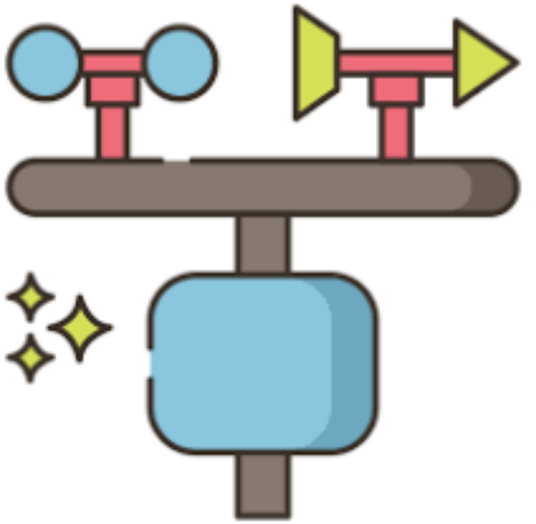
WeatherTop



A detailed walkthtough of a Java/Play WeatherTop implementation

Assignment Concept

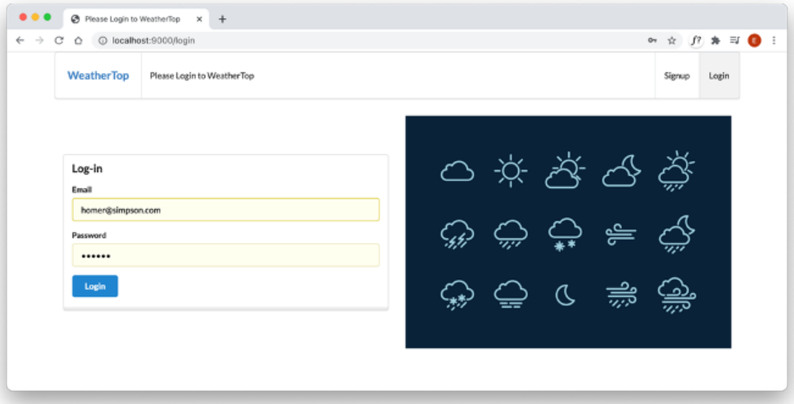
- Owners of a consumer level weather station purchase and install a weather station kit.

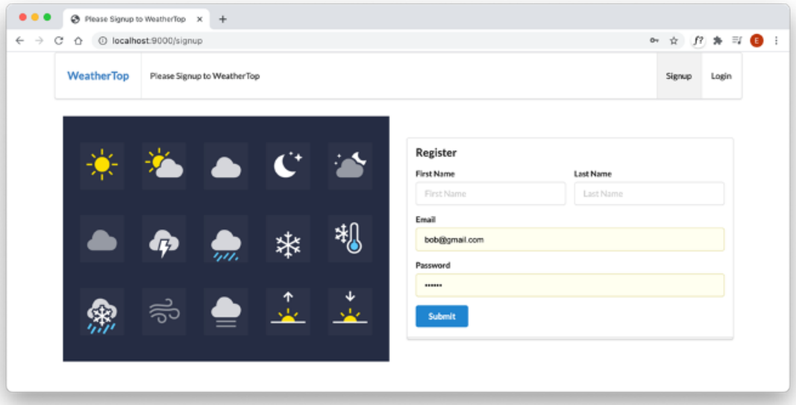
- Periodically members submit weather reports from their station to a web site, capturing readings from the station at a specific time

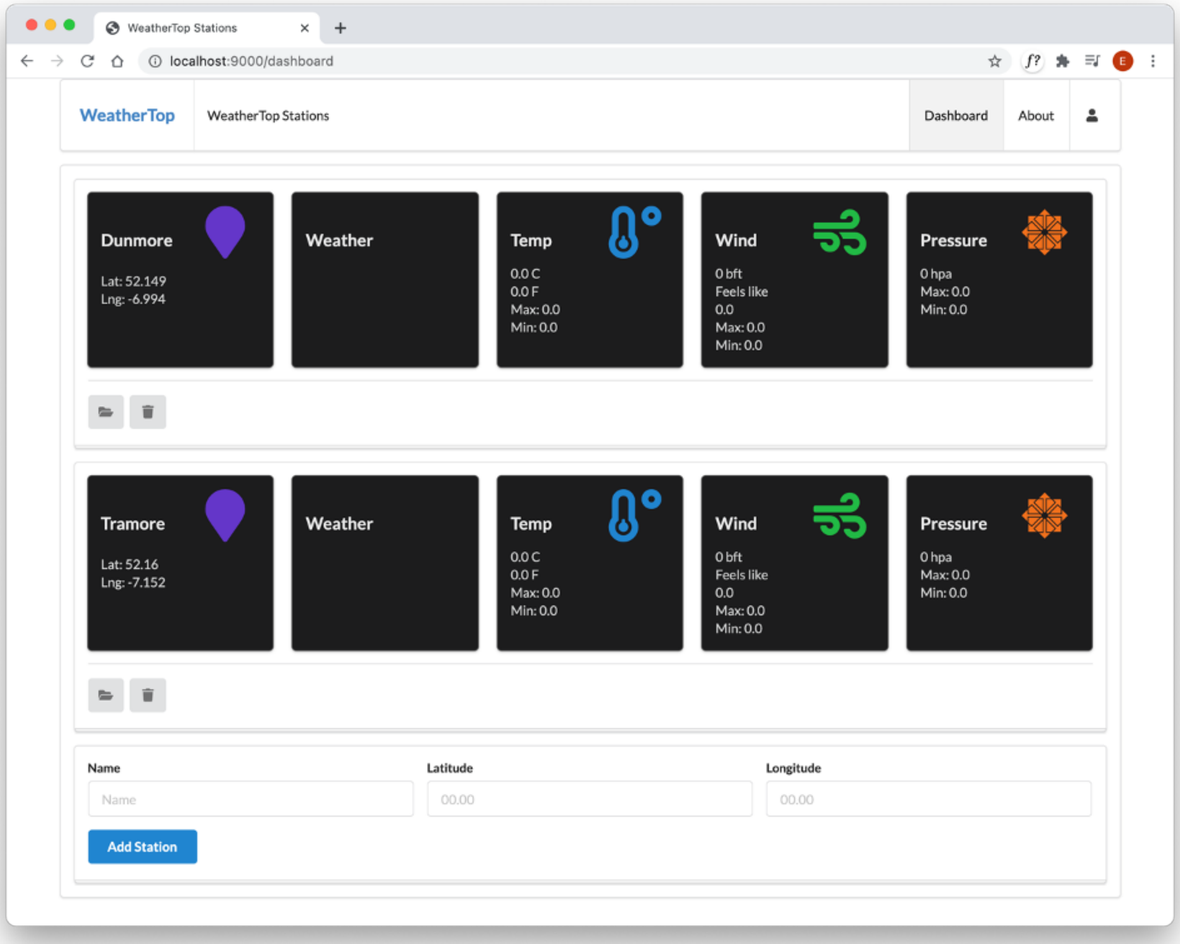
- The application displays weather analytics for the station

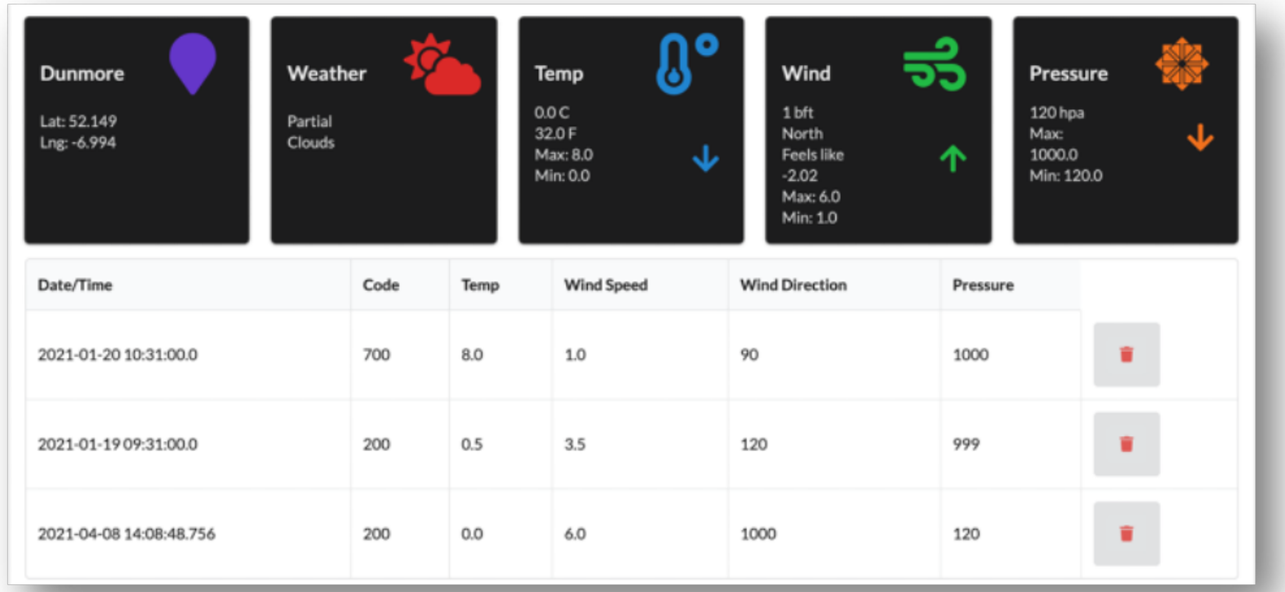
- The owner may own multiple stations

	Reading	Station	Member	Features	Code
Starter	Code Temp Wind Speed	Station Name	None	Load and display stations + their readings from Yaml file	Zipped archive
Baseline	+ Pressure	+ Latest weather, Temp C, F, Wind Bft, pressure	None	+ display latest weather for station	Zipped archive + Readme
Release 1	+ Wind Direction	+ Wind Chill, Wind Compass	None	Dashboard shows station list + button to open station view. Include forms to add new Station + new reading	Github repo
Release 2		+Lat, Lng Max/Min (Temp, Wind, Pressure)	First Name, Last Name, Email, Password + Stations	Members can signup/log in. Members may create any number of weather stations. Members + sample stations + readings loaded from YAML	Deployed + Github repo + history
Release 3	+ Time/Date	Temp, Wind Pressure Trends	User can edit their personal details.	Member dashboard list summary lists latest conditions for all stations. (alphabetically). Members can delete reports or stations	Deployed + Github repo + history tags)


Please Login to WeatherTop
localhost:9000/login
WeatherTop
Please Login to WeatherTop
Signup
Login
Log-in
Email
homer@simpson.com
Password
Login

