WeatherForecast, REST Learning And Simulation In Android Appium Test

A demonstration of the REST backend service learning and simulation on Android.

This demo uses the Forecastie open source application using OpenWeatherMap REST API.

Development and testing of such application involves testing the application with various weather conditions which may be hard to achieve using real data. The simulation helps to provide different weather conditions including extremes at one moment.

There are 3 ways how to experience this demo:

1. **Manual Setup And Out-of-box Simulation**
   Use the ready-made learned simulation model to demonstrate the simulation of the application's backend service (https://openweathermap.org/).
2. **Learning The Simulation Models**
   Walk through the learning of the current weather conditions at a custom location, editing and running the simulation.
3. **Automatic Test Setup and Simulation**
   Use *Service Virtualization* together with *Mobile Center* to run an automatic *JUnit* test of the application using *Maven*.

Prerequisites

- A smartphone with *Android 5.0 Lollipop* or newer connected to WiFi
- Mobile Center (for all 3 demo cases) or ADB tool (manual setup/simulation and learning demo only)
- Mobile Center access key for execution created in *Settings/Access Keys* in Mobile Center

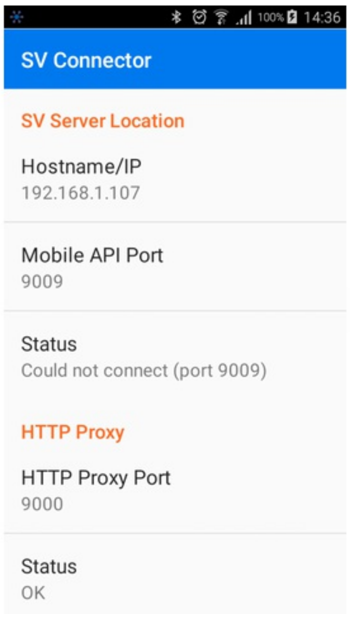
Step 1: Installing The Forecastie Application

Download the *"Forecastie - Weather app"* from F-Droid to disk and upload to Mobile Center. Install the packaged version of application on Android 7.0+ (because of server certificate trust instrumentation). You can use non-packaged version on older Android versions.

Step 2: Mobile Phone Configuration

Now you need to configure the *Android* phone to connect to your PC running the simulation. Most of the configuration is done automatically once you connect the phone to Mobile Center (the SV integration must be enabled for that particular phone in `Mobile Center/server/conf/connector.properties`, see Mobile Center documentation for more information).

You can verify this step by locating the SV Connector Configuration utility installed on the phone by Mobile Center and checking the connection to HTTP Proxy connector:


SV Connector

SV Server Location

Hostname/IP
192.168.1.107

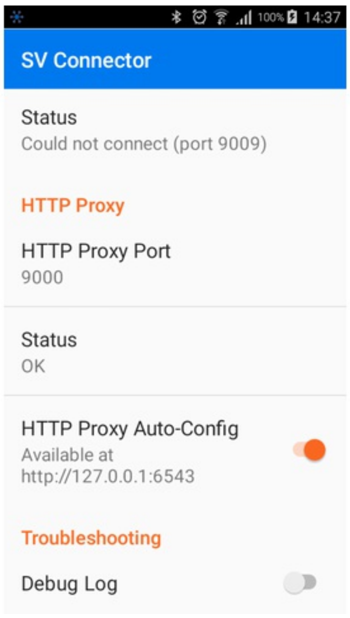
Mobile API Port
9009

Status
Could not connect (port 9009)

HTTP Proxy

HTTP Proxy Port
9000

Status
OK


SV Connector

Status
Could not connect (port 9009)

HTTP Proxy

HTTP Proxy Port
9000

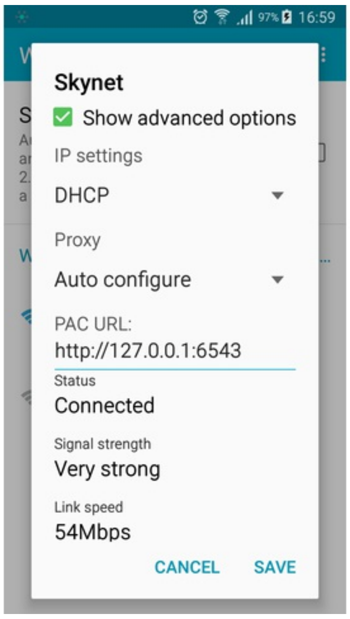
Status
OK

HTTP Proxy Auto-Config
Available at
http://127.0.0.1:6543

Troubleshooting

Debug Log

To virtualize REST services on Android phone, you need to make it talk to the the *HTTP proxy connector*. You achieve this by configuring proxy settings of your current WiFi connection on the phone. Within the WiFi settings, enable proxy selecting *Auto-config* and set the *PAC web address* to `http://127.0.0.1:6543`:


