

🔍 91d2cbe70a ▾

...

node-oauth2-server / lib / grant-types / authorization-code-grant-type.js / <> Jump to ▾

 danbell Pass correct "this" to the model.

🕒 History

👤 4 contributors 

206 lines (167 sloc) | 5.82 KB

...

```

1  'use strict';
2
3  /**
4   * Module dependencies.
5   */
6
7  var AbstractGrantType = require('./abstract-grant-type');
8  var InvalidArgumentError = require('../errors/invalid-argument-error');
9  var InvalidGrantError = require('../errors/invalid-grant-error');
10 var InvalidRequestError = require('../errors/invalid-request-error');
11 var Promise = require('bluebird');
12 var promisify = require('promisify-any').use(Promise);
13 var ServerError = require('../errors/server-error');
14 var is = require('../validator/is');
15 var util = require('util');
16
17 /**
18  * Constructor.
19  */
20
21 function AuthorizationCodeGrantType(options) {
22   options = options || {};
23
24   if (!options.model) {
25     throw new InvalidArgumentError('Missing parameter: `model`');
26   }
27
28   if (!options.model.getAuthorizationCode) {
29     throw new InvalidArgumentError('Invalid argument: model does not implement `getAuthorizationCode()`');
30   }
31
32   if (!options.model.revokeAuthorizationCode) {
33     throw new InvalidArgumentError('Invalid argument: model does not implement `revokeAuthorizationCode()`');
34   }
35
36   if (!options.model.saveToken) {
37     throw new InvalidArgumentError('Invalid argument: model does not implement `saveToken()`');
38   }
39
40   AbstractGrantType.call(this, options);
41 }
42
43 /**
44  * Inherit prototype.
45  */
46
47 util.inherits(AuthorizationCodeGrantType, AbstractGrantType);
48
49 /**
50  * Handle authorization code grant.
51  *
52  * @see https://tools.ietf.org/html/rfc6749#section-4.1.3
53  */
54
55 AuthorizationCodeGrantType.prototype.handle = function(request, client) {
56   if (!request) {
57     throw new InvalidArgumentError('Missing parameter: `request`');
58   }
59
60   if (!client) {
61     throw new InvalidArgumentError('Missing parameter: `client`');
62   }
63
64   return Promise.bind(this)
65     .then(function() {
66       return this.getAuthorizationCode(request, client);
67     })
68     .tap(function(code) {
69       return this.validateRedirectUri(request, code);
70     })
71     .tap(function(code) {
72       return this.revokeAuthorizationCode(code);
73     })
74     .then(function(code) {
75       return this.saveToken(code.user, client, code.authorizationCode, code.scope);
76     });
77 };
78

```

```

79  /**
80  * Get the authorization code.
81  */
82
83  AuthorizationCodeGrantType.prototype.getAuthorizationCode = function(request, client) {
84    if (!request.body.code) {
85      throw new InvalidRequestError('Missing parameter: `code`');
86    }
87
88    if (!is.vschar(request.body.code)) {
89      throw new InvalidRequestError('Invalid parameter: `code`');
90    }
91    return promisify(this.model.getAuthorizationCode, 1).call(this.model, request.body.code)
92      .then(function(code) {
93        if (!code) {
94          throw new InvalidGrantError('Invalid grant: authorization code is invalid');
95        }
96
97        if (!code.client) {
98          throw new ServerError('Server error: `getAuthorizationCode()` did not return a `client` object');
99        }
100
101        if (!code.user) {
102          throw new ServerError('Server error: `getAuthorizationCode()` did not return a `user` object');
103        }
104
105        if (code.client.id !== client.id) {
106          throw new InvalidGrantError('Invalid grant: authorization code is invalid');
107        }
108
109        if (!(code.expiresAt instanceof Date)) {
110          throw new ServerError('Server error: `expiresAt` must be a Date instance');
111        }
112
113        if (code.expiresAt < new Date()) {
114          throw new InvalidGrantError('Invalid grant: authorization code has expired');
115        }
116
117        if (code.redirectUri && !is.uri(code.redirectUri)) {
118          throw new InvalidGrantError('Invalid grant: `redirect_uri` is not a valid URI');
119        }
120
121        return code;
122      });
123  };
124
125  /**
126  * Validate the redirect URI.
127  *
128  * "The authorization server MUST ensure that the redirect_uri parameter is
129  * present if the redirect_uri parameter was included in the initial
130  * authorization request as described in Section 4.1.1, and if included
131  * ensure that their values are identical."
132  *
133  * @see https://tools.ietf.org/html/rfc6749#section-4.1.3
134  */
135
136  AuthorizationCodeGrantType.prototype.validateRedirectUri = function(request, code) {
137    if (!code.redirectUri) {
138      return;
139    }
140
141    var redirectUri = request.body.redirect_uri || request.query.redirect_uri;
142
143    if (!is.uri(redirectUri)) {
144      throw new InvalidRequestError('Invalid request: `redirect_uri` is not a valid URI');
145    }
146
147    if (redirectUri !== code.redirectUri) {
148      throw new InvalidRequestError('Invalid request: `redirect_uri` is invalid');
149    }
150  };
151
152  /**
153  * Revoke the authorization code.
154  *
155  * "The authorization code MUST expire shortly after it is issued to mitigate
156  * the risk of leaks. [...] If an authorization code is used more than once,
157  * the authorization server MUST deny the request."
158  *
159  * @see https://tools.ietf.org/html/rfc6749#section-4.1.2
160  */
161
162  AuthorizationCodeGrantType.prototype.revokeAuthorizationCode = function(code) {
163    return promisify(this.model.revokeAuthorizationCode, 1).call(this.model, code)
164      .then(function(status) {
165        if (!status) {
166          throw new InvalidGrantError('Invalid grant: authorization code is invalid');
167        }
168
169        return code;
170      });
171  };
172
173  /**
174  * Save token.
175  */
176

```

```
177 AuthorizationCodeGrantType.prototype.saveToken = function(user, client, authorizationCode, scope) {
178   var fns = [
179     this.validateScope(user, client, scope),
180     this.generateAccessToken(client, user, scope),
181     this.generateRefreshToken(client, user, scope),
182     this.getAccessTokenExpiresAt(),
183     this.getRefreshTokenExpiresAt()
184   ];
185
186   return Promise.all(fns)
187     .bind(this)
188     .spread(function(scope, accessToken, refreshToken, accessTokenExpiresAt, refreshTokenExpiresAt) {
189       var token = {
190         accessToken: accessToken,
191         authorizationCode: authorizationCode,
192         accessTokenExpiresAt: accessTokenExpiresAt,
193         refreshToken: refreshToken,
194         refreshTokenExpiresAt: refreshTokenExpiresAt,
195         scope: scope
196       };
197
198       return promisify(this.model.saveToken, 3).call(this.model, token, client, user);
199     });
200 };
201
202 /**
203  * Export constructor.
204  */
205
206 module.exports = AuthorizationCodeGrantType;
```