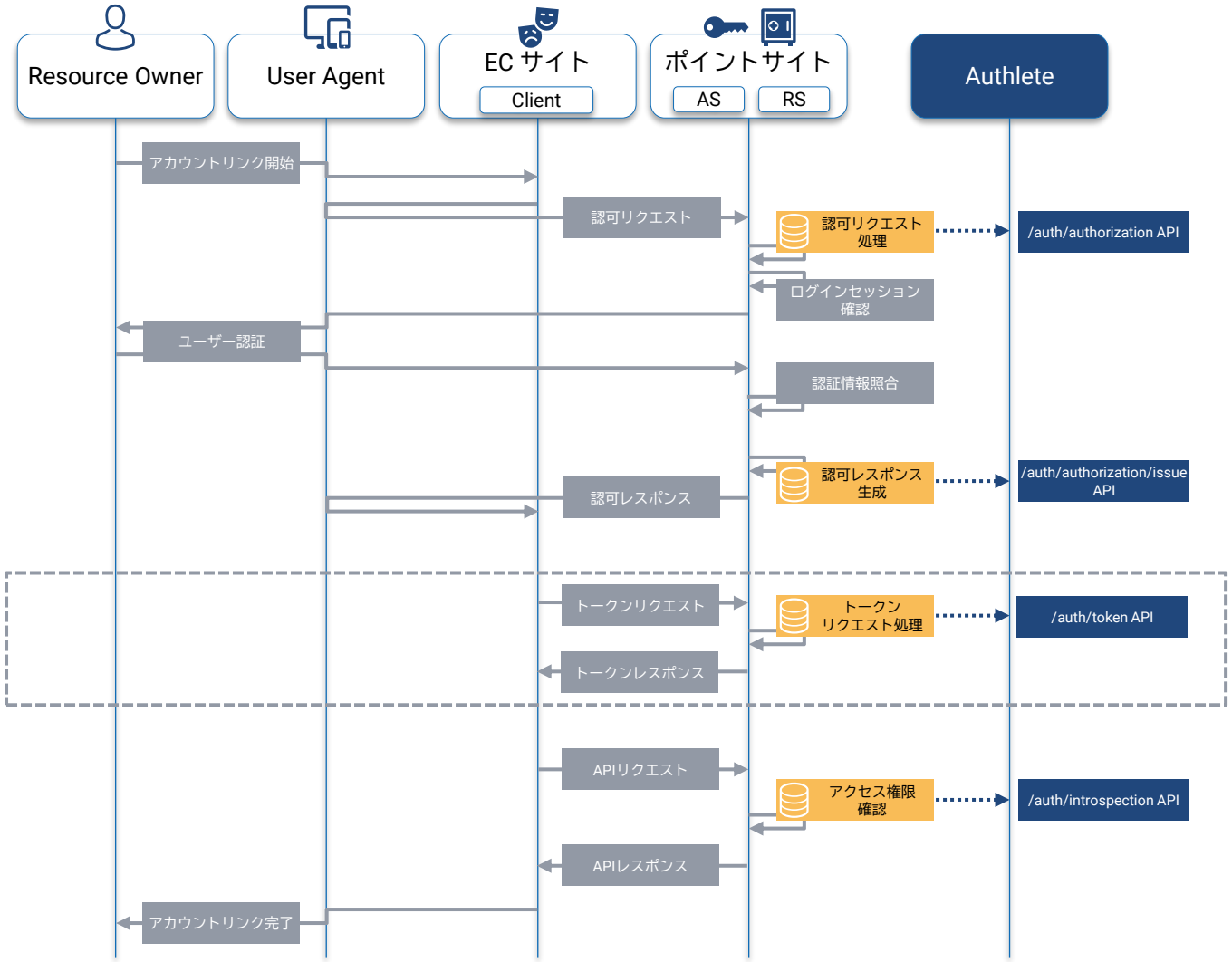
/java-getting-started/loyalty/src/main/java/com/authlete/simpleauth/oauth/OAuthUtils.java

```java
public static Map<String, Object> handleAuthleteApiCall(ServletContext context,
        HttpServletResponse response, String api, Map<String, Object> requestMap)
        throws IOException {
    logger.debug("Calling API {} with params {}", api, OAuthUtils.prettyPrint(requestMap));

    Map<String, Object> responseMap = getClient(context).target(AUTHLETE_BASE + api).request()
            .post(Entity.entity(requestMap, MediaType.APPLICATION_JSON_TYPE),
                    new GenericType<>() {});

    logger.debug("Received API response {}", OAuthUtils.prettyPrint(responseMap));

    String action = (String) responseMap.get("action");
    String responseContent = (String) responseMap.get("responseContent");

    switch (action) {
        case "INTERNAL_SERVER_ERROR":
            setResponseBody(response, HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    responseContent);
            return null;
        case "BAD_REQUEST":
            setResponseBody(response, HttpServletResponse.SC_BAD_REQUEST, responseContent);
            return null;
        case "LOCATION":
            response.setStatus(HttpServletResponse.SC_FOUND);
            response.setHeader("Location", responseContent);
            response.setHeader("Cache-Control", "no-store");
            response.setHeader("Pragma", "no-cache");
            return null;
    }

    return responseMap;
}
```