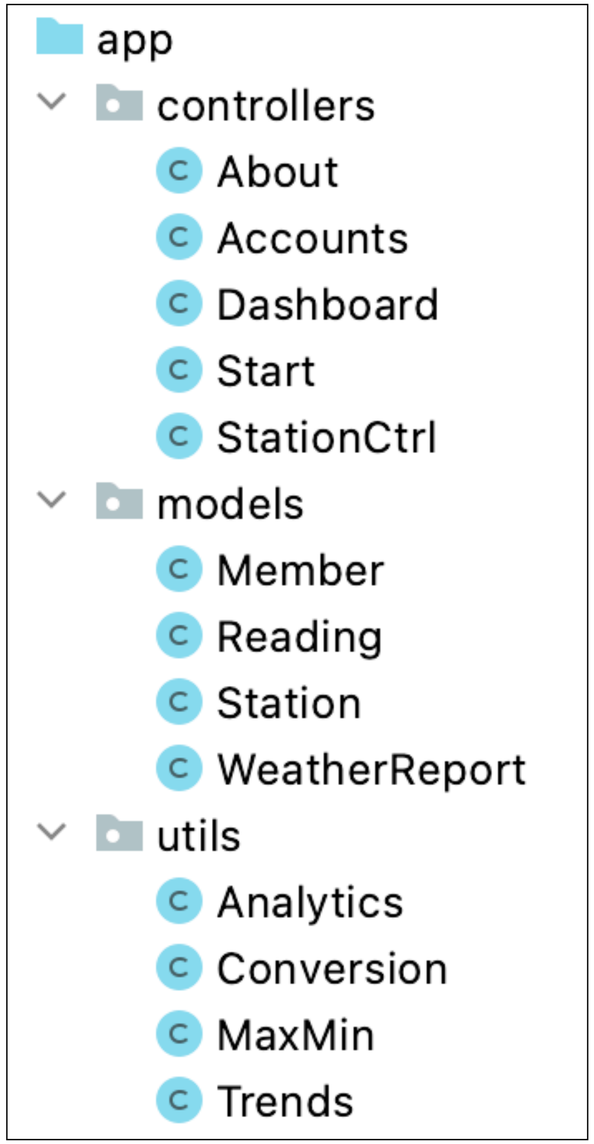

Please Signup to WeatherTop
localhost:9000/signup
WeatherTop
Please Signup to WeatherTop
Signup
Login
Register
First Name
Last Name
Email
bob@gmail.com
Password
Submit


WeatherTop Stations
localhost:9000/dashboard
WeatherTop
WeatherTop Stations
Dashboard
About
Dunmore
Lat: 52.149
Lng: -6.994
Weather
Temp
0.0 C
0.0 F
Max: 0.0
Min: 0.0
Wind
0 bft
Feels like
0.0
Max: 0.0
Min: 0.0
Pressure
0 hpa
Max: 0.0
Min: 0.0
Tramore
Lat: 52.16
Lng: -7.152
Weather
Temp
0.0 C
0.0 F
Max: 0.0
Min: 0.0
Wind
0 bft
Feels like
0.0
Max: 0.0
Min: 0.0
Pressure
0 hpa
Max: 0.0
Min: 0.0
Name
Latitude
Longitude
Name
00.00
00.00
Add Station

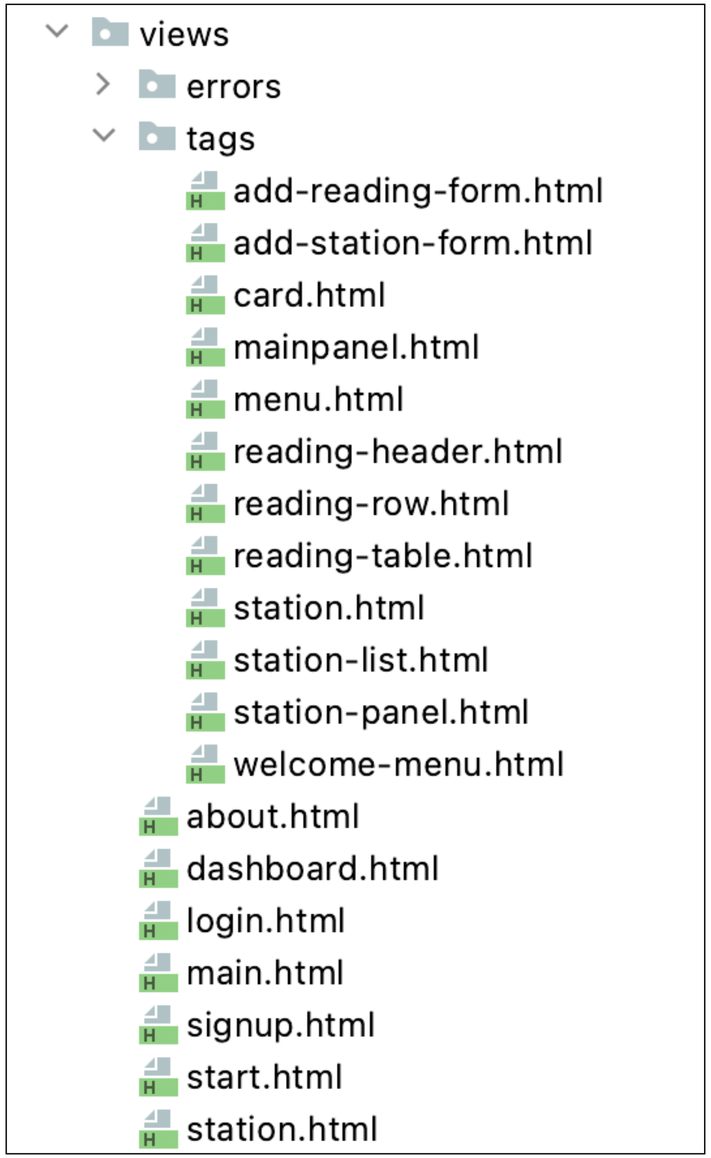

Dunmore
Lat: 52.149
Lng: -6.994
Weather
Partial
Clouds
Temp
0.0 C
32.0 F
Max: 8.0
Min: 0.0
Wind
1 bft
North
Feels like
-2.02
Max: 6.0
Min: 1.0
Pressure
120 hpa
Max:
1000.0
Min: 120.0
Date/Time
Code
Temp
Wind Speed
Wind Direction
Pressure
2021-01-20 10:31:00.0
700
8.0
1.0
90
1000
2021-01-19 09:31:00.0
200
0.5
3.5
120
999
2021-04-08 14:08:48.756
200
0.0
6.0
1000
120

Solution

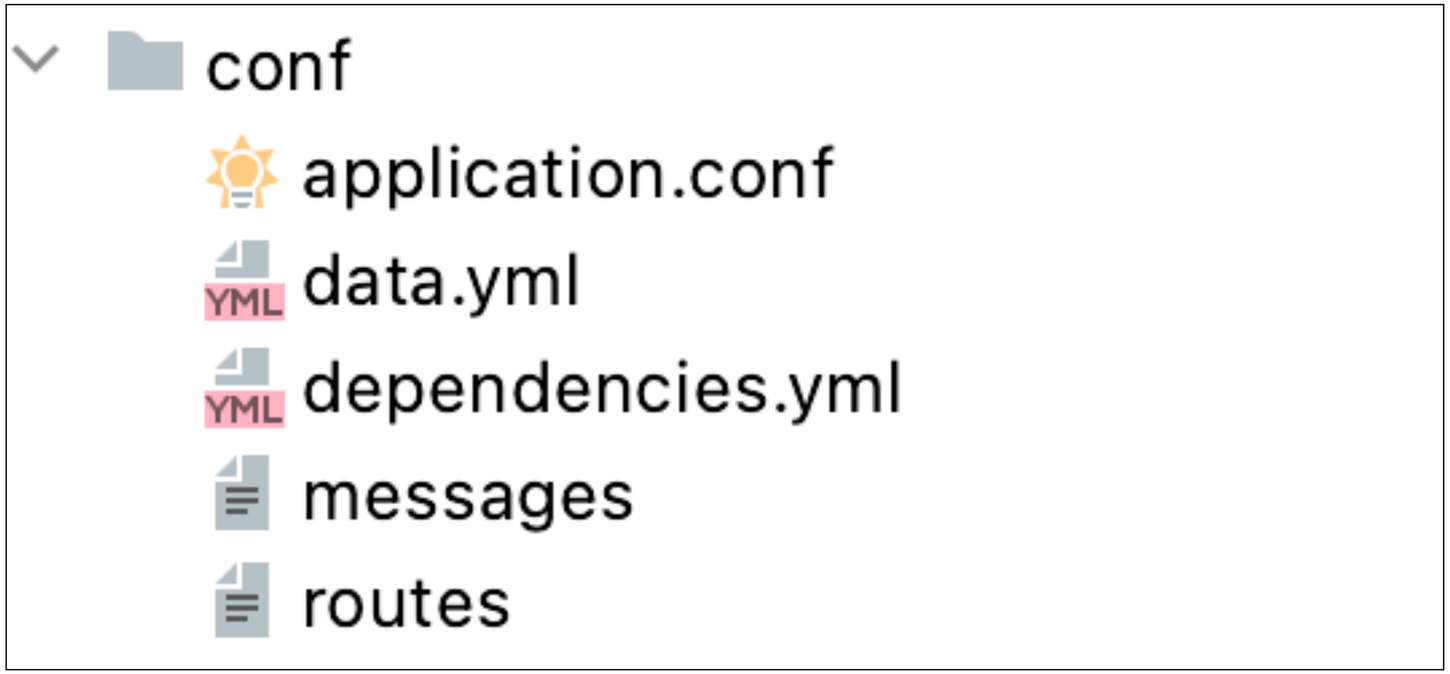
java



html



config


conf
application.conf
data.yml
dependencies.yml
messages
routes

config

Routes

```
# Routes
# This file defines all application routes (Higher priority routes first)
# ~~~~

# Accounts
GET     /                                       Accounts.index
GET     /signup                                 Accounts.signup
GET     /login                                  Accounts.login
GET     /logout                                 Accounts.logout
POST    /authenticate                           Accounts.authenticate
POST    /register                               Accounts.register

# Home page
GET      /dashboard                             Dashboard.index
GET      /about                                 About.index
POST     /dashboard/addstation                  Dashboard.addStation
GET      /dashboard/deletestation/{id}          Dashboard.deleteStation

GET      /stations/{id}                         StationCtrl.index
POST     /stations/{id}/addreading              StationCtrl.addReading
GET      /stations/{stationid}/deletereading/{id} StationCtrl.deleteReading

# Ignore favicon requests
GET      /favicon.ico                           404

# Map static resources from the /app/public folder to the /public path
GET      /public/                               staticDir:public

# Catch all
*        /{controller}/{action}                 {controller}.{action}
```

