



Distributed Computing

Homework 5: Web Services

Thomas Weise · 汤卫思

tweise@hfu.edu.cn · <http://www.it-weise.de>

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

- 1 Big Picture
- 2 Weather
- 3 Task



website

- Use Java ^[1-4] and Axis2 ^[5-7] to
- Build a Web Service
- Which makes information from a website accessible as module in a program
- Side effect: learn why HTML and traditional websites are not suitable for information presentation *for computers*

- Web Services are independent components that offer well-defined functionality to an application

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines
- Realization: Transition from websites as presentation+function units to websites for presentation and web services for function

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines
- Realization: Transition from websites as presentation+function units to websites for presentation and web services for function
- This homework: Take an existing website and make its functionality available as web service

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines
- Realization: Transition from websites as presentation+function units to websites for presentation and web services for function
- This homework: Take an existing website and make its functionality available as web service
- Follow all the steps of web service creation and integration

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines
- Realization: Transition from websites as presentation+function units to websites for presentation and web services for function
- This homework: Take an existing website and make its functionality available as web service
- Follow all the steps of web service creation and integration
 - ① service implementation

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines
- Realization: Transition from websites as presentation+function units to websites for presentation and web services for function
- This homework: Take an existing website and make its functionality available as web service
- Follow all the steps of web service creation and integration
 - ① service implementation
 - ② service installation

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines
- Realization: Transition from websites as presentation+function units to websites for presentation and web services for function
- This homework: Take an existing website and make its functionality available as web service
- Follow all the steps of web service creation and integration
 - ① service implementation
 - ② service installation
 - ③ service client creation

- Web Services are independent components that offer well-defined functionality to an application
- Original goal: Allow information and processes designed for human beings to be accessed and processed by machines
- Realization: Transition from websites as presentation+function units to websites for presentation and web services for function
- This homework: Take an existing website and make its functionality available as web service
- Follow all the steps of web service creation and integration
 - ① service implementation
 - ② service installation
 - ③ service client creation
 - ④ service usage

- There exist various websites that can offer us information about the current weather at a place

- There exist various websites that can offer us information about the current weather at a place
- Example: `http://www.timeanddate.com/weather/`

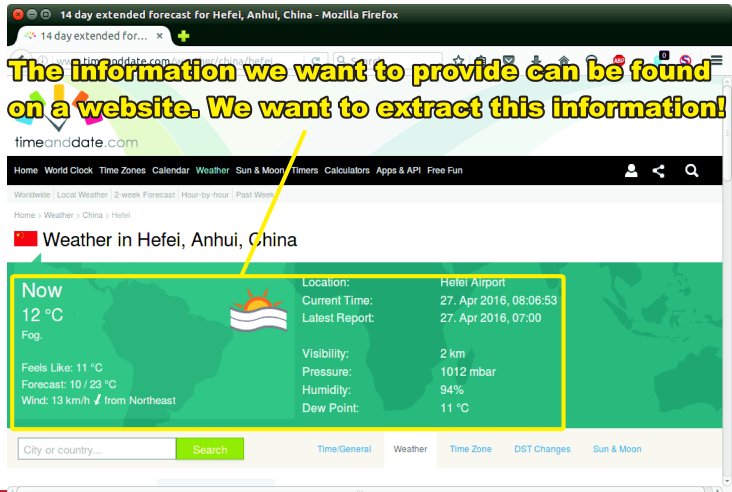
- There exist various websites that can offer us information about the current weather at a place
- Example: `http://www.timeanddate.com/weather/`
- Information provided for a human user as HTML ^[8, 9]

- There exist various websites that can offer us information about the current weather at a place
- Example: `http://www.timeanddate.com/weather/`
- Information provided for a human user as HTML ^[8, 9]
- We can develop a web service that offers us weather information by obtaining it from a web site!

- The weather website allows a human user to search for weather at a given place

- The weather website allows a human user to search for weather at a given place

The information we want to provide can be found on a website. We want to extract this information!





timeanddate.com

Home World Clock Time Zones Calendar Weather Sun & Moon Timers Calculators Apps & API Free Fun

Worldwide Local Weather 2-week Forecast Hour-by-hour Past Week

Home > Weather > China > Hefei

 Weather in Hefei, Anhui, China

Now 12 °C Fog. Feels Like: 11 °C Forecast: 10 / 23 °C Wind: 13 km/h ↗ from Northeast		Location: Hefei Airport Current Time: 27. Apr 2016, 08:06:53 Latest Report: 27. Apr 2016, 07:00 Visibility: 2 km Pressure: 1012 mbar Humidity: 94% Dew Point: 11 °C
---	---	---

City or country... Search Time/General Weather Time Zone DST Changes Sun & Moon

- We are interested in the weather, high and low temperature, as well as humidity

- We are interested in the weather, high and low temperature, as well as humidity

14 day extended forecast for Hefei, Anhui, China - Mozilla Firefox

14 day extended For...

