Open Source Portlet Incubation

Parker Grimes
Southern Utah University
JA-SIG Spring 2008 Conference
Apr 30, 2008

Agenda

- Introduction
- Why Open Source Portlets?
- Portlet Specific Challenges
- Questions to Ask Yourself
- Useful Guidelines
- Starting Out Open Source
- Case Study: JA-SIG Weather Portlet

Why Open Source Portlets?

- All of the common open source arguments apply.
- Don't re-invent the wheel.
- Share your solution.
- Extend someone elses work.
- Common needs.
- Etc.

Portlet Specific Challenges

- Broad needs
  - Portals are inherently broad in feature scope.
- Institution specific requirements
  - My institution implemented feature X, but do you really want it?
- Portlets typically have a small code base
  - This makes it very tempting to write your own solution, rather than utilize someone elses code.

Questions to Ask Yourself

- Does your portlet fill a common need?
- Is your code easy to understand?
- Can it just be dropped in or will others have to modify your code?
- How configurable is your portlet?
- Do you need to offer more configuration hooks?
- Is your solution generic enough to share?

Useful Guidelines

- Before you write the code, think about your architecture. (If you decide to open source it, how many changes will you have to make so that it is generic enough for others to use?)
- Make portlets as easy to configure as possible.
- Follow good design patters.
- Document code well.
- Allow for internationalization.
- Provide extension points to allow custom modification. (i.e. Code to Interfaces, Implement and Extend don't modify)
- Spring!

Advantage of Starting Out Open Source

- Get buy in from the community.
- Others can help steer the project.
- Clearly define requirements up-front.
- Others can help in the development.

Case Study: JA-SIG Weather Portlet

- Development lead by Dustin Schultz.
- The community asked for a weather portlet that fit the needs of uPortal deployers.
- Lots of weather portlets exist, none fit the requirements.
- The community came up with requirements.
- Lots of discussion on the email lists.

Requirements

- Global weather data
- Internationalization (i18n) and localization (l10n)
- Multiple locations
- Friendly terms of use

Challenges

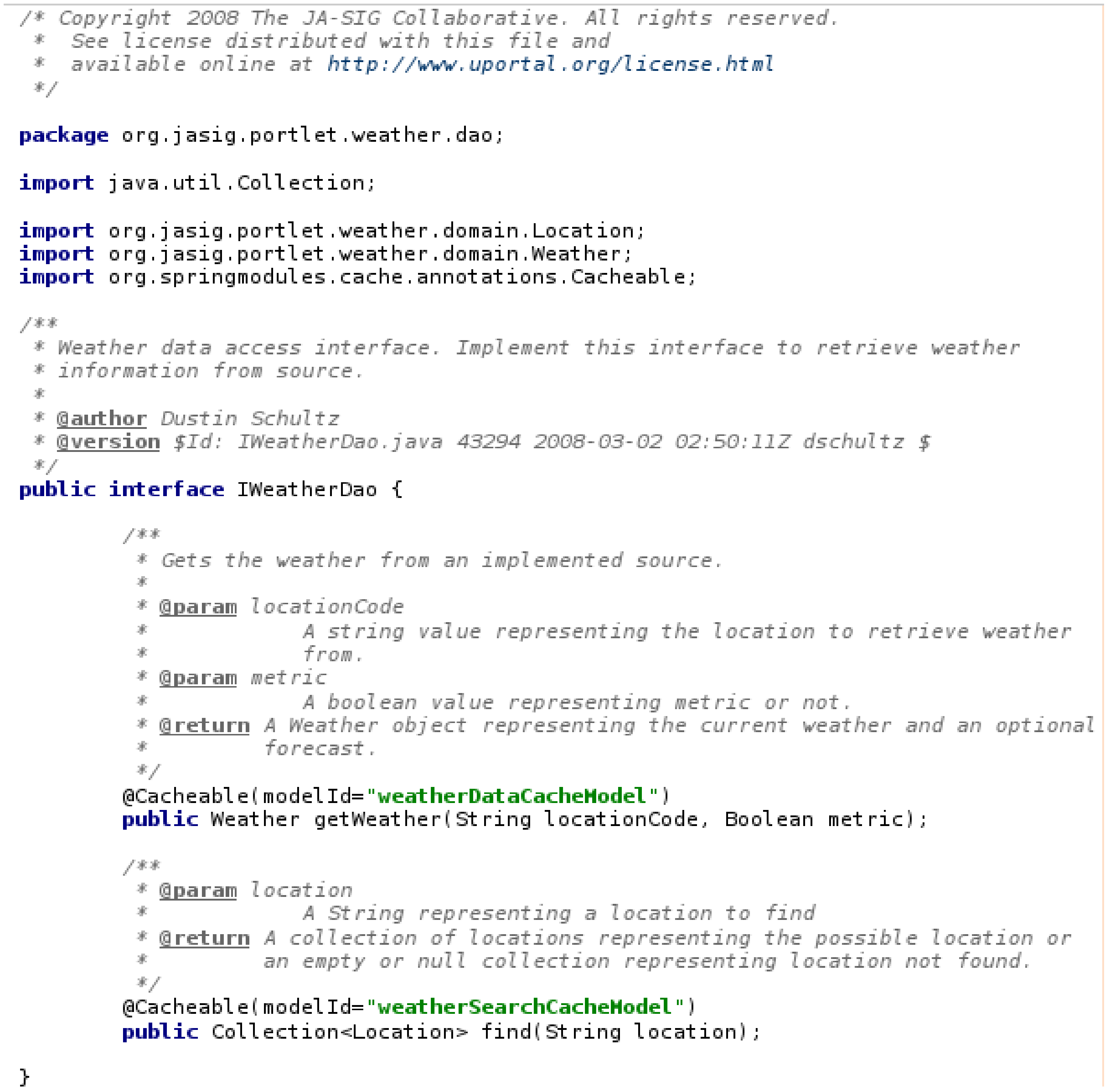
- Finding a suitable weather feed!
  - Yahoo Weather - Difficult to obtain foreign locations, logo requirement, non-commercial use only.
  - NOAA - Only U.S. and surrounding waters. Unfriendly web service.
  - Weather.com - Would need an API key and limited number of requests per month (10,000? or 50,000?), also non-commercial. Requirements on logo size and links.
  - METAR - Difficult to parse, limited locations.
  - Weatherbug.com - Non-free service for dedicated use, non-commercial use.

