JASONSTRIMPEL'S BLOG

Backbone.js and socket.io

Blog entry by JasonStrimpel on 18 Nov 2011

Backbone.js is a lightweight JavaScript MVC framework. The default transport mechanism for CRUD operations is XHR. Socket.io is a Node.js Web Socket server. This article provides an overview of using socket.io as the transport mechanism for Backbone.js CRUD operations via a custom Backbone.sync method.

Overview

Backbone.js is a lightweight JavaScript MVC framework that provides structure for front-end driven web applications. It connects to the back-end through a RESTful JSON interface via XHR. Backbone.js provides the ability to override the XHR default through a custom Backbone.sync method. For instance, Backbone.sync can use local storage (Web Storage), IndexedDB, Web Sockets, etc. The focus of this article will be the integration of Backbonse.js CRUD operations with socket.io a Node.js Web Socket server. This article assumes a basic knowledge of Backbone.js, Node.js, and socket.io.

Backbone.sync

When a model or collection is modified and a server save state operation is called (fetch, save, destroy, etc.) in Backbone.js Backbone.sync is called and a CRUD operation is executed against the backend.

Enough Show Me Some Code Already

In the code below the only item added to the default model or collection is this.ctx, which is optional. It is used to provide a context for the operation and should be defined upon initializing a Backbone model or collection. *Please read the comments. This is ONLY a starting point that can be expanded upon to meet your specific needs. Your mileage may vary. Use at your own risk. Author assumes no responsibility for any outcomes good, bad, or ugly. This is a discovery process. Please comment with gentle critiques or recommendations.*

```
Backbone.sync = function (method, model, options) {
    var socket = window.NAMESPACE.socket; // grab active socket from global namespace; io.connect() w

    /*
     * Create signature object that will emitted to server with every request.
     * This is used on the server to push an event back to the client listener.
     */
    var signature = function () {
        var sig = {};

        sig.endPoint = model.url + (model.id ? ('/' + model.id) : '');
        if (model.ctx) sig.ctx = model.ctx;

        return sig;
    };

    /*
     * Create an event listener for server push. The server notifies
     * the client upon success of CRUD operation.
     */
    var event = function (operation, sig) {
        var e = operation + ':';
```

JASONSTRIMPEL'S ARCHIVE

September 2012	1 post
July 2012	1 post
June 2012	2 posts
May 2012	2 posts
April 2012	1 post
March 2012	1 post
January 2012	1 post
December 2011	1 post
November 2011	3 posts

JASONSTRIMPEL'S BLOG STATS

Blogging since	November 2011
Number of posts	13
Number of comments	20
Number of views	145945
Most popular tags	viewpoint, javascript, amd

```
        e += sig.endPoint;
        if (sig.ctx) e += (':' + sig.ctx);

        return e;
    };

    // Save a new model to the server.
    var create = function () {
        var sign = signature(model);
        var e = event('create', sign);
        socket.emit('create', {'signature' : sign, item : model.attributes });
        socket.once(e, function (data) {
            model.id = data.id;
            console.log(model);
        });
    };

    // Get a collection or model from the server.
    var read = function () {
        var sign = signature(model);
        var e = event('read', sign);
        socket.emit('read', {'signature' : sign});
        socket.once(e, function (data) {
            options.success(data); // updates collection, model; fetch
        });
    };

    // Save an existing model to the server.
    var update = function () {
        var sign = signature(model);
        var e = event('update', sign);
        socket.emit('update', {'signature' : sign, item : model.attributes }); // model.attribues is
        socket.once(e, function (data) {
            console.log(data);
        });
    };

    // Delete a model on the server.
    var destroy = function () {
        var sign = signature(model);
        var e = event('delete', sign);
        socket.emit('delete', {'signature' : sign, item : model.attributes }); // model.attribues is
        socket.once(e, function (data) {
            console.log(data);
        });
    };

    // entry point for method
    switch (method) {
        case 'create':
            create();
            break;
        case 'read':
            read();
            break;
        case 'update':
            update();
            break;
        case 'delete':
            destroy();
            break;
    }
};
```

Node.js, socket.io Stub

```
/*
 * This is a stub for a socket.io server that responds to CRUD operations
 */
var io = require('socket.io').listen(3000);

var create = function (socket, signature) {
    var e = event('create', signature), data = [];
    socket.emit(e, {id : 1});
};

var read = function (socket, signature) {
    var e = event('read', signature), data;
    data.push({})
    socket.emit(e, data);
};

var update = function (socket, signature) {
    var e = event('update', signature), data = [];
    socket.emit(e, {success : true});
};

var destroy = function (socket, signature) {
    var e = event('delete', signature), data = [];
    socket.emit(e, {success : true});
};

// creates the event to push to listening clients
var event = function (operation, sig) {
    var e = operation + ':';
    e += sig.endPoint;
    if (sig.ctx) e += (':' + sig.ctx);

    return e;
};

io.sockets.on('connection', function (socket) {
    socket.on('create', function (data) {
        create(socket, data.signature);
    });
    socket.on('read', function (data) {
        read(socket, data.signature);
    });
    socket.on('update', function (data) {
        update(socket, data.signature);
    });
    socket.on('delete', function (data) {
        destroy(socket, data.signature);
    });
});
```

Conclusion

There is nothing amazing going on in the code above. The main point is that it is loosely coupled. The Backbone models and collections are completely ignorant of socket.io. If Backbone model and collection url properties follow a common method mapping such as RFC 2616 then the Backbone.sync code above can be removed with none or minimal front-end refactoring. The transport mechanism will fall back to XHR or another custom Backbone.sync method can be written. Again, this has not been fully fleshed out, but it is a good starting point for using socket.io with Backbone.js.

DISCUSSION

06 Jan 2013

It is very usefull information. Thans / BIK

BIK
1 comment
Joined 01/13