data.yaml

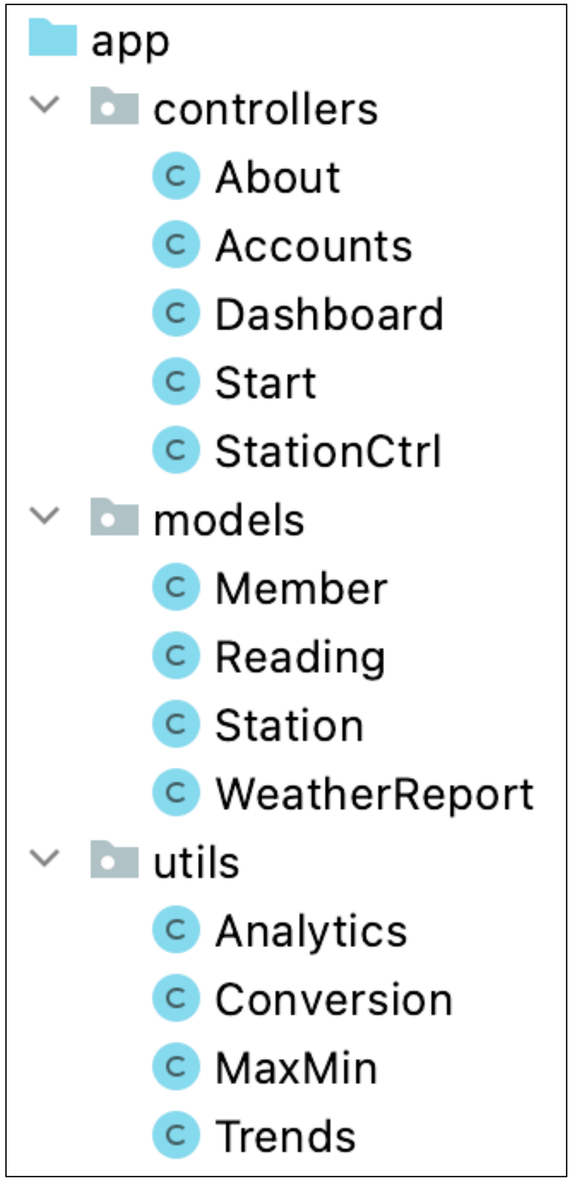
```yaml
Reading(r1):
  date: 2021-01-19 08:31:00
  code: 800
  temperature: 0.5
  windSpeed: 3.5
  windDirection: 220
  pressure: 1001

Reading(r2):
  date: 2021-01-20 09:34:00
  code: 600
  temperature: 6.0
  windSpeed: 2
  windDirection: 200
  pressure: 1004

Reading(r3):
  date: 2021-01-20 10:31:00
  code: 700
  temperature: 8.0
  windSpeed: 1
  windDirection: 90
  pressure: 1000

Reading(r4):
  date: 2021-01-19 09:31:00
  code: 200
  temperature: 0.5
  windSpeed: 3.5
  windDirection: 120
  pressure: 999

Station(s1):
  name: Tramore
  lat: 52.160
  lng: -7.152
  readings:
    - r1
    - r2

Station(s2):
  name: Dunmore
  lat: 52.149
  lng: -6.994
  readings:
    - r3
    - r4

Member(m1):
  email: homer@simpson.com
  password: secret
  firstname: Homer
  lastname : Simpson
  stations :
    - s1
    - s2
```

```java
@OnApplicationStart
public class Bootstrap extends Job
{
  public void doJob()
  {
    if (Member.count() == 0)
    {
      Fixtures.loadModels("data.yml");
    }
  }
}
```

config

```
# Database configuration
# ~~~~~
# Enable a database engine if needed.
#
# To quickly set up a development database, use either:
#   - mem : for a transient in memory database (H2 in memory)
#   - fs  : for a simple file written database (H2 file stored)

db.default=mem
```

```
# Database configuration
# ~~~~~
# Enable a database engine if needed.
#
# To quickly set up a development database, use either:
#   - mem : for a transient in memory database (H2 in memory)
#   - fs  : for a simple file written database (H2 file stored)

# db.default=mem

db=postgres://iwbagwwl:2DWjtsdfgsfgsdfgI1revvB3@tai.db.elephantsql.com:5432/iwbagwwl
jpa.dialect=org.hibernate.dialect.PostgreSQLDialect
jpa.ddl=update
```