The information we want to provide can be found on a website. We want to extract this information!

timeanddate.com

Home World Clock Time Zones Calendar Weather Sun & Moon Timers Calculators Apps & API Free Fun

Worldwide Local Weather 2-week Forecast Hour-by-hour Past Week

Home > Weather > China > Hefei

Weather in Hefei, Anhui, China

Now
12 °C

weather

Fog

low temperature

Feels Like: 11 °C

Forecast 10 23 °C

Wind: 13 km/h from high temperature

Location: Hefei Airport

Current Time: 27. Apr 2016, 08:06:53

Latest Report: 27. Apr 2016, 07:00

Visibility: 2 km

Pressure: 1012 mbar

Humidity: **humidity** 94 %

Dew Point: 11 °C

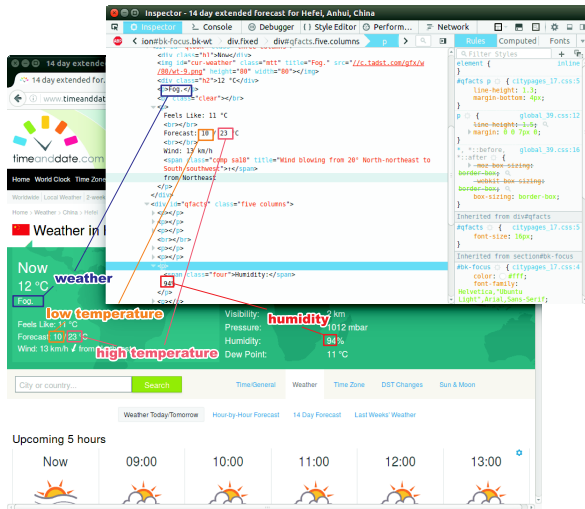
City or country... Search

Time/General Weather Time Zone DST Changes Sun & Moon

Figure: Webpage with interesting information

- HTML document contains human-readable information with info inside

- HTML document contains human-readable information with info inside



The screenshot shows a web browser displaying a weather page for Hefei, Anhui, China. The page includes a search bar, a "Search" button, and tabs for "Weather", "Time Zone", "DST Changes", and "Sun & Moon". The main content area shows the current weather (12°C, fog) and a 14-day forecast. The developer tools are open, showing the HTML source code. Red annotations highlight specific data points: "low temperature" points to the current temperature (12°C), "high temperature" points to the forecast high (23°C), and "humidity" points to the current humidity (94%).

Figure: Webpage + HTML source

- Implement a webservice which has one function

- Implement a webservice which has one function
- that receives a location/place definition as parameter

- Implement a webservice which has one function
- that receives a location/place definition as parameter
- and returns a record describing the current weather in that location

- Implement a webservice which has one function
- that receives a location/place definition as parameter
- and returns a record describing the current weather in that location
- This information is obtained by querying
`http://http://www.timeanddate.com/weather/`

- Implement a webservice which has one function
- that receives a location/place definition as parameter
- and returns a record describing the current weather in that location
- This information is obtained by querying
`http://http://www.timeanddate.com/weather/:`
 - extend the basic class `WeatherService`

- Implement a webservice which has one function
- that receives a location/place definition as parameter
- and returns a record describing the current weather in that location
- This information is obtained by querying `http://http://www.timeanddate.com/weather/:`
 - extend the basic class `WeatherService`
 - you can use a pre-implemented method `String getWebsite(final Location location)` to obtain all the HTML text of the page corresponding to a given `Location`

- Implement a webservice which has one function
- that receives a location/place definition as parameter
- and returns a record describing the current weather in that location
- This information is obtained by querying `http://http://www.timeanddate.com/weather/:`
 - extend the basic class `WeatherService`
 - you can use a pre-implemented method `String getWebsite(final Location location)` to obtain all the HTML text of the page corresponding to a given `Location`
 - We can find the relevant information in that (HTML) string by using e.g., `indexOf`, or something like `JSoup`

- Implement a webservice which has one function
- that receives a location/place definition as parameter
- and returns a record describing the current weather in that location
- This information is obtained by querying `http://http://www.timeanddate.com/weather/:`
 - extend the basic class `WeatherService`
 - you can use a pre-implemented method `String getWebsite(final Location location)` to obtain all the HTML text of the page corresponding to a given `Location`
 - We can find the relevant information in that (HTML) string by using e.g., `indexOf`, or something like `JSoup`
- With the information, an answer record is populated which then returned as service result

- Implement a webservice which has one function
- that receives a location/place definition as parameter
- and returns a record describing the current weather in that location
- This information is obtained by querying `http://http://www.timeanddate.com/weather/:`
 - extend the basic class `WeatherService`
 - you can use a pre-implemented method `String getWebsite(final Location location)` to obtain all the HTML text of the page corresponding to a given `Location`
 - We can find the relevant information in that (HTML) string by using e.g., `indexOf`, or something like `JSoup`
- With the information, an answer record is populated which then returned as service result
- This way, we make an external information source available as building block for an application!