Skynet

✓ Show advanced options

IP settings
DHCP

Proxy
Auto configure

PAC URL:
http://127.0.0.1:6543

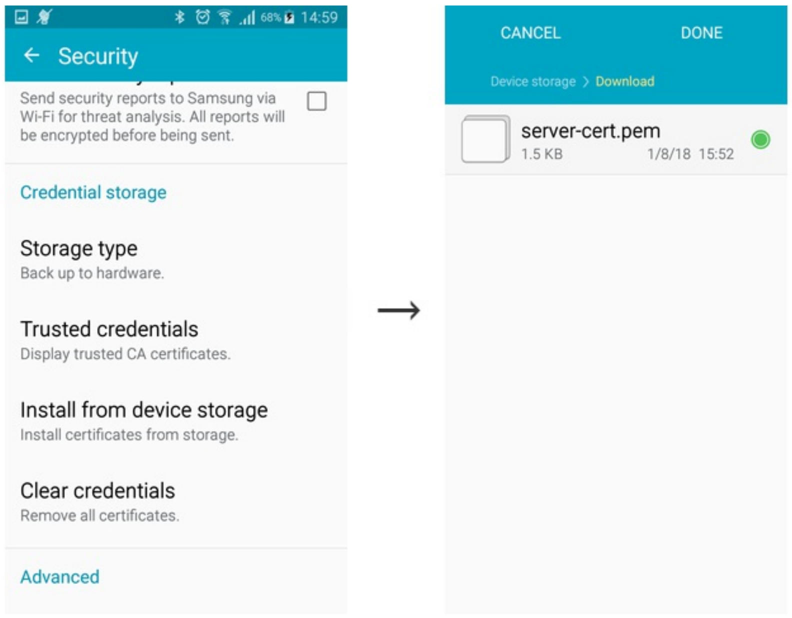
Status
Connected

Signal strength
Very strong

Link speed
54Mbps

To enable virtualization of services over HTTPS, import the `SV_LAB_HOME/bin/sv-capture.pem` certificate into *User Certificates* on the phone. Choose to use the certificate for *VPN and apps* when asked.

- One way is to upload the file over USB cable and go to *Settings/Security/Credential storage* and perform *Install from device storage*.
- Another option is to send the certificate by e-mail to phone account and open the attachment in e-mail app.


Security

Send security reports to Samsung via Wi-Fi for threat analysis. All reports will be encrypted before being sent.

Credential storage

Storage type
Back up to hardware.

Trusted credentials
Display trusted CA certificates.

Install from device storage
Install certificates from storage.

Clear credentials
Remove all certificates.

Advanced

CANCEL DONE

Device storage > Download

server-cert.pem
1.5 KB 1/8/18 15:52

To verify the settings, launch a discovery mode using sv-capture tool, open a web browser in your smartphone and try to display some web page, e.g. https://openweathermap.org/. Provide your Mobile Center server URL, access key and mobile phone name with `-mn` argument or mobile phone ID with `-mi` parameter:

```
../../bin/sv-capture.sh -ms http://localhost:8080 -ma "client=oauth2-..." \
                        -mn SM-G800F -d
```

In Windows:

```
..\..\bin\sv-capture.bat -ms http://localhost:8080 -ma "client=oauth2-..." ^
                         -mn SM-G800F -d
```

The page should display in browser and you should see the endpoint printed out by the *sv-capture* tool.

Now **stop the virtual lab pressing the <Enter> key** in the console window where you've run the *sv-capture* tool.

Manual Setup And Out-of-box Simulation

Lets follow this high level workflow for a quick simulation of the backend service to test the application with specific weather conditions. There is a ready-to-run simulation model bundled with the demo in so the only thing you need to do is to launch the virtual lab.