WeatherReport

Reading

Station

MaxMin

Trends

Member

Bootstrap

Conversion

Accounts

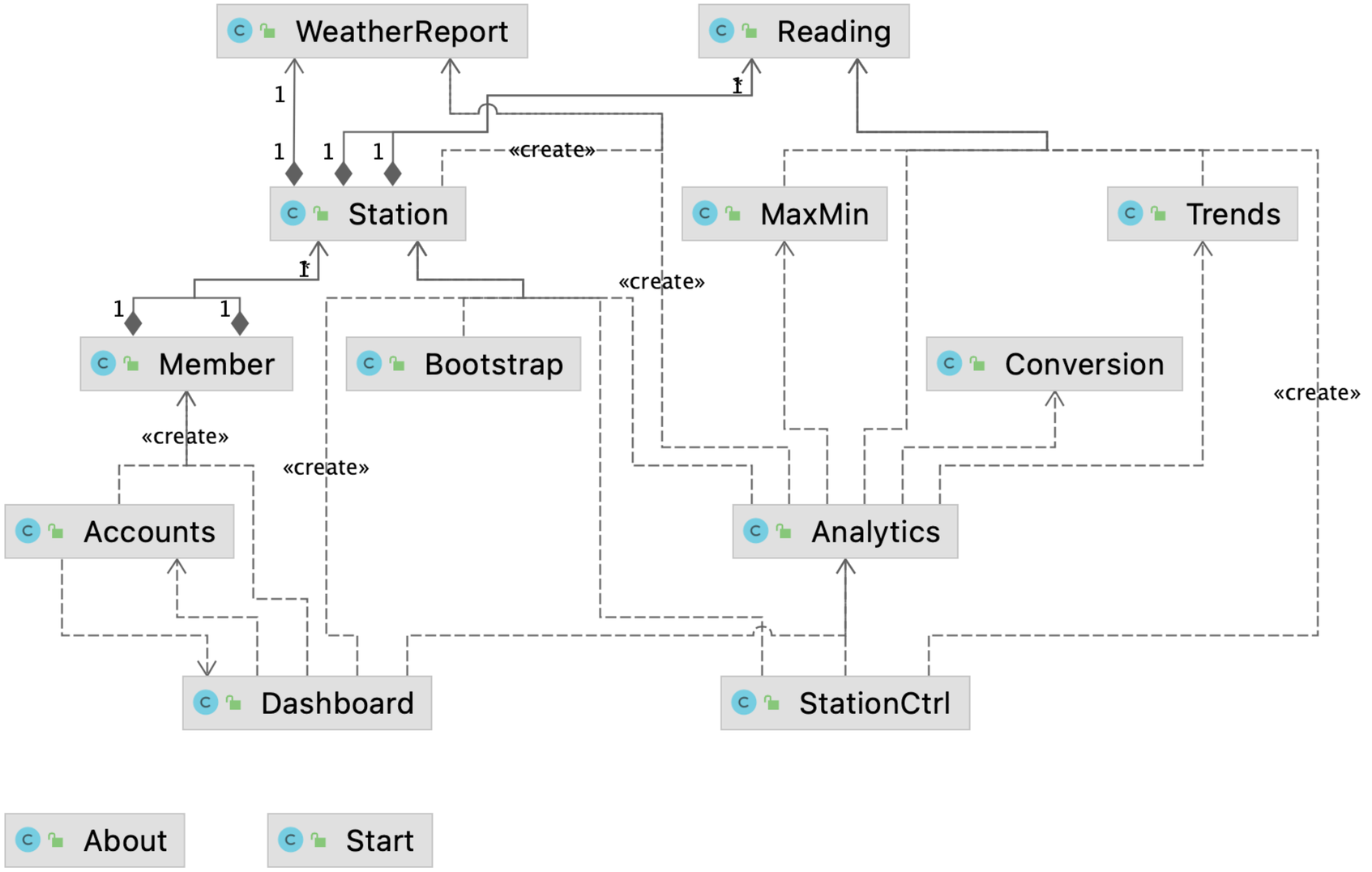
Analytics

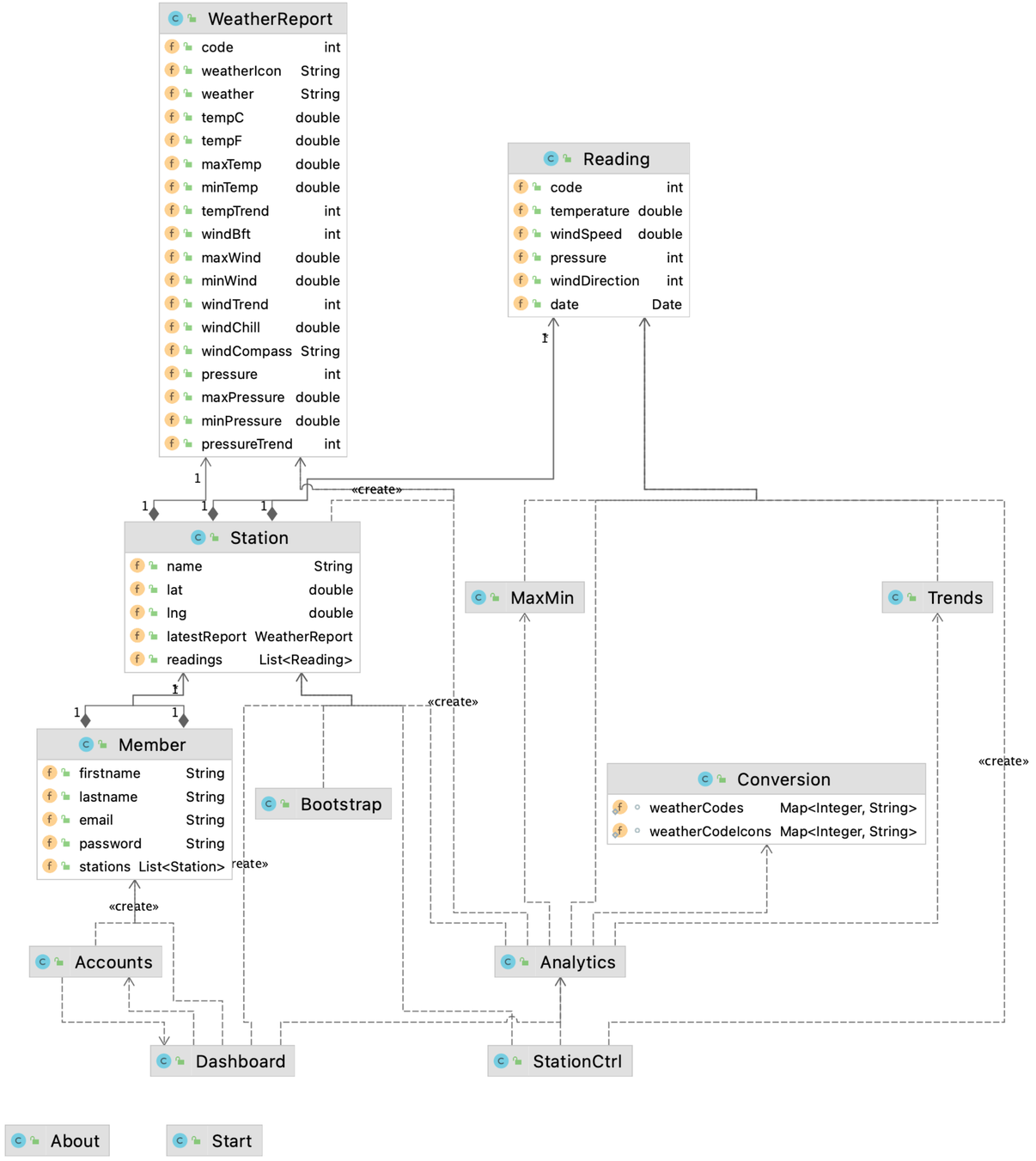
Dashboard

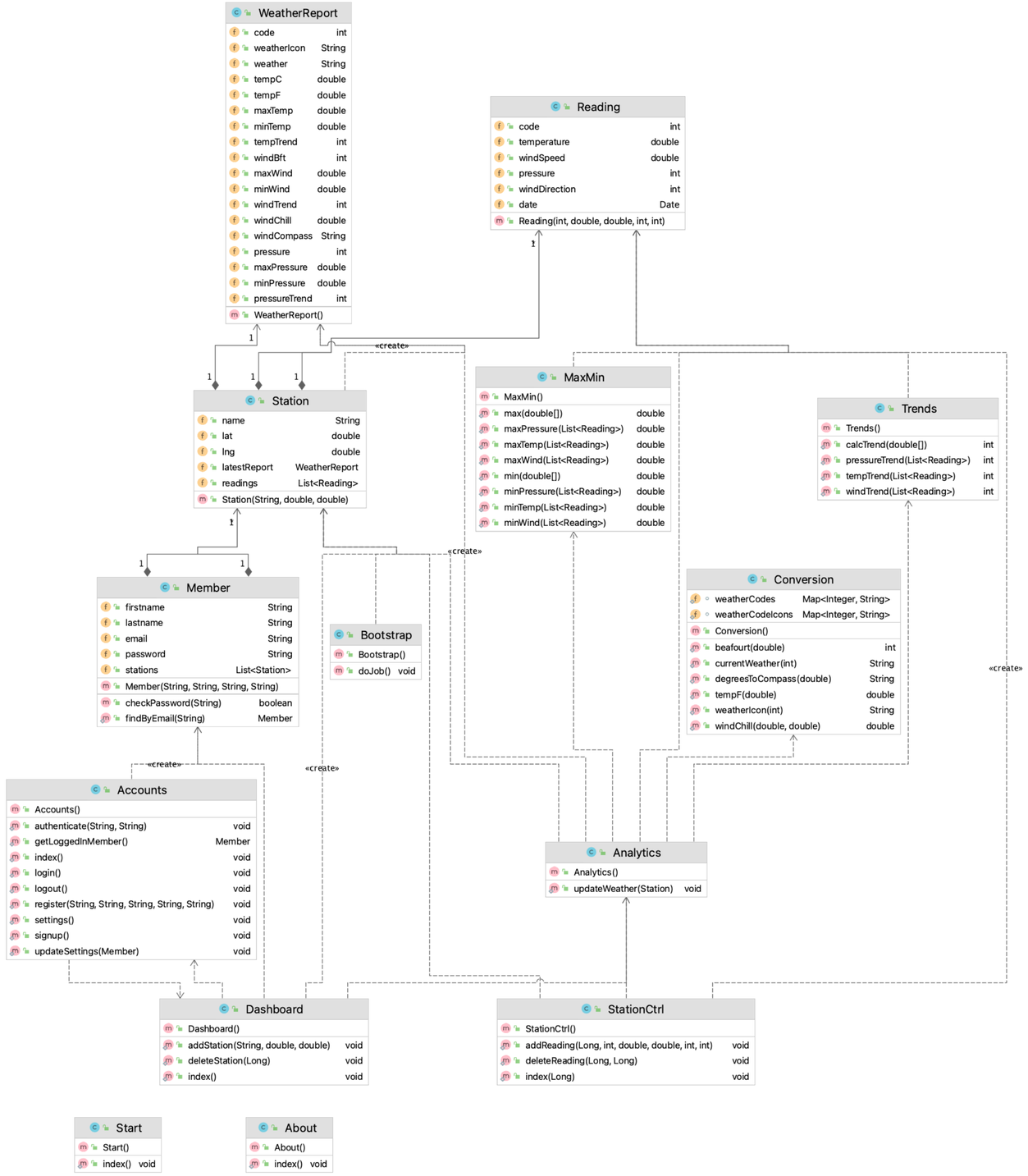
StationCtrl

About

Start

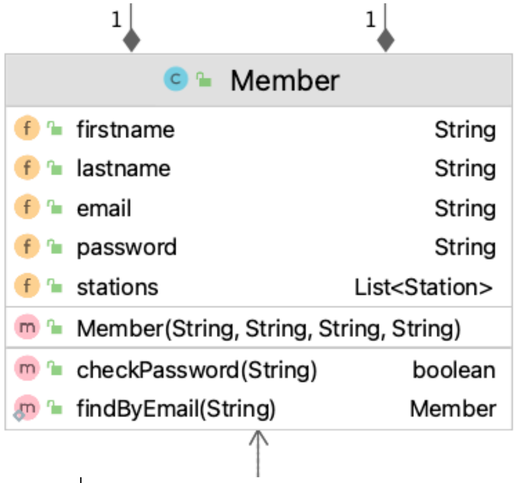

WeatherReport
Reading
1
1
1
1
1
«create»
Station
MaxMin
Trends
1
1
1
«create»
1
1
Member
Bootstrap
Conversion
«create»
«create»
«create»
Accounts
Analytics
Dashboard
StationCtrl
About
Start


WeatherReport
code int
weatherIcon String
weather String
tempC double
tempF double
maxTemp double
minTemp double
tempTrend int
windBft int
maxWind double
minWind double
windTrend int
windChill double
windCompass String
pressure int
maxPressure double
minPressure double
pressureTrend int
Reading
code int
temperature double
windSpeed double
pressure int
windDirection int
date Date
Station
name String
lat double
lng double
latestReport WeatherReport
readings List<Reading>
MaxMin
Trends
Member
firstname String
lastname String
email String
password String
stations List<Station>
Bootstrap
Conversion
weatherCodes Map<Integer, String>
weatherCodeIcons Map<Integer, String>
Accounts
Analytics
Dashboard
StationCtrl
About
Start
«create»
1


WeatherReport
code int
weatherIcon String
weather String
tempC double
tempF double
maxTemp double
minTemp double
tempTrend int
windBft int
maxWind double
minWind double
windTrend int
windChill double
windCompass String
pressure int
maxPressure double
minPressure double
pressureTrend int
WeatherReport()
Reading
code int
temperature double
windSpeed double
pressure int
windDirection int
date Date
Reading(int, double, double, int, int)
Station
name String
lat double
lng double
latestReport WeatherReport
readings List<Reading>
Station(String, double, double)
MaxMin
MaxMin()
max(double[]) double
maxPressure(List<Reading>) double
maxTemp(List<Reading>) double
maxWind(List<Reading>) double
min(double[]) double
minPressure(List<Reading>) double
minTemp(List<Reading>) double
minWind(List<Reading>) double
Trends
Trends()
calcTrend(double[]) int
pressureTrend(List<Reading>) int
tempTrend(List<Reading>) int
windTrend(List<Reading>) int
Member
firstname String
lastname String
email String
password String
stations List<Station>
Member(String, String, String, String)
checkPassword(String) boolean
findByEmail(String) Member
Bootstrap
Bootstrap()
doJob() void
Conversion
weatherCodes Map<Integer, String>
weatherCodeIcons Map<Integer, String>
Conversion()
beafort(double) int
currentWeather(int) String
degreesToCompass(double) String
tempF(double) double
weatherIcon(int) String
windChill(double, double) double
Accounts
Accounts()
authenticate(String, String) void
getLoggedInMember() Member
index() void
login() void
logout() void
register(String, String, String, String, String) void
settings() void
signup() void
updateSettings(Member) void
Analytics
Analytics()
updateWeather(Station) void
Dashboard
Dashboard()
addStation(String, double, double) void
deleteStation(Long) void
index() void
StationCtrl
StationCtrl()
addReading(Long, int, double, double, int, int) void
deleteReading(Long, Long) void
index(Long) void
Start
Start()
index() void
About
About()
index() void
«create»
1

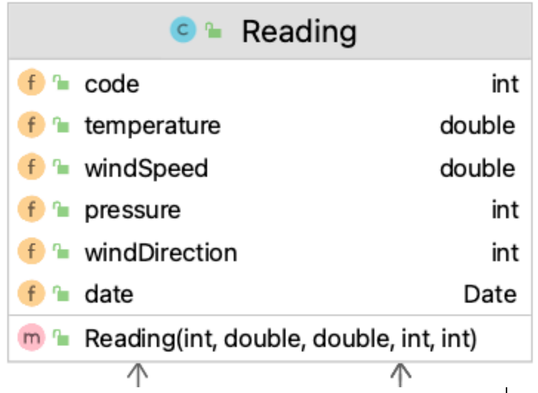
Member

```
@Entity
public class Member extends Model {
  public String firstname;
  public String lastname;
  public String email;
  public String password;

  @OneToMany(cascade = CascadeType.ALL)
  public List<Station> stations = new ArrayList<Station>();

  public Member(String firstname, String lastname, String email, String password) {
    this.firstname = firstname;
    this.lastname = lastname;
    this.email = email;
    this.password = password;
  }

  public static Member findByEmail(String email) {
    return find("email", email).first();
  }

  public boolean checkPassword(String password) {
    return this.password.equals(password);
  }
}
```




```
Member(m1):
  email: homer@simpson.com
  password: secret
  firstname: Homer
  lastname : Simpson
  stations :
    - s1
    - s2
```

Reading

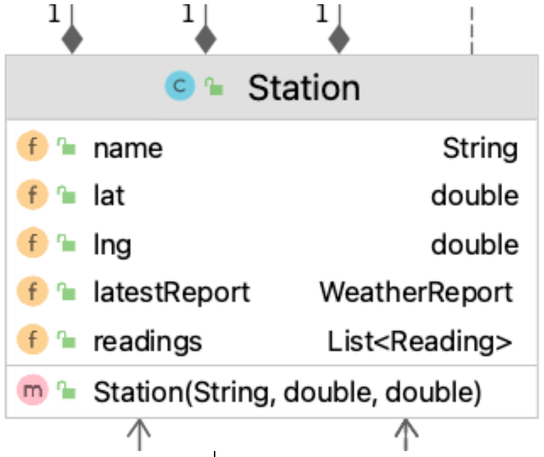



```java
@Entity
public class Reading extends Model {
  public int code;
  public double temperature;
  public double windSpeed;
  public int pressure;
  public int windDirection;
  public Date date;

  public Reading(int code, double temperature, double windSpeed, int pressure, int windDirection) {
    this.code = code;
    this.temperature = temperature;
    this.windSpeed = windSpeed;
    this.pressure = pressure;
    this.windDirection = windDirection;
    this.date = new Date();
  }
}
```

```
Reading(r1):
  date: 2021-01-19 08:31:00
  code: 800
  temperature: 0.5
  windSpeed: 3.5
  windDirection: 220
  pressure: 1001
```

Station

```
@Entity
public class Station extends Model {
  public String name;
  public double lat;
  public double lng;

  @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER, orphanRemoval = true)
  public WeatherReport latestReport = new WeatherReport();

  @OneToMany(cascade = CascadeType.ALL)
  public List<Reading> readings = new ArrayList<Reading>();

  public Station(String name, double lat, double lng) {
    this.name = name;
    this.lat = lat;
    this.lng = lng;
  }
}
```



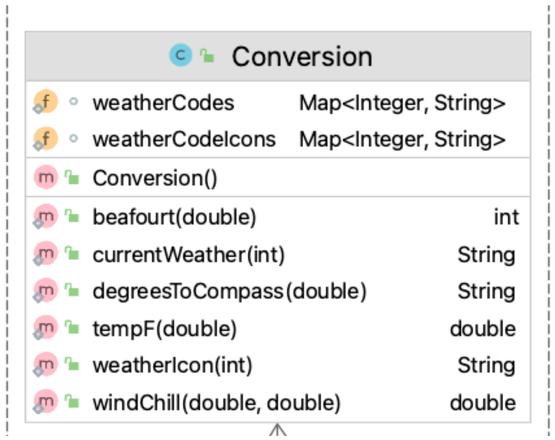

```
Station(s1):
  name: Tramore
  lat: 52.160
  lng: -7.152
  readings:
    - r1
    - r2
```

WeatherReport

```java
@Entity
public class WeatherReport extends Model {

  public int code;
  public String weatherIcon;
  public String weather;
  public double tempC, tempF;
  public double maxTemp, minTemp;
  public int tempTrend;

  public int windBft;
  public double maxWind, minWind;
  public int windTrend;
  public double windChill;
  public String windCompass;

  public int pressure;
  public double maxPressure, minPressure;
  public int pressureTrend;
}
```