- 直前にコールした Authlete API から得た内容に応じたレスポンスを Web ブラウザに返す
  - `action` レスポンスパラメーター
    - HTTP ステータスコードを決定する
  - `responseContent` レスポンスパラメーター
    - レスポンスボディまたはヘッダーに設定する
- ここまでの処理が成功していると `action` の値は `LOCATION` になり `responseContent` には認可コードが含まれます

ステップ 3. トークンリクエストの処理を行う

- クライアントから送られてきたトークンリクエストのパラメーターを取り出す
- Authlte の /auth/token API をコールする
  - トークンリクエストのパラメーターを「そのまま」送信
- Authlete API から得たレスポンスの内容をクライアントに送る



ステップ 3. トークンリクエストの処理を行う

/java-getting-started/loyalty/src/main/java/com/authlete/simpleauth/oauth/OAuthTokenServlet.java

```java
@WebServlet("/oauth/token")
public class OAuthTokenServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final ObjectMapper mapper = new ObjectMapper();

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        // 1. Call the /auth/token endpoint, passing the request body
        String body = new String(request.getInputStream().readAllBytes(), StandardCharsets.UTF_8);
        Map<String, Object> authApiResponse = OAuthUtils.handleAuthleteApiCall(getServletContext(),
                response, "/auth/token", Collections.singletonMap("parameters", body));

        // 2. handleAuthleteApiCall() returns null if it already returned a response to the client
        if (authApiResponse == null) {
            return;
        }

        // 3. Perform the action
        String action = (String) authApiResponse.get("action");
        if (action.equals("OK")) {
            String responseContent = (String) authApiResponse.get("responseContent");
            OAuthUtils.setResponseBody(response, HttpServletResponse.SC_OK, responseContent);
            return;
        }

        // 4. We should never get here!
        Map<String, String> errorResponse = Map.of("error", "unexpected_error", "error_description",
                "Contact the service owner for details");
        OAuthUtils.setResponseBody(response, HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                mapper.writeValueAsString(errorResponse));
    }
}
```

- クライアントから送られてきたトークンリクエストのパラメーターを Authlete の /auth/token API にそのまま渡す
- Authlete の /auth/token API の `action` レスポンスパラメーターの値が `OK` の場合は `responseContent` をクライアントに返す
  - `responseContent` にはアクセストークンが含まれます

