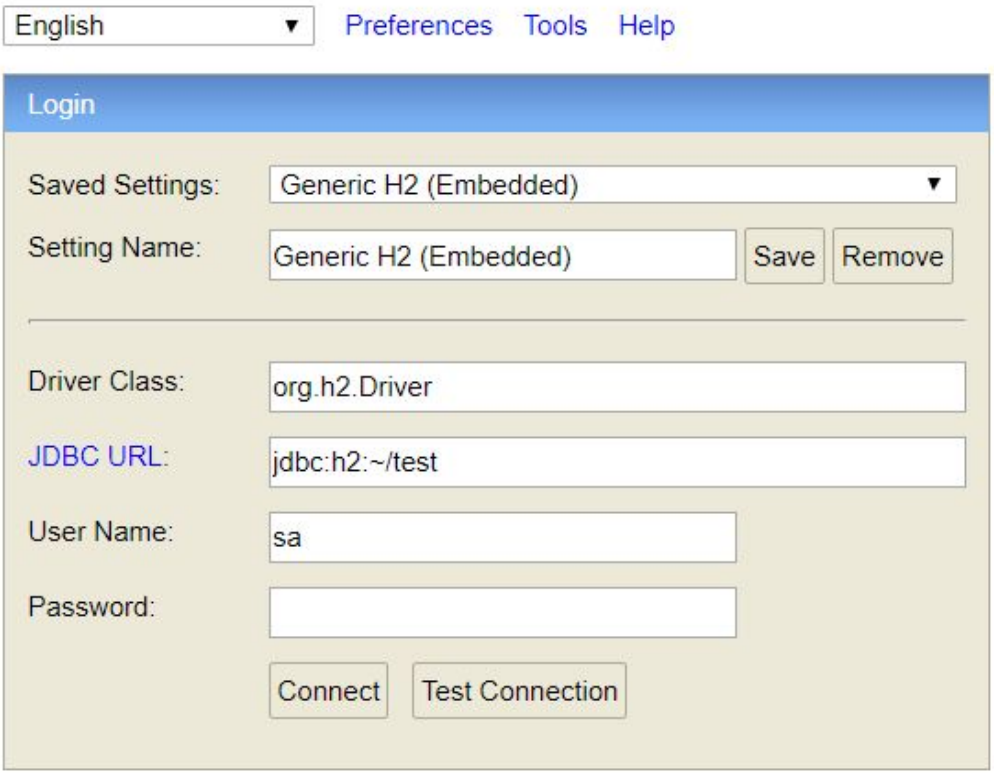
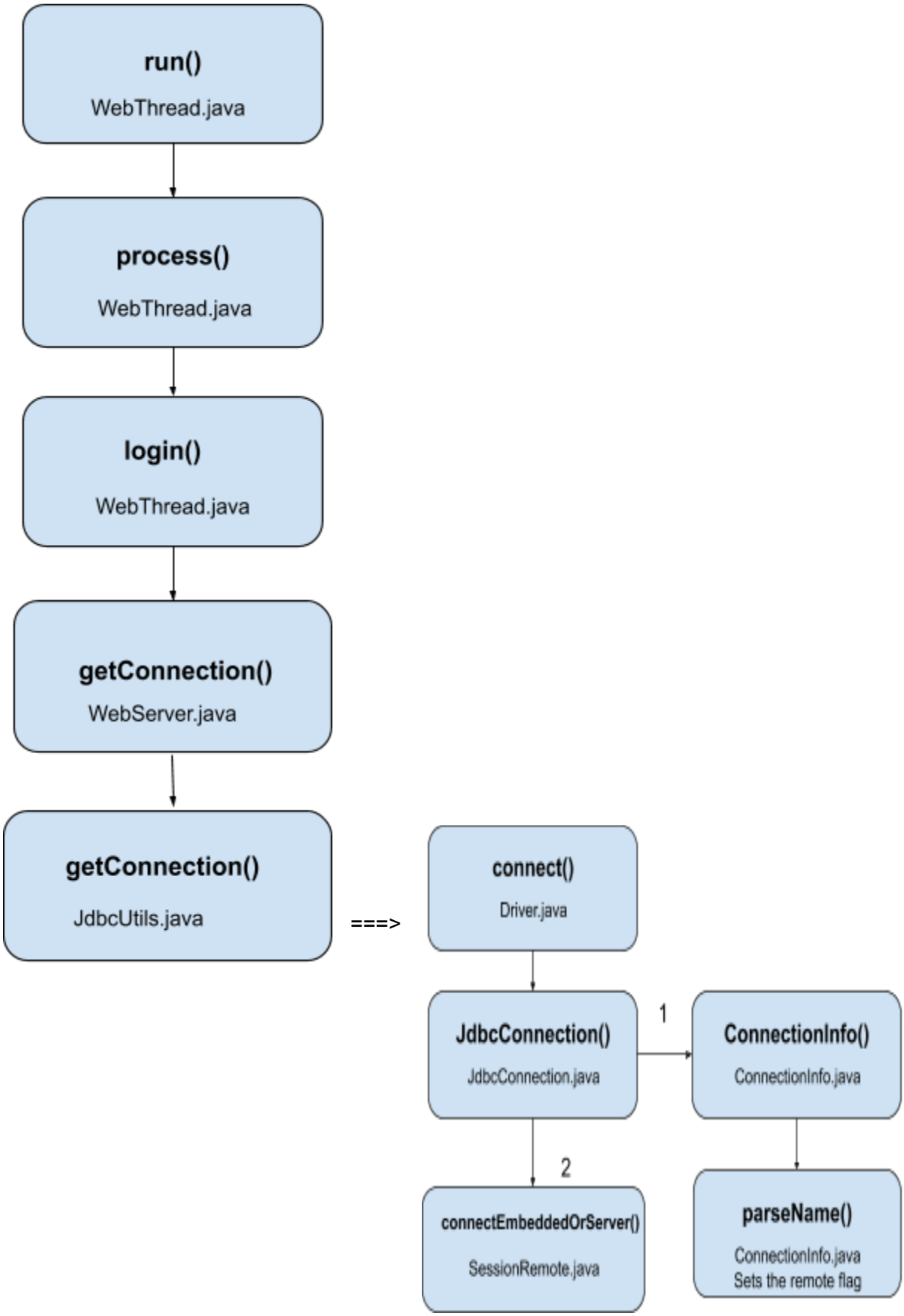
Feature 1: How does the h2 database support Embedded and Server mode?