Conversion

```java
public class Conversion {
  static Map<Integer, String> weatherCodes = new HashMap<>();
  static Map<Integer, String> weatherCodeIcons = new HashMap<>();

  static {
    weatherCodes.put(100, "Clear");
    weatherCodes.put(200, "Partial Clouds");
    weatherCodes.put(300, "Cloudy");
    weatherCodes.put(400, "Light Showers");
    weatherCodes.put(500, "Heavy Showers");
    weatherCodes.put(600, "Rain");
    weatherCodes.put(700, "Snow");
    weatherCodes.put(800, "Thunder");

    weatherCodeIcons.put(100, "sun");
    weatherCodeIcons.put(200, "cloud sun");
    weatherCodeIcons.put(300, "cloud");
    weatherCodeIcons.put(400, "cloud sun rain");
    weatherCodeIcons.put(500, "cloud showers heavy");
    weatherCodeIcons.put(600, "cloud rain");
    weatherCodeIcons.put(700, "snowflake");
    weatherCodeIcons.put(800, "bolt");
  }
```

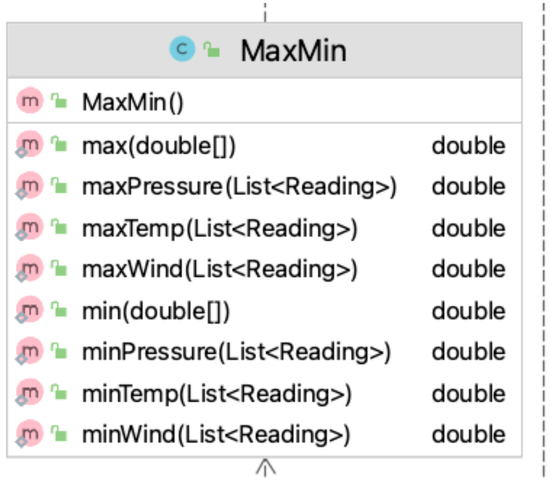



```java
  public static String weatherIcon(int code) {
    return weatherCodeIcons.get(code);
  }

  public static String currentWeather(int code) {
    return weatherCodes.get(code);
  }

  public static double tempF(double tempC) {
    return (tempC * 1.8) + 32;
  }

  public static int beafourt(double windspeed) {
    if (windspeed == 0) {
      return 0;
    } else if (windspeed >= 1 && windspeed <= 6) {
      return 1;
    } else if (windspeed >= 7 && windspeed <= 11) {
      return 2;
    } else if (windspeed >= 12 && windspeed <= 19) {
      return 3;
    } else if (windspeed >= 20 && windspeed <= 29) {
      return 4;
    } else if (windspeed >= 30 && windspeed <= 39) {
      return 5;
    } else if (windspeed >= 40 && windspeed <= 50) {
      return 6;
    } else if (windspeed >= 51 && windspeed <= 62) {
      return 7;
    } else if (windspeed >= 63 && windspeed <= 75) {
      return 8;
    } else if (windspeed >= 76 && windspeed <= 87) {
      return 9;
    } else if (windspeed >= 88 && windspeed <= 102) {
      return 10;
    } else if (windspeed >= 103 && windspeed <= 117) {
      return 11;
    } else if (windspeed >= 117) {
      return 12;
    }
    return -1;
  }
```

```java
  public static String degreesToCompass(double deg) {
    if (deg > 11.25 && deg <= 33.75) {
      return "North North East";
    } else if (deg > 33.75 && deg <= 56.25) {
      return "East North East";
    } else if (deg > 56.25 && deg <= 78.75) {
      return "East";
    } else if (deg > 78.75 && deg <= 101.25) {
      return "East South East";
    } else if (deg > 101.25 && deg <= 123.75) {
      return "East South East";
    } else if (deg > 123.75 && deg <= 146.25) {
      return "South East";
    } else if (deg > 146.25 && deg <= 168.75) {
      return "South South East";
    } else if (deg > 168.75 && deg <= 191.25) {
      return "South";
    } else if (deg > 191.25 && deg <= 213.75) {
      return "South South West";
    } else if (deg > 213.75 && deg <= 236.25) {
      return "South West";
    } else if (deg > 236.25 && deg <= 258.75) {
      return "West South West";
    } else if (deg > 258.75 && deg <= 281.25) {
      return "West";
    } else if (deg > 281.25 && deg <= 303.75) {
      return "West North West";
    } else if (deg > 303.75 && deg <= 326.25) {
      return "North West";
    } else if (deg > 326.25 && deg <= 348.75) {
      return "North North West";
    } else {
      return "North";
    }
  }
}
```

MaxMin

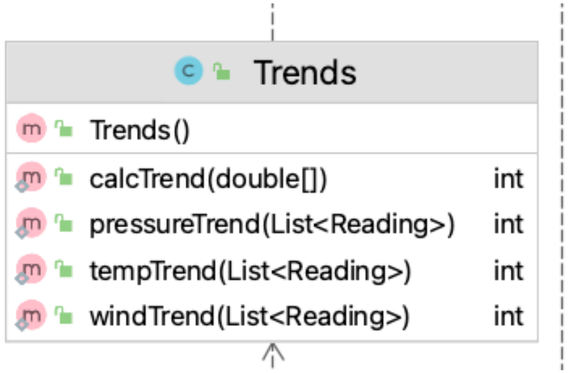
```java
public class MaxMin {
  public static double max(double values[]) {
    double max = values[0];
    for (double value : values) {
      if (value > max) {
        max = value;
      }
    }
    return max;
  }

  public static double min(double values[]) {
    double min = values[0];
    for (double value : values) {
      if (value < min) {
        min = value;
      }
    }
    return min;
  }
```




```java
  public static double maxTemp(List<Reading> readings) {
    double values[] = new double[readings.size()];
    for (int i = 0; i < readings.size(); i++) values[i] = readings.get(i).temperature;
    return max(values);
  }

  public static double minTemp(List<Reading> readings) {
    double values[] = new double[readings.size()];
    for (int i = 0; i < readings.size(); i++) values[i] = readings.get(i).temperature;
    return min(values);
  }

  public static double maxWind(List<Reading> readings) {
    double values[] = new double[readings.size()];
    for (int i = 0; i < readings.size(); i++) values[i] = readings.get(i).windSpeed;
    return max(values);
  }

  public static double minWind(List<Reading> readings) {
    double values[] = new double[readings.size()];
    for (int i = 0; i < readings.size(); i++) values[i] = readings.get(i).windSpeed;
    return min(values);
  }

  public static double maxPressure(List<Reading> readings) {
    double values[] = new double[readings.size()];
    for (int i = 0; i < readings.size(); i++) values[i] = readings.get(i).pressure;
    return max(values);
  }

  public static double minPressure(List<Reading> readings) {
    double values[] = new double[readings.size()];
    for (int i = 0; i < readings.size(); i++) values[i] = readings.get(i).pressure;
    return min(values);
  }
}
```

Trends

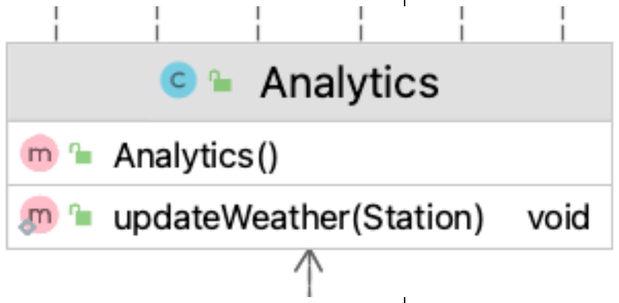
```java
public class Trends {
  public static int calcTrend(double values[]) {
    int trend = 0;
    if (values.length > 2) {
      if ((values[2] > values[1]) && (values[1] > values[0])) {
        trend = 1;
      } else if ((values[2] < values[1]) && (values[1] < values[0])) {
        trend = -1;
      }
    }
    return trend;
  }
```



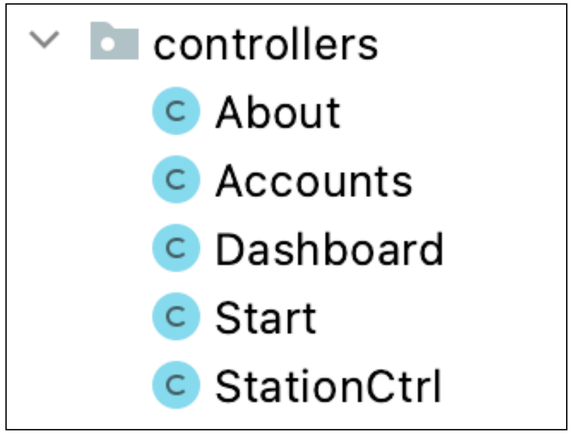

```java
public class Trends {
  public static int tempTrend(List<Reading> readings) {
    int trend = 0;
    if (readings.size() > 2) {
      double values[] = {readings.get(readings.size() - 3).temperature, readings.get(readings.size() - 2).temperature, readings.get(readings.size() - 1).temperature};
      trend = calcTrend(values);
    }
    return trend;
  }

  public static int windTrend(List<Reading> readings) {
    int trend = 0;
    if (readings.size() > 2) {
      double values[] = {readings.get(readings.size() - 3).windSpeed, readings.get(readings.size() - 2).windSpeed, readings.get(readings.size() - 1).windSpeed};
      trend = calcTrend(values);
    }
    return trend;
  }

  public static int pressureTrend(List<Reading> readings) {
    int trend = 0;
    if (readings.size() > 2) {
      double values[] = {readings.get(readings.size() - 3).pressure, readings.get(readings.size() - 2).pressure, readings.get(readings.size() - 1).pressure};
      trend = calcTrend(values);
    }
    return trend;
  }
}
```