The Weather Feed

- Weather feed provided by Accuweather
  - Provides rich weather data
  - Weather data from all over the world
  - Custom feed http://uport.accu-weather.com/
  - Most agreeable terms of use
- Special thanks to Michael Sylvie (sylvie@accuweather.com) from Accuweather for setting this all up.

Terms of Use

- “The data feed of AccuWeather is provided to you free of charge in exchange for your promise to display the AccuWeather.com® logo in your application and, if feasible, provide a link which clicks through to AccuWeather.com. Use of the data feed is for personal, non-commercial purposes only... Commercial usage is possible with explicit permission from AccuWeather.com. Please contact developer@accuweather.com for more details.”
- Yes, educational institutions are considered “non-commercial” by Accuweather.
- Not happy with these terms of use?
  - Implement org.jasig.portlet.weather.dao.IWeatherDao using a different weather feed. (only 2 methods to implement)



```
/* Copyright 2008 The JA-SIG Collaborative. All rights reserved.
 *  See license distributed with this file and
 *  available online at http://www.uportal.org/license.html
 */

package org.jasig.portlet.weather.dao;

import java.util.Collection;

import org.jasig.portlet.weather.domain.Location;
import org.jasig.portlet.weather.domain.Weather;
import org.springmodules.cache.annotations.Cacheable;

/**
 * Weather data access interface. Implement this interface to retrieve weather
 * information from source.
 *
 * @author Dustin Schultz
 * @version $Id: IWeatherDao.java 43294 2008-03-02 02:50:11Z dschultz $
 */
public interface IWeatherDao {

        /**
         * Gets the weather from an implemented source.
         *
         * @param locationCode
         *            A string value representing the location to retrieve weather
         *            from.
         * @param metric
         *            A boolean value representing metric or not.
         * @return A Weather object representing the current weather and an optional
         *         forecast.
         */
        @Cacheable(modelId="weatherDataCacheModel")
        public Weather getWeather(String locationCode, Boolean metric);

        /**
         * @param location
         *            A String representing a location to find
         * @return A collection of locations representing the possible location or
         *         an empty or null collection representing location not found.
         */
        @Cacheable(modelId="weatherSearchCacheModel")
        public Collection<Location> find(String location);

}
```

Implementation

- Accuweather feed provides global weather.
- User can choose metric units.
- I18n for display text achieved via Spring Portlet MVC. (we're looking for translators)
- Multiple locations per portlet.
- Locations saved in portlet preferences.
- Displays current conditions +5 day forecast.
- Forecast data dynamically adjusts to the size of the portlet window via CSS.
- Caching using ehcache. 15 min for weather data, 1 hour for location searches, configurable via xml.

Questions?

Parker Grimes
Southern Utah University
grimesp@suu.edu

Dustin Schultz
Southern Utah University
http://www.ja-sig/wiki/display/~dschultz