One of the primary features of the h2 database is that it supports both Embedded and Server modes.
First, in the login page, we have the option to select server connection or embedded from the Settings dropdown. In this case, we have selected the Embedded option.



Once we click on the Connect method it calls the run() method in WebThread.java. It then calls the process() method from the class to process the request, which calls the login() method from the same class. It then creates and passes the connection info object, which contains the user name, password, driver details, and other related information that was present on the UI. It calls the getConnection() method from the WebServer.java class that in turn calls the getConnection() method from the JdbcUtils.java class. In that method, connect() from Driver.java is called that creates an object of JdbcConnection.java. This calls its constructor where an object of ConnectionInfo is created. Hence the constructor of ConnectionInfo is called which executes the function parseName() from the same class. This class parses the connectionName. If the connectionName contains SSL or TCP it sets the remote flag on. After the creation of the ConnectionInfo object we call the connectEmbeddedOrServer() method from the SessionRemote.java class. This method will execute connectServer() or embedded based on the remote flag. If the remote flag is set, then it makes a server connection else embedded connection

Basic Code Flow:



```
/**
* Open a new (remote or embedded) session.
*
* @param openNew whether to open a new session in any case
* @return the session
*/
public SessionInterface connectEmbeddedOrServer(boolean openNew) {
  ConnectionInfo ci = connectionInfo;
  if (ci.isRemote()) {
    connectServer(ci);
    return this;
  }
  // create the session using reflection,
  // so that the JDBC layer can be compiled without it
  boolean autoServerMode = ci.getProperty("AUTO_SERVER", false);
  ConnectionInfo backup = null;
  try {
    if (autoServerMode) {
      backup = ci.clone();
      connectionInfo = ci.clone();
    }
    if (openNew) {
      ci.setProperty("OPEN_NEW", "true");
    }
    if (sessionFactory == null) {
      sessionFactory = (SessionFactory) Class.forName(
          "org.h2.engine.Engine").getMethod("getInstance").invoke(null);
    }
    return sessionFactory.createSession(ci);
  } catch (Exception re) {
    DbException e = DbException.convert(re);
    if (e.getErrorCode() == ErrorCode.DATABASE_ALREADY_OPEN_1) {
      if (autoServerMode) {
        String serverKey = ((JdbcException) e.getSQLException()).getSQL();
        if (serverKey != null) {
          backup.setServerKey(serverKey);
          // OPEN_NEW must be removed now, otherwise
          // opening a session with AUTO_SERVER fails
          // if another connection is already open
          backup.removeProperty("OPEN_NEW", null);
          connectServer(backup);
          return this;
        }
      }
    }
    throw e;
  }
}
```