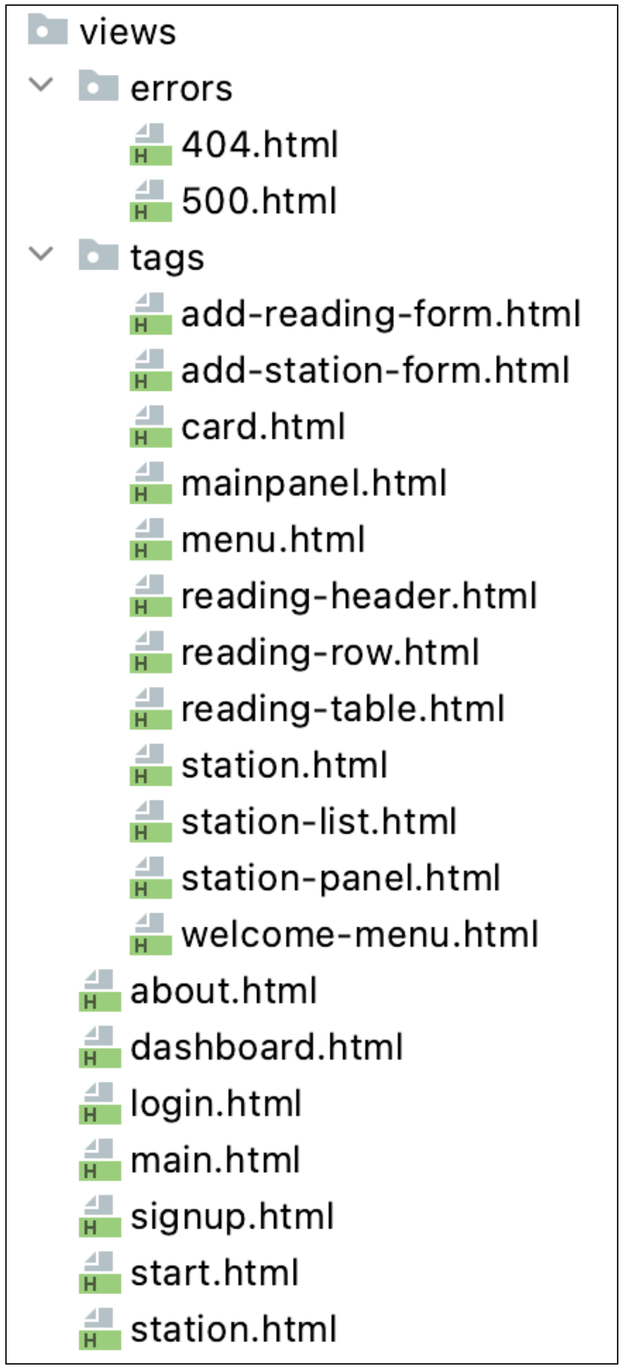
Analytics

```
public class Analytics {
  public static void updateWeather(Station station) {
    if (station.readings.size() > 0) {
      Reading lastReading = station.readings.get(station.readings.size() - 1);

      station.latestReport = new WeatherReport();
      station.latestReport.code = lastReading.code;
      station.latestReport.weatherIcon = Conversion.weatherIcon(lastReading.code);
      station.latestReport.weather = Conversion.currentWeather(lastReading.code);

      station.latestReport.tempC = lastReading.temperature;
      station.latestReport.tempF = Conversion.tempF(lastReading.temperature);
      station.latestReport.maxTemp = MaxMin.maxTemp(station.readings);
      station.latestReport.minTemp = MaxMin.minTemp(station.readings);
      station.latestReport.tempTrend = Trends.tempTrend(station.readings);

      station.latestReport.windBft = Conversion.beafourt(lastReading.windSpeed);
      station.latestReport.maxWind = MaxMin.maxWind(station.readings);
      station.latestReport.minWind = MaxMin.minWind(station.readings);
      station.latestReport.windTrend = Trends.windTrend(station.readings);
      String str = String.format("%1.2f", Conversion.windChill(lastReading.temperature, lastReading.windSpeed));
      station.latestReport.windChill = Double.valueOf(str);
      station.latestReport.windCompass = Conversion.degreesToCompass(lastReading.windDirection);

      station.latestReport.pressure = lastReading.pressure;
      station.latestReport.maxPressure = MaxMin.maxPressure(station.readings);
      station.latestReport.minPressure = MaxMin.minPressure(station.readings);
      station.latestReport.pressureTrend = Trends.pressureTrend(station.readings);

      station.save();
    }
  }
```



Controllers + Views





main

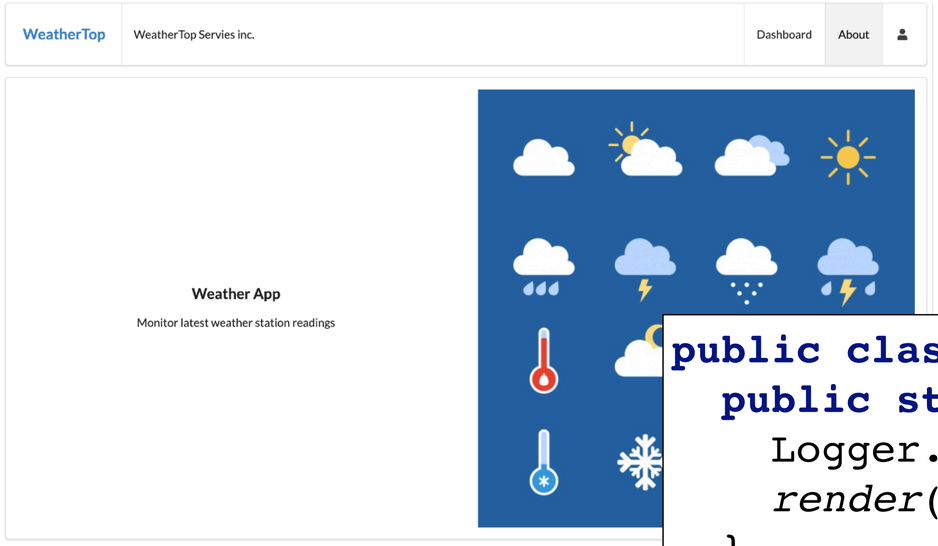
```
<!DOCTYPE html>
<html>
  <head>
    <title>#{get 'title' /}</title>
    <script src="https://cdn.jsdelivr.net/npm/jquery@3.3.1/dist/jquery.min.js"></script>
    <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivr.net/npm/fomantic-ui@2.8.7/dist/semantic.min.css">
    <script src="https://cdn.jsdelivr.net/npm/fomantic-ui@2.8.7/dist/semantic.min.js"></script>
  </head>
  <body>
    <section class="ui container">
      #{doLayout /}
    </section>
    <script>
      $('.ui.dropdown').dropdown();
    </script>
  </body>
</html>
```

welcomemenu


```
<nav class="ui menu">
  <header class="ui header item"> <a href="/"> WeatherTop </a></header>
  <div class="ui center aligned basic segment">
    <p> #{get 'title' /}</p>
  </div>
  <div class="right menu">
    <a id="signup" class="item" href="/signup"> Signup  </a>
    <a id="login" class="item" href="/login">  Login   </a>
  </div>
</nav>

<script>
  $("#${_id}").addClass("active item");
</script>
```

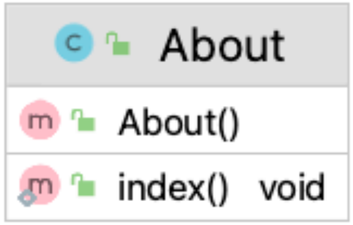
menu


```
<nav class="ui menu">
  <header class="ui header item"> <a href="/"> WeatherTop </a></header>
  <div class="ui center aligned basic segment">
    <p> #{get 'title' /}</p>
  </div>
  <div class="right menu">
    <a id="dashboard" class="item" href="/dashboard"> Dashboard  </a>
    <a id="about" class="item" href="/about"> About </a>
    <div class="ui dropdown item">
      <i class="user icon"></i>
      <div class="menu">
        <a class="item" href="/logout">Logout</a>
      </div>
    </div>
  </div>
</nav>

<script>
  $("#${_id}").addClass("active item");
</script>
```



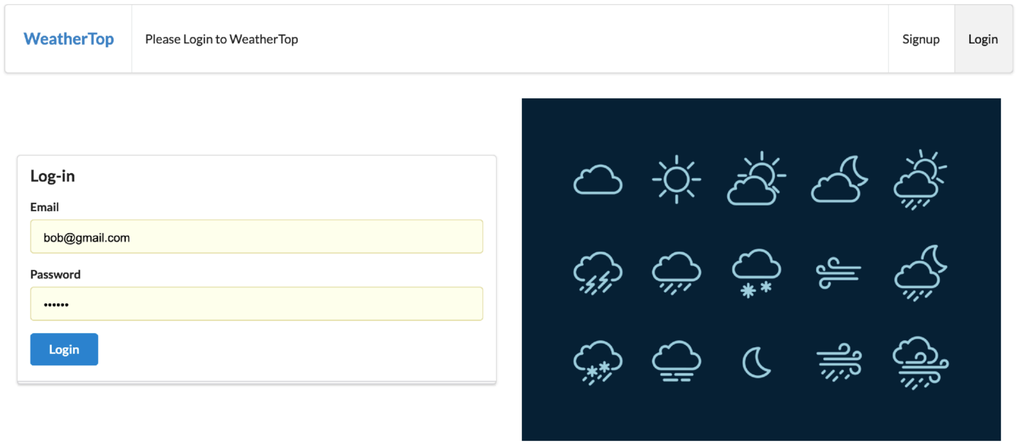

```
public class About extends Controller {
  public static void index() {
    Logger.info("Rendering about");
    render("about.html");
  }
}
```

about

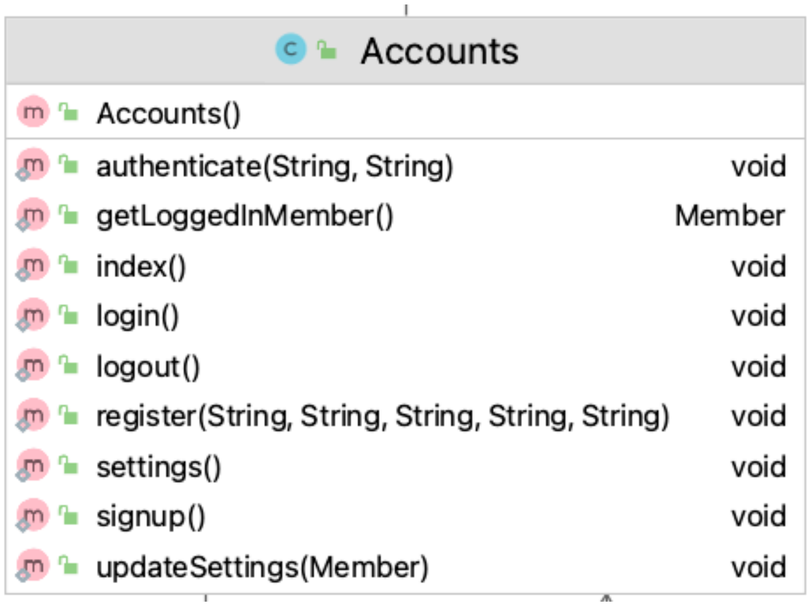



```
#{extends 'main.html' /}
#{set title:'WeatherTop Servies inc.' /}
#{menu id:"about"/}

<section class="ui center aligned middle aligned two column grid segment">
  <div class="column">
    <header class="ui header"> Weather App </header>
    <p> Monitor latest weather station readings </p>
  </div>
  <div class="column">
    <img class="ui image" src="/public/images/icons.jpg">
  </div>
</section>n>
```