AUTHLETE

ステップ 3. トークンリクエストの処理を行う

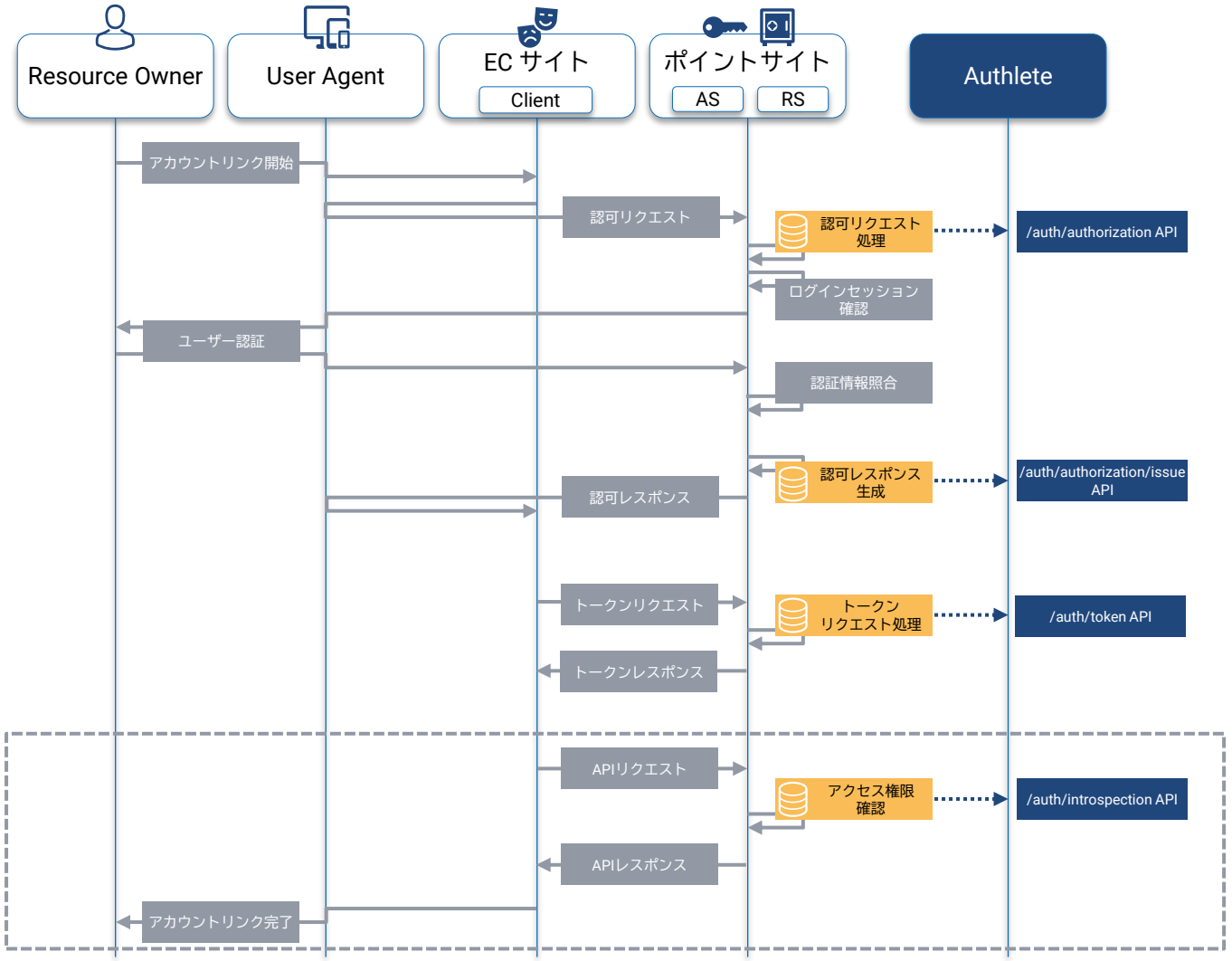
/java-getting-started/loyalty/src/main/java/com/authlete/simpleauth/oauth/OAuthUtils.java

```java
public static Map<String, Object> handleAuthleteApiCall(ServletContext context,
        HttpServletResponse response, String api, Map<String, Object> requestMap)
        throws IOException {
    logger.debug("Calling API {} with params {}", api, OAuthUtils.prettyPrint(requestMap));

    Map<String, Object> responseMap = getClient(context).target(AUTHLETE_BASE + api).request()
            .post(Entity.entity(requestMap, MediaType.APPLICATION_JSON_TYPE),
                    new GenericType<>() {});

    logger.debug("Received API response {}", OAuthUtils.prettyPrint(responseMap));

    String action = (String) responseMap.get("action");
    String responseContent = (String) responseMap.get("responseContent");

    switch (action) {
        case "INTERNAL_SERVER_ERROR":
            setResponseBody(response, HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    responseContent);
            return null;
        case "BAD_REQUEST":
        case "INVALID_CLIENT":
            setResponseBody(response, HttpServletResponse.SC_BAD_REQUEST, responseContent);
            return null;
        case "LOCATION":
            response.setStatus(HttpServletResponse.SC_FOUND);
            response.setHeader("Location", responseContent);
            response.setHeader("Cache-Control", "no-store");
            response.setHeader("Pragma", "no-cache");
            return null;
    }

    return responseMap;
}
```

