**Overview**

The used case in the proposed IEM framework involves several hydrological models: Precipitation Model, Hargreaves model, and TOPMODEL. The Precipitation Model, Hargreaves model are time-dependent model that will generate runoff on an area. The primary model used to generate runoff is the Topography-based hydrological MODEL (TOPMODEL), which is a continuous simulation model and has been applied in various regions throughout the world. The model takes the daily precipitation, daily evapotranspiration, and topographic index as inputs, and outputs daily runoff. In the case, we wrapped TOPMODEL as a Http and WebSocket service separately.

**Key Class/Interface**

**OpenMIWebSocketClient:**

* + **cn.edu.whu.openmi.models.data.PrecipitationEngine**: Precipitation model (OpenMI component). Initialize method is used to read the daily precipitation; performTimeStep method is to wait TOPMODEL pull the daily precipitation via getValues.
  + **cn.edu.whu.openmi.models.Hargreaves.HargreavesEngine:** Hargreaves model (OpenMI component). Initialize method is used to read the daily temperature data and latitude data; perfromTimeStep method is used to calculate the daily evapotranspiration and to wait TOPMODEL to pull; finish method is used to write the daily evapotranspiration to local file system.
  + **cn.edu.whu.httpmodel.HttpTopModelEngine:** The HTTP client of TOPMODEL (OpenMI component). Initialize method is used to initialize some time-independent parameters and established the connection with these parameters, then get a TOPMODEL id from the response; performTimeStep method is used to pull the daily precipitation and evapotranspiration from another two models, then send a request with the data to HTTP server of TOPMODEL.

(**note:** cn.edu.whu.openmi.util.RequestMethodStore: Establish HTTP connection and send some parameters, the HTTP connection way can be changed to short-alive or long-alive by modify the connection argument “connection”)

* + **cn.edu.whu.wssync.WSSyncTopModelEngine:** The WebSocket client of TOPMODEL (OpenMI component). Initialize method is used to initialize some time-independent parameters and established a WebSocket connection with these kvp; performTimeStep method is used to pull the daily precipitation and evapotranspiration from another two models, then send a request with the data to WebSocket server of TOPMODEL.

(**note**: WebSocket follows asynchronous mechanism, which means the client send request and doesn’t wait the response from server. But the model computation flow requires that the next step computation should wait until the client receives the results. To solve the inconsistency, the thread mechanism is used. The main thread in the client is used to perform time steps, and the other thread is used to send and receive messages.)

**OpenMIWebSocketServer: (Tomcat/8.0 and servlet is used)**

* + **cn.edu.whu.openmi.http.HttpTopmodel:** The HTTP Server of TOPMODEL. (which is wrapped with OpenMI and Servlet). Initialize method is used to get the time-independent parameters sent by client via URL, performTimeStep is used to get the daily precipitation and evapotranspiration, then calculate the daily runoff and return to client.

(**note**: For maintaining the state of the model, we used UUID to identify the model and save some initial parameters)

* + **cn.edu.whu.openmi. websocket2.WSTopModel2:** There are three event handlers that must be supported by the server to implement the WebSocket API: onOpen, onMessage, and onClose. Which match the three phases of OpenMI component (Fig. 1). The onOpen method is used to confirm the connecting request from the client. In this method, the initialize phase of the OpenMI model on the server can be invoked, where some time-independent values are set. The onMessage method is used for the time-step based computation. The data exchange during each time step will invoke the onMessage method, which in turn drive the run phase of the OpenMI component and calculate the daily runoff. The onClose method is used to close the connection between the client and the server. It will trigger the finish phase of the OpenMI component



Fig. 1 Wrapping OpenMI components as WebSocket services.

**Execution portal (In OpenMIWebSocketClient)：**

* **cn.edu.whu.openmi.models.topmodel.TopmodelLocalMainTest**: TOPMODEL case in local environment.
* **cn.edu.whu.httpmodel. HttpTopmodelMainTes**t: TOPMODEL case in HTTP approach.
* **cn.edu.whu.wssync. WSSyncMainTest**：TOPMODEL case in WebSocket approach.