Login

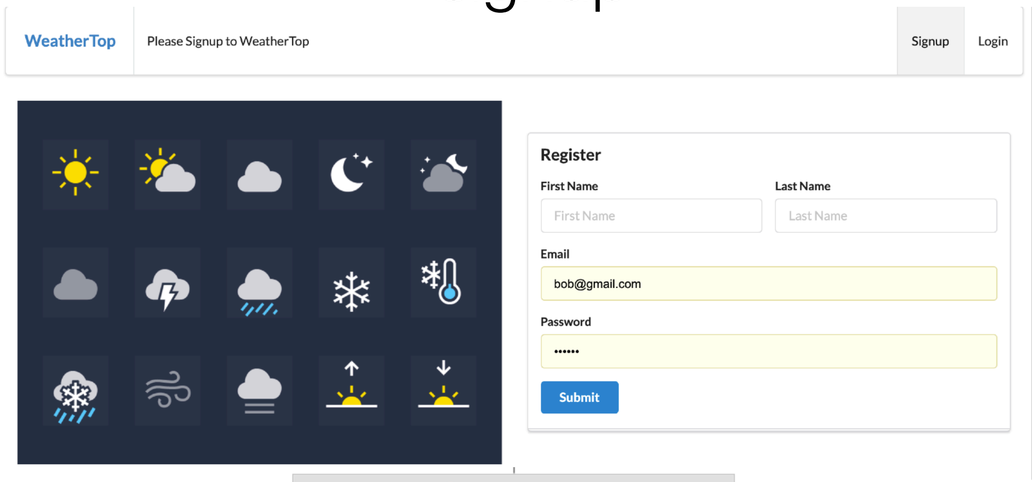


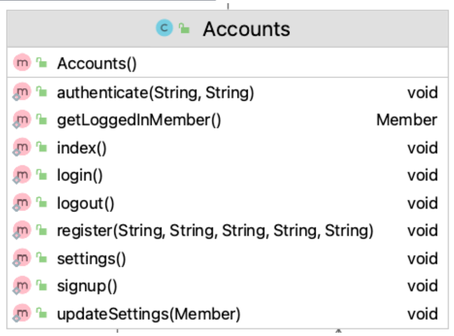

```
#{extends 'main.html' /}
#{set title:'Please Login to WeatherTop' /}
#{welcome-menu id:"login"/}

<div class="ui two column middle aligned grid basic segment">
  <div class="column">
    <form class="ui stacked segment form" action="/authenticate" method="POST">
      <h3 class="ui header">Log-in</h3>
      <div class="field">
        <label>Email</label> <input placeholder="Email" name="email">
      </div>
      <div class="field">
        <label>Password</label> <input type="password" name="password">
      </div>
      <button class="ui blue submit button">Login</button>
    </form>
  </div>
  <div class="column">
    <img class="ui image" src="/public/images/icons-2.jpg">
  </div>
</div>
```




```
public class Accounts extends Controller {
  …
  public static void login() {
    render("login.html");
  }

  public static void authenticate(String email, String password) {
    Logger.info("Attempting to authenticate with " + email + ":" + password);

    Member member = Member.findByEmail(email);
    if ((member != null) && (member.checkPassword(password) == true)) {
      Logger.info("Authentication successful");
      session.put("logged_in_Memberid", member.id);
      Dashboard.index();
    } else {
      Logger.info("Authentication failed");
      login();
    }
  }
}
```

signup



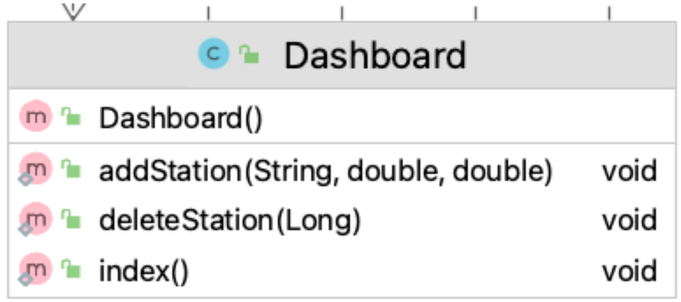


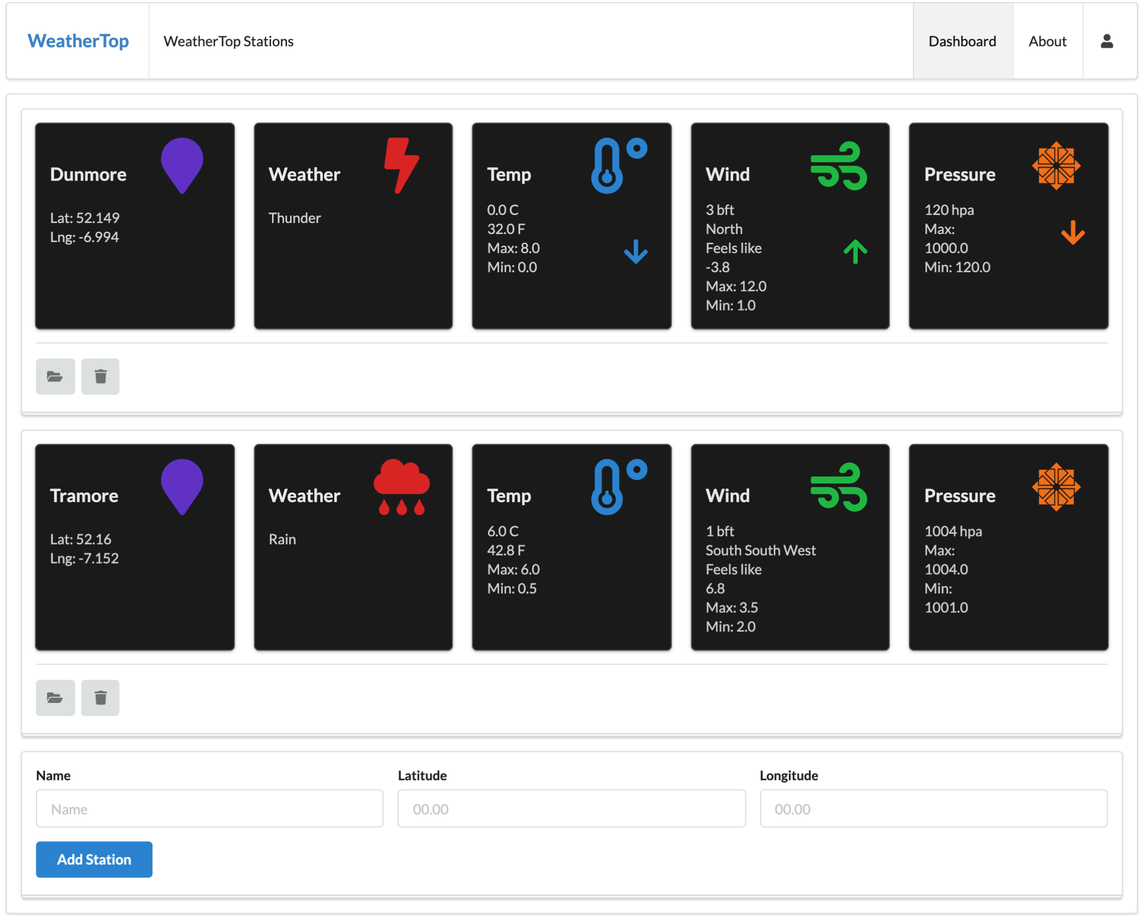

```
public class Accounts extends Controller {

  public static void signup() {
    render("signup.html");
  }

  public static void register(String email, String firstname,
                              String lastname, String password,
                              String stationname) {
    Member member = new Member(firstname, lastname, email, password);
    member.save();
    index();
  }
}
```

```
#{extends 'main.html' /}
#{set title:'Please Signup to WeatherTop' /}
#{welcome-menu id:"signup"/}

<div class="ui two column grid basic middle aligned segment">
  <div class="column">
    <img class="ui image" src="/public/images/icons-3.png">
  </div>
  <div class="column">
    <form class="ui stacked segment form" action="/register" method="POST">
      <h3 class="ui header">Register</h3>
      <div class="two fields">
        <div class="field">
          <label>First Name</label>
          <input placeholder="First Name" type="text" name="firstname">
        </div>
        <div class="field">
          <label>Last Name </label>
          <input placeholder="Last Name" type="text" name="lastname">
        </div>
      </div>
      <div class="field">
        <label>Email</label>
        <input placeholder="Email" type="text" name="email">
      </div>
      <div class="field">
        <label>Password</label>
        <input type="password" name="password">
      </div>
      <button class="ui blue submit button">Submit</button>
    </form>
  </div>
</div>
```

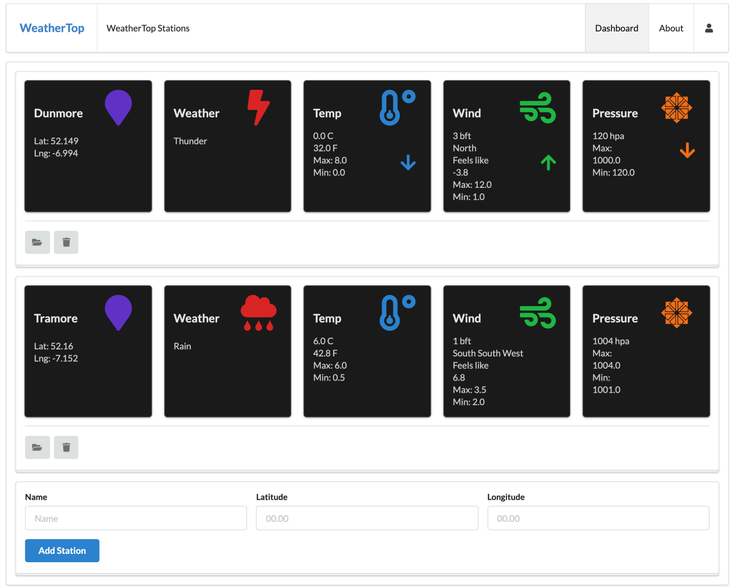
dashboard

```
public class Dashboard extends Controller {
  public static void index() {
    Logger.info("Rendering Dashboard");
    Member member = Accounts.getLoggedInMember();
    List<Station> stations = member.stations;
    for (Station station : member.stations) {
      Analytics.updateWeather(station);
    }
    Collections.sort(stations, (a, b) -> a.name.compareToIgnoreCase(b.name));
    render("dashboard.html", stations);
  }

  public static void addStation(String name, double lat, double lng) {
    Logger.info("Adding a Station");
    Member member = Accounts.getLoggedInMember();
    Station station = new Station(name, lat, lng);
    member.stations.add(station);
    member.save();
    redirect("/dashboard");
  }

  public static void deleteStation(Long id) {
    Logger.info("Deleting a Station");
    Member member = Accounts.getLoggedInMember();
    Station station = Station.findById(id);
    member.stations.remove(station);
    member.save();
    station.delete();
    redirect("/dashboard");
  }
}
```






```
#{extends 'main.html' /}
#{set title:'Dashboard' /}
#{set title:'WeatherTop Stations' /}
#{menu id:"dashboard"/}

<section class="ui segment">
  #{station-list stations:stations/}
  #{add-station-form /}
</section>
```



station-list

```
#{list items:_stations, as:'station'}
  <section class="ui stacked segment">
    #{station-panel station:station /}
    <div class="ui divider"></div>
    <a href="/stations/${station.id}" class="ui icon button">
      <i class="icon folder open"></i>
    </a>
    <a href="/dashboard/deletestation/${station.id}" class="ui icon button">
      <i class="icon trash"></i>
    </a>
  </section>
#{/list}
```