- Authlete の /auth/token API から得た内容に応じたレスポンスをクライアントに返す
  - `action` レスポンスパラメーター
    - HTTP ステータスコードを決定する
  - `responseContent` レスポンスパラメーター
    - レスポンスボディに設定する

ステップ 4. API リクエストの処理を行う

- クライアントからの API リクエストに含まれるアクセストークンを取り出す
- Authlete の /auth/introspection API をコールする
  - クライアントから提示されたアクセストークンを渡す
- /auth/introspection API の結果に応じたレスポンスをクライアントに返す



ステップ 4. API リクエストの処理を行う

/java-getting-started/loyalty/src/main/java/com/authlete/simpleauth/oauth/OAuthFilter.java

```java
@Override
public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
        throws IOException, ServletException {
    HttpServletRequest request = (HttpServletRequest) req;
    HttpServletResponse response = (HttpServletResponse) resp;

    // 1. Is there an Authorization header with an access token?
    String authHeader = request.getHeader("Authorization");
    if (authHeader == null || !authHeader.startsWith(BEARER_SPACE)) {
        chain.doFilter(request, response);
        return;
    }

    // 2. Extract the bearer token from the header value and send it to the introspection
    // endpoint.
    String token = authHeader.substring(BEARER_SPACE.length());
    logger.info("Found access token {} - validating it with Authlete", token);
    Map<String, Object> authApiResponse =
            OAuthUtils.handleAuthleteApiCall(req.getServletContext(), response,
                    "/auth/introspection", Collections.singletonMap("token", token));

    if (authApiResponse == null) {
        // 3. There was an error calling the Authlete API. OAuthUtils.handleAuthleteApiCall has
        // handled it.
        // Nothing more to do here.
        logger.error("Disallowing API request with access token {}", token);
        return;
    }

    // 4. Attach the username to the request and pass the request down the chain
    String action = (String) authApiResponse.get("action");
    if (action.equals("OK")) {
        String username = (String) authApiResponse.get("subject");
        logger.info("Allowing API request to {} for user {}", request.getRequestURI(),
                username);
        chain.doFilter(new UserRequestWrapper(username, request), response);
        return;
    }
    // 5. We should never get here!
    Map<String, String> errorResponse = Map.of("error", "unexpected_error", "error_description",
            "Contact the service owner for details");
    OAuthUtils.setResponseBody(response, HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
            mapper.writeValueAsString(errorResponse));
}
```

- クライアントからの API リクエストに Authorization ヘッダーが含まれているか検証する
- Authorization ヘッダーからアクセストークンを取り出す
- Authlete の /auth/introspection API をコールする
  - `token` パラメーター
    - クライアントから提示されたアクセストークン
- Authlete の /auth/introspection API の `action` レスポンスパラメーターの値が `OK` の場合はアクセストークンに紐づくリソースオーナーのリソースをクライアントに返す

