Using SDEverywhere to Make a Vensim Model into a Web Application

Revised: 2019-07-24

This tutorial shows you how to take your Vensim model and turn it into an interactive web application using the open-source SDEverywhere toolkit. SDEverywhere currently requires the macOS operating system in development, but the resulting web app can be run on any web server, or even offline using the Firefox browser.

Getting started

Install SDEverywhere

First install SDEverywhere using the instructions in the "Installing" section of the README.

Install Emscripten

The Emscripten toolchain converts the C code generated by SDEverywhere into JavaScript, and then compiles it into WebAssembly that runs in a browser.

1. Install the Java JDK. Emscripten uses Java to run the Google Closure compiler, which shrinks generated JavaScript code size considerably. Download and install the JDK from the Oracle download page. Be sure to install the JDK and not the JRE.

2. Install Python 3. The easiest way to get the necessary SSL certificates for downloading Emscripten tools from GitHub using the Emscripten SDK is to install Python 3 with the pyenv Python version manager. Follow the instructions for installing pyenv using Homebrew.

3. Clone the Emscripten SDK from GitHub.

```
git clone https://github.com/emscripten-core/emsdk.git
```

4. Install and activate the Emscripten SDK using the upstream LLVM backend.

```
cd emsdk
./emsdk install latest-upstream
./emsdk activate latest-upstream
```

5. You will need to run the following command from the `emsdk` directory every time you want to use Emscripten —that is, every time you generate web apps with SDEverywhere. This will temporarily use the Node and clang

compiler bundled with Emscripten. When you close the terminal tab, the Emscripten environment will go away.

```
source ./emsdk_env.sh
```

6. Test your Emscripten installation.

```
emcc -v
```

Generating model code and validating it

The first step generates C code for your model and validates it against a Vensim run. This is necessary to ensure that SDEverywhere can handle all the Vensim constructs in your model and that it generates correct code for your equations. In a later step, the C code will be converted to JavaScript code that will be embedded in your web app.

Create a model directory.

Copy the model `.mdl` file into the model directory using a short, lower-case name, since you will be typing it in SDEverywhere commands. The placeholder for the model name (without the `.mdl` extension) in these instructions is `{model}`.

Run the model in Vensim using `{model}` as the run name.

Export the `.vdf` run in Vensim DAT format using Model > Export Dataset.

Generate C code, compile and run it, and validate the results against Vensim.

```
sde test {model}
```

SDEverywhere will list discrepancies between the Vensim run and data generated by the C model, up to a default precision of 10^{-5}. If you are getting errors, and somewhat less precision, such as 10^{-3}, is acceptable, run the test with the `-p` option.

```
sde test -p 1e-3 {model}
```

Designing your web app

SDEverywhere generates a web application based on a standard template merged with app configuration in CSV files. Refer to the *SDEverywhere Web App Configuration Reference* document for details. There is a fully worked-out example in the `models/sir` directory.

Copy the `config` directory from the `notes` directory to your model directory. This will give you empty CSV files with the proper headers.

If you use Excel to edit the CSV files, open them from the Finder instead of from within Excel. This will avoid the Text Import Wizard.

Generating the web app

The development scripts assume that your model is in the `model` subdirectory of your project directory. The CSV config files go in the `model/config` directory. You can run the scripts from the project directory or any of its subdirectories.

1. Activate the Emscripten SDK in your `emsdk` directory (see Install Emscripten above).

```
source ./emsdk_env.sh
```

2. Generate WebAssembly code for the model and embed it in a web app.

```
cd model
sde generate --genhtml {model}
```

3. Start the development web server, and then open the web app with the URL it prints.

```
npx http-server build/web -c-1 -o
```

If you are using your own web server, configure it to serve files from the `model/build/web` directory.

The files to deploy to a web server in production are found in the `model/build/web` directory.