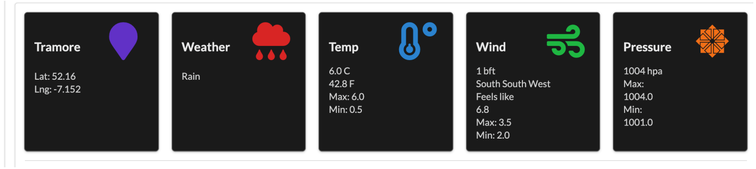
dashboard

```
#{extends 'main.html' /}
#{set title:'Dashboard' /}
#{set title:'WeatherTop Stations' /}
#{menu id:"dashboard"/}

<section class="ui segment">
  #{station-list stations:stations/}
  #{add-station-form /}
</section>
```

add-station-form

```
<form class="ui stacked segment form" action="/dashboard/addstation" method="POST">
  <div class="three fields">
    <div class="field">
      <label>Name</label>
      <input placeholder="Name" type="text" name="name">
    </div>
    <div class="field">
      <label>Latitude</label>
      <input placeholder="00.00" type="text" name="lat">
    </div>
    <div class="field">
      <label>Longitude</label>
      <input placeholder="00.00" type="text" name="lng">
    </div>
  </div>
  <button class="ui blue submit button">Add Station</button>
</form>
```

station-list




```
#{list items:_stations, as:'station'}
  <section class="ui stacked segment">
    #{station-panel station:station /}
    <div class="ui divider"></div>
    <a href="/stations/${station.id}" class="ui icon button">
      <i class="icon folder open"></i>
    </a>
    <a href="/dashboard/deletestation/${station.id}" class="ui icon button">
      <i class="icon trash"></i>
    </a>
  </section>
#{/list}
```

station-panel

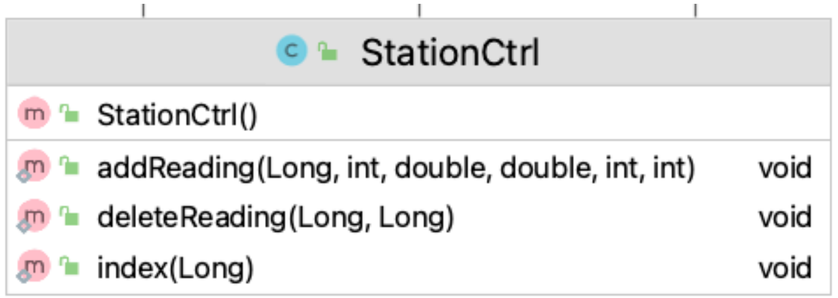
```
<div class="ui five doubling inverted cards">

  #{card title:_station.name,  icon:"marker", colour:"violet",
                               valueC:"Lat: " + _station.lat, valueD:"Lng: " + _station.lng /}

  #{card title:"Weather",      icon:_station.latestReport.weatherIcon, colour:"red",
                               valueC:_station.latestReport.weather /}

  #{card title:"Temp",         icon:"temperature low", colour:"blue",
                               valueA:_station.latestReport.tempC + " C",    valueB:_station.latestReport.tempF + " F",
                               valueC:"Max: " + _station.latestReport.maxTemp, valueD:"Min: " + _station.latestReport.minTemp,
                               trend:_station.latestReport.tempTrend /}

  #{card title:"Wind",         icon:"wind", colour:"green",
                               valueA:_station.latestReport.windBft+" bft", valueB:_station.latestReport.windCompass,
                               valueC:"Feels like " + _station.latestReport.windChill,
                               valueD:"Max: " + _station.latestReport.maxWind, valueE:"Min: " + _station.latestReport.minWind,
                               trend:_station.latestReport.windTrend /}

  #{card title:"Pressure",     icon:"centos", colour:"orange",
                               valueA:_station.latestReport.pressure + " hpa",
                               valueC:"Max: " + _station.latestReport.maxPressure, valueD:"Min: " + _station.latestReport.minPressure,
                               trend:_station.latestReport.pressureTrend /}
</div>
```

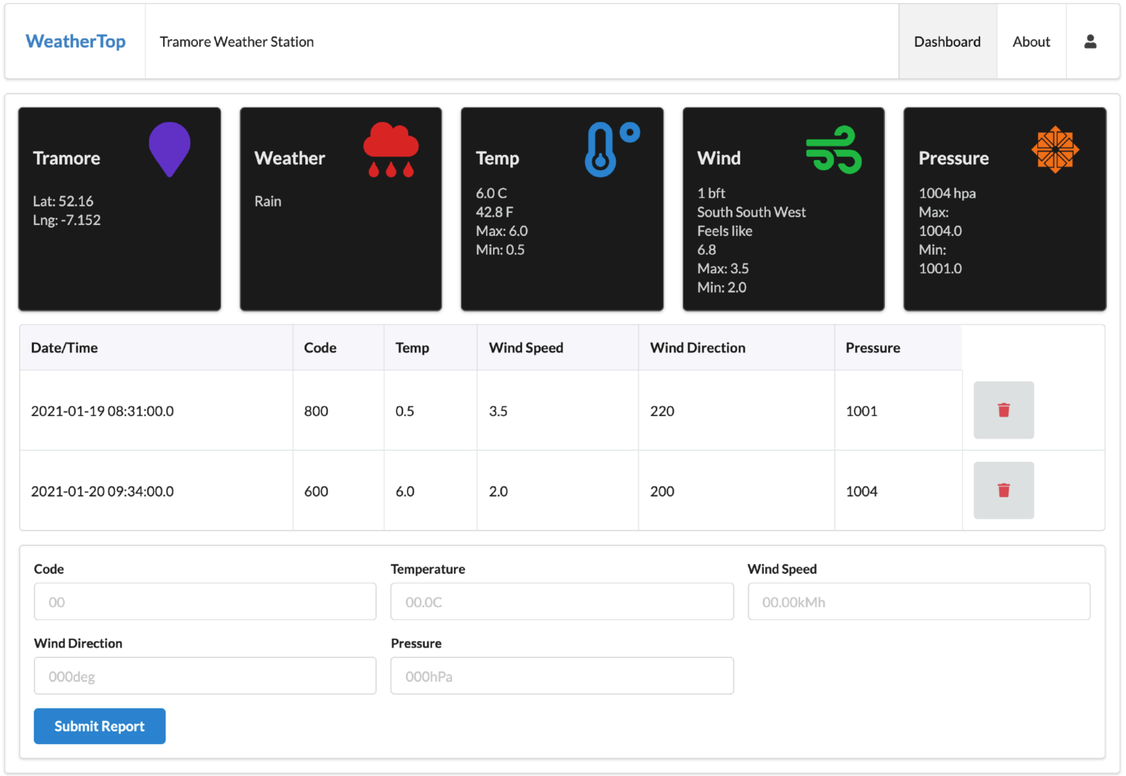
card

```
<div class="card">
  <div class="content">
    <i class="right floated huge ${_icon} ${_colour} icon"></i>
    <h3 class="ui header">${_title}</h3>
    <div class="description">
      <div class="ui text"> ${_valueA}</div>
      <div class="ui text"> ${_valueB}</div>
    </div>
    <div class="description ui two column grid">
      <div class="column">
        <div class="ui text">
          <div class="ui text"> ${_valueC}</div>
          <div class="ui text"> ${_valueD}</div>
          <div class="ui text"> ${_valueE}</div>
        </div>
      </div>
      <div class="column">
        #{if _trend == 1}
        <i class="right floated big arrow up ${_colour} icon"></i>
        #{/if}
        #{elseif _trend == -1}
        <i class="right floated big arrow down ${_colour} icon"></i>
        #{/elseif}
      </div>
    </div>
  </div>
</div>
```

Station

```
public class StationCtrl extends Controller {
  public static void index(Long id) {
    Logger.info("Rendering Dashboard");
    Station station = Station.findById(id);
    Analytics.updateWeather(station);
    render("station.html", station);
  }

  public static void addReading(Long id, int code, double temperature,
                                double windSpeed, int windDirection,
                                int pressure) {
    Logger.info("Adding a Reading");
    Station station = Station.findById(id);
    Reading reading = new Reading(code, temperature, windSpeed, windDirection, pressure);
    station.readings.add(reading);
    Analytics.updateWeather(station);
    station.save();
    redirect("/stations/" + id);
  }

  public static void deleteReading (Long stationid, Long id)
  {
    Logger.info("Deleting a Reading");
    Station station = Station.findById(stationid);
    Reading reading = Reading.findById(id);
    station.readings.remove(reading);
    station.save();
    reading.delete();
    redirect ("/stations/" + stationid);
  }
}
```






```
#{extends 'main.html' /}
#{set title: "${station.name} Weather Station"/}
#{menu id:"dashboard"/}

<section class="ui segment">
  #{station-panel station:station /}
  #{reading-table readings:station.readings, station:station/}
  #{add-reading-form station:station/}
</section>
```

station

```
#{extends 'main.html' /}
#{set title: "${station.name} Weather Station"/}
#{menu id:"dashboard"/}

<section class="ui segment">
  #{station-panel station:station /}
  #{reading-table readings:station.readings, station:station/}
  #{add-reading-form station:station/}
</section>
```

Date/Time	Code	Temp	Wind Speed	Wind Direction	Pressure	
2021-01-19 08:31:00.0	800	0.5	3.5	220	1001	
2021-01-20 09:34:00.0	600	6.0	2.0	200	1004	

reading-table

```
<table class="ui celled table">
  <thead>
    #{reading-header /}
  </thead>
  <tbody>
    #{list items:_station.readings, as:'reading'}
      #{reading-row reading:reading, station:_station /}
    #{/list}
  </tbody>
</table>
```

reading-header

```
<tr>
  <th> Date/Time </th>
  <th> Code </th>
  <th> Temp </th>
  <th> Wind Speed </th>
  <th> Wind Direction </th>
  <th> Pressure </th>
</tr>
```

reading-row

```
<tr>
  <td> ${_reading.date}   </td>
  <td> ${_reading.code}   </td>
  <td> ${_reading.temperature} </td>
  <td> ${_reading.windSpeed} </td>
  <td> ${_reading.windDirection} </td>
  <td> ${_reading.pressure} </td>
  <td>
    <a href="/stations/${_station.id}/deletereading/${_reading.id}" class="ui icon button">
      <button class="ui icon button"> <i class="red trash icon"></i> </button>
    </a>
  </td>
</tr>
```

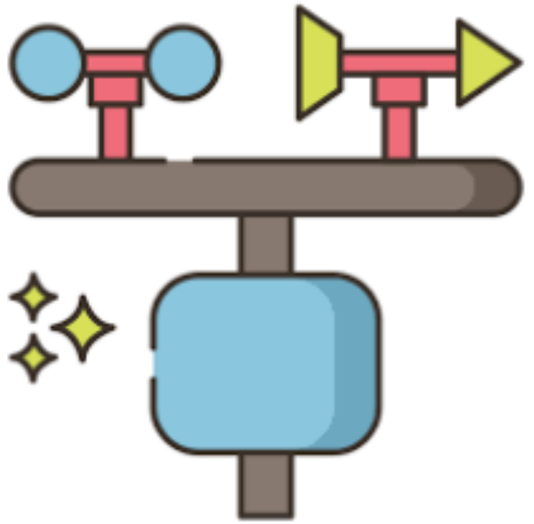
station

```
#{extends 'main.html' /}
#{set title: "${station.name} Weather Station"/}
#{menu id:"dashboard"/}

<section class="ui segment">
  #{station-panel station:station /}
  #{reading-table readings:station.readings, station:station/}
  #{add-reading-form station:station/}
</section>
```

add-reading-form

```
<form class="ui segment form" action="/stations/${_station.id}/addreading" method="POST">
  <div class="three fields">
    <div class="field">
      <label>Code</label>
      <input placeholder="00" type="text" name="code">
    </div>
    <div class="field">
      <label>Temperature</label>
      <input placeholder="00.0C" type="text" name="temperature">
    </div>
    <div class="field">
      <label>Wind Speed</label>
      <input placeholder="00.00kMh" type="text" name="windSpeed">
    </div>
  </div>
  <div class="three fields">
    <div class="field">
      <label>Wind Direction</label>
      <input placeholder="000deg" type="text" name="windDirection">
    </div>
    <div class="field">
      <label>Pressure</label>
      <input placeholder="000hPa" type="text" name="pressure">
    </div>
  </div>
  <button class="ui blue submit button"> Submit Report </button>
</form>
```

WeatherTop



A detailed walkthtough of a Java/Play WeatherTop implementation