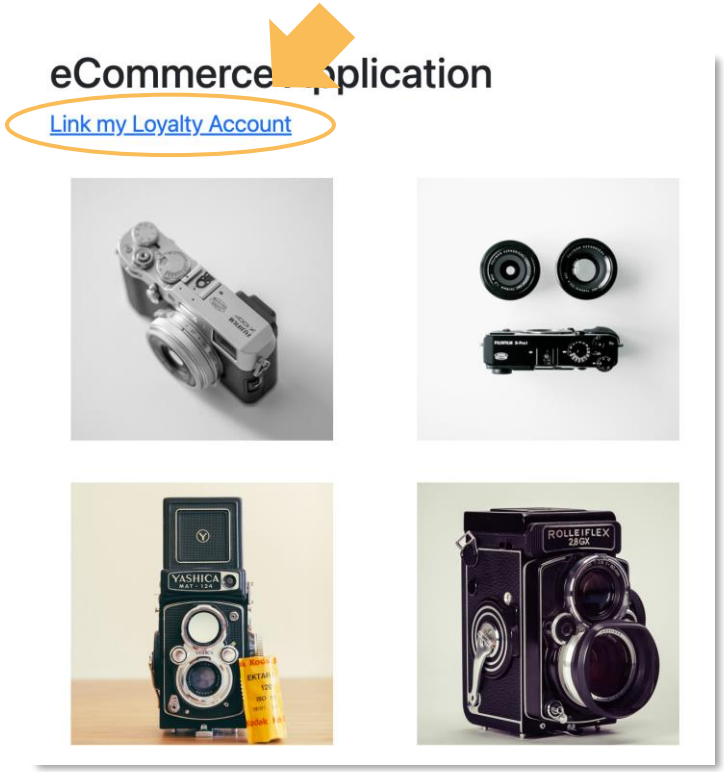
ステップ 4. API リクエストの処理を行う

/java-getting-started/loyalty/src/main/java/com/authlete/simpleauth/oauth/OAuthUtils.java

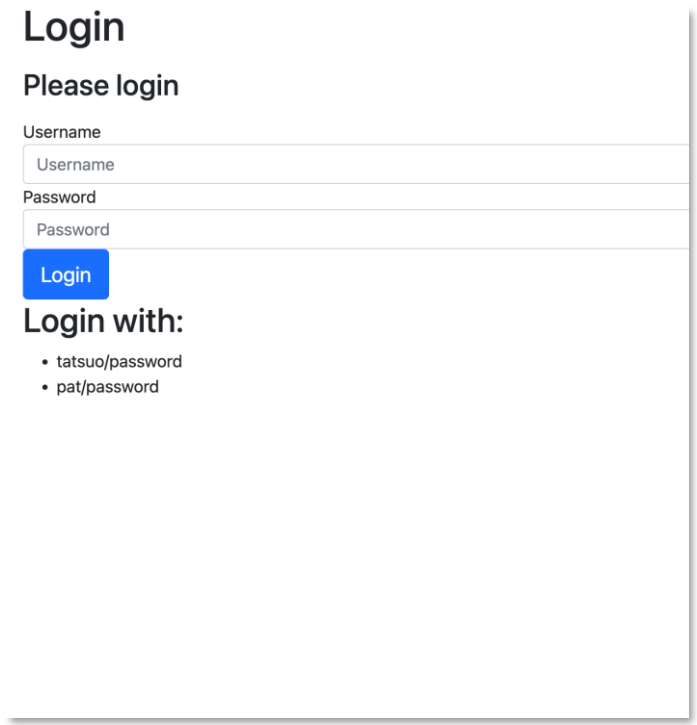
```java
public static Map<String, Object> handleAuthleteApiCall(ServletContext context, HttpServletResponse response,
                                                         String api, Map<String, Object> requestMap) throws
IOException {
    logger.debug("Calling API {} with params {}", api, OAuthUtils.prettyPrint(requestMap));

    Map<String, Object> responseMap = getClient(context).target(AUTHLETE_BASE + api)
            .request()
            .post(Entity.entity(requestMap, MediaType.APPLICATION_JSON_TYPE), new GenericType<>() {
            });

    logger.debug("Received API response {}", OAuthUtils.prettyPrint(responseMap));

    String action = (String)responseMap.get("action");
    String responseContent = (String)responseMap.get("responseContent");

    switch (action) {
        case "INTERNAL_SERVER_ERROR":
            setResponseBody(response, HttpServletResponse.SC_INTERNAL_SERVER_ERROR, responseContent);
            return null;
        case "BAD_REQUEST":
        case "INVALID_CLIENT":
            setResponseBody(response, HttpServletResponse.SC_BAD_REQUEST, responseContent);
            return null;
        case "UNAUTHORIZED":
            setAuthenticateHeader(response, HttpServletResponse.SC_UNAUTHORIZED, responseContent);
            return null;
        case "FORBIDDEN":
            setAuthenticateHeader(response, HttpServletResponse.SC_FORBIDDEN, responseContent);
            return null;
        case "LOCATION":
            response.setStatus(HttpServletResponse.SC_FOUND);
            response.setHeader("Location", responseContent);
            response.setHeader("Cache-Control", "no-store");
            response.setHeader("Pragma", "no-cache");
            return null;
        default:
            break;
    }

    return responseMap;
}
```