Simulate the Backend Service

To run the backend service simulation using an out-of-box model, run:

```
../../bin/sv-capture.sh -ms http://localhost:8080 -ma "client=oauth2-..." \
             -mn SM-G800F \
             -m SIMULATE -vsl src/test/resources/demo -as capture \
             -r https://api.openweathermap.org \
```

In Windows:

```
..\..\bin\sv-capture.bat -ms http://localhost:8080 -ma "client=oauth2-..." ^
             -mn SM-G800F ^
             -m SIMULATE -vsl src/test/resources/demo -as capture ^
             -r https://api.openweathermap.org
```

Open the *Forecastie* app on your smartphone and tap on the *Refresh button*. Now you should see simulation logs appearing on your console running *sv-capture* tool.

You can modify the service model `src/test/resources/demo/WeatherServiceModel.js` and restart the simulation to simulate different weather conditions.

Learning The Backend Service Simulation Model

Start learning by running:

```
../../bin/sv-capture.sh -ms http://localhost:8080 -ma "client=oauth2-..." \
             -r https://api.openweathermap.org -o ./learned-model
```

In Windows:

```
..\..\bin\sv-capture.bat -ms http://localhost:8080 -ma "client=oauth2-..." ^
             -r https://api.openweathermap.org -o learned-model
```

Then open the *Forecastie* app on your smartphone and tap on the *Refresh* button. Alternatively, you can change your location to display a forecast for another location. The messages to backend services are being recorded as you can see in the *sv-capture* console output.

Now stop the learning by pressing *Enter* within the console running the *sv-capture* tool.

The learned model is located witihn the `learned-model` directory. You can check and modify learned scenarios and data.

Simulate the learned model by running:

```
../../bin/sv-capture.sh -ms http://localhost:8080 -ma "client=oauth2-..." \
             -m SIMULATE -vsl ./learned-model -as capture \
             -r https://api.openweathermap.org
```

In Windows:

```
..\..\bin\sv-capture.bat -ms http://localhost:8080 -ma "client=oauth2-..." ^
             -m SIMULATE -vsl ./learned-model -as capture ^
             -r https://api.openweathermap.org
```

Restart the *Forecastie* app and repeat the actions against the simulation using the learned model.

Automatic Test Setup and Simulation

The *SV Lab* allows you to develop an automatic UI test of your *Android* application which can run within CI environment like Jenkins or TeamCity. In such environment, there are typically more build agents running on different machines using different hostnames or IP addresses.

The *Digital Lab* Java library enables automatic setup of a test environment so that the tested app on the mobile phone talks to the correct instance of a running virtual lab.

In previous steps, you had to enter the IP address of the machine running the _SV Lab Server_ manually, which is not suitable for the automatic test environment. When you use the _Digital Lab_ to configure your test environment, it reconfigures the _SV Connector_ on the phone automatically according to the contents of the _Digital Lab_ configuration file:

```
DigitalLabConfig digitalLabConfig = DigitalLabConfig.fromURL(
    new URL("file:src/test/resources/digital-lab.json")
);
DigitalLab digitalLab = new DigitalLab(
    digitalLabConfig,
    "file:/myvsldir/*"
);
```

The class *DigitalLab* provides also methods allowing to modify its configuration in runtime if necessary:

```
digitalLabConfig
    .getSvLab()
    .setServerUrl("https://my.svlab.server:8445/api");
```

Prerequisites

- Mobile Center installed and running
    - A connected and available Android phone
- Maven tool
- The *SV Lab Server* must be stopped before running the test

Running The Automated Test

Edit the demo.properties file and enter valid values for your environment:

- `mc.url` is a URL of the *Mobile Center* server.
- `mc.oauthClientId`, `mc.oauthClientSecret` and `mc.tenantId` are the components of execution access key generated in Mobile Center *Settings/Access Keys*
- please use the `MC_DEFAULT_WORKSPACE_NAME` value for the `mc.mcWorkspaceName` property

Then run `mvn test` within the demo directory.

Source Code

```
.
|  pom.xml ..................................... Maven automatic test project file
|  demo.properties ............................. automatic test property file
\--src
   +--main
   |  \--resources
   |     \--demo                                      the ready-to-run
   |           WeatherApplicationModel.js ........... 'capture' scenario
   |           WeatherServiceInterface.js ........... service interface...
   |           WeatherServiceInterfaceSwagger.json .. and the Swagger description
   |           WeatherServiceModel.js ............... service scenarios
   \--test
      +--java
      |  \--demo
      |          WeatherForecastTest.java ................ automcatic JUnit test
      \--resources
            digital-lab.json ........................... Digital Lab configuration
```