- Authlete の /auth/introspection API から得た内容に応じたレスポンスをクライアントに返す
  - `action` レスポンスパラメーター
    - HTTP ステータスコードを決定する
  - `responseContent` レスポンスパラメーター
    - レスポンスボディに設定する

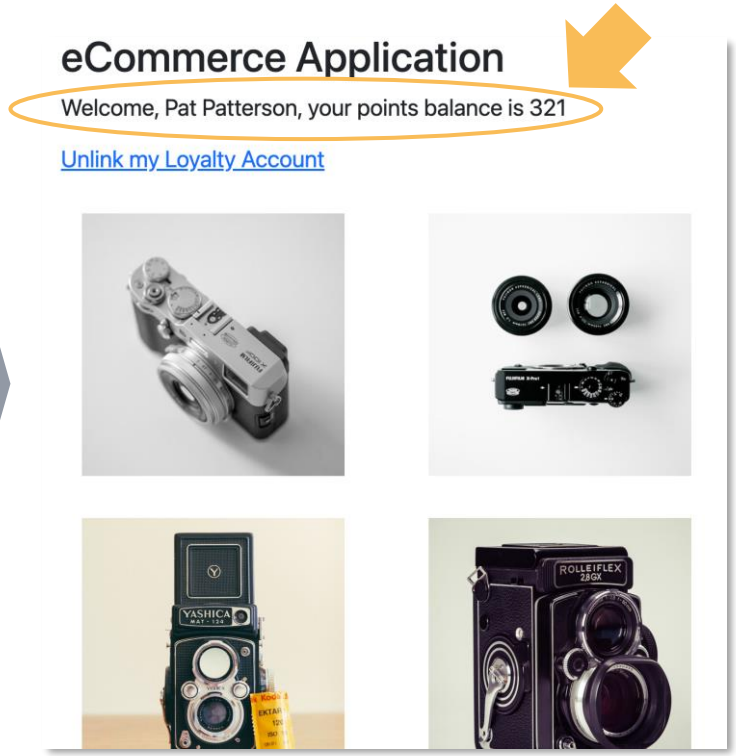
動作テスト



EC サイト



ポイントサイト



EC サイト

まとめ

- OAuth 関連のリクエストを Authlete API に渡し、API のレスポンスに対応する動作を追加するだけで認可サーバー/リソースサーバの実装が完了します

- OAuth の複雑な処理は Authlete におまかせ
  - OAuth クライアントからのリクエストの検証
  - OAuth クライアントへのレスポンスの組み立て
  - 発行後のトークン管理

- Authlete API を利用する前後でお客さま独自の処理を組み込むことも可能です

Thank You

www.authlete.com

info@authlete.com

Palo Alto, Tokyo, Dubai

AUTHLETE

AUTHLETE

We Are Hiring!!

https://www.authlete.com/ja/careers/