Listing: The location record (JavaBean) (Location.java).

```
package weatherService;

import java.io.Serializable;

/** a location specification */
public class Location implements Serializable {
    /** the city */
    private String m_city;
    /** the province */
    private String m_province;
    /** the country */
    private String m_country;
    public String getCity() {
        return this.m_city;
    }
    public void setCity(final String city) {
        this.m_city = city;
    }
}
```


Listing: The weather record (JavaBean) (Weather.java).

```
package weatherService;

import java.io.Serializable;

/** A weather description */
public class Weather implements Serializable {
    /** the weather: is it sunny, cloudy, windy, or ... */
    private String m_weather;
    /** the temperature high */
    private int m_temperatureHigh;
    /** the temperature low */
    private int m_temperatureLow;
    /** the humidity */
    private int m_humidity;
    public Weather() {
        super();
    }
    /**
    public String getWeather() {
        return this.m_weather;
    }
    public void setWeather(final String weather) {
        this.m_weather = weather;
    }
    public int getHighTemperature() {
        return this.m_temperatureHigh;
    }
    public void setHighTemperature(final int temperature) {
        this.m_temperatureHigh = temperature;
    }
}
```

- The homework has four parts, listed on the following slides

- The homework has four parts, listed on the following slides
- Put everything into an archive called `hw05_[your_student_id].zip` (where `[your_student_id]` is replaced with your student id) and send it to me.

- Implement the weather service by filling code into `weatherService/service/WeatherService.java`

- Implement the weather service by filling code into `weatherService/service/WeatherService.java`
- The service receives one instance of class `Location`

- Implement the weather service by filling code into `weatherService/service/WeatherService.java`
- The service receives one instance of class `Location`
- It builds a query to `http://www.timeanddate.com/weather/` and downloads the answer webpage via the pre-defined method `getWebsite`

- Implement the weather service by filling code into `weatherService/service/WeatherService.java`
- The service receives one instance of class `Location`
- It builds a query to `http://www.timeanddate.com/weather/` and downloads the answer webpage via the pre-defined method `getWebsite`
- From this page, it extracts all the information for the Weather record (see documentation), i.e., the weather description, the low and high temperature, and humidity (marked with colored boxes in Figure 2)

- Implement the weather service by filling code into `weatherService/service/WeatherService.java`
- The service receives one instance of class `Location`
- It builds a query to `http://www.timeanddate.com/weather/` and downloads the answer webpage via the pre-defined method `getWebsite`
- From this page, it extracts all the information for the Weather record (see documentation), i.e., the weather description, the low and high temperature, and humidity (marked with colored boxes in Figure 2)
- The description is stored in an instance of class `Weather` and returned

- Implement the weather service by filling code into `weatherService/service/WeatherService.java`
- The service receives one instance of class `Location`
- It builds a query to `http://www.timeanddate.com/weather/` and downloads the answer webpage via the pre-defined method `getWebsite`
- From this page, it extracts all the information for the Weather record (see documentation), i.e., the weather description, the low and high temperature, and humidity (marked with colored boxes in Figure 2)
- The description is stored in an instance of class `Weather` and returned
- In order to build an `Eclipse / Maven` project for web services, you could copy one of my example server-side web service projects, rename it accordingly (by editing the `pom.xml` and `.project` files), delete the code inside its original `src` folder, and copy the code from the homework's `src` folder there in place, and edit the `services.xml` file.

Listing: The weather service class (WeatherService.java).

```
package weatherService;
public class WeatherService {
    public Weather getWeather(final Location location) {
        final Weather weather;
        weather = new Weather();
        try {
            WeatherService.getWebsite(location);

            // TODO: fill in your code here

        } catch (final Throwable error) {
            /** ignore */
        } finally {
            return weather;
        }
    }
}
```

- Write a suitable `services.xml` file and put it into a `META-INF` folder in your project

- Write a suitable `services.xml` file and put it into a `META-INF` folder in your project
- Create a Maven `pom.xml` file for building the service `aar` archive

- Write a suitable `services.xml` file and put it into a `META-INF` folder in your project
- Create a Maven `pom.xml` file for building the service `aar` archive
- Use Maven to build the `aar` archive

- Write a suitable `services.xml` file and put it into a `META-INF` folder in your project
- Create a Maven `pom.xml` file for building the service `aar` archive
- Use Maven to build the `aar` archive
- Install your new service under Axis2 by copying it into the `services` folder
- (see the examples in the repository and the lecture slides on how to do that)

- Create a new project and Maven `pom.xml` to build a client for our web service

- Create a new project and Maven `pom.xml` to build a client for our web service
- Run the project once to generate the client stub classes for calling the service

- Create a new project and Maven `pom.xml` to build a client for our web service
- Run the project once to generate the client stub classes for calling the service
- In order to build an Eclipse / Maven project for web services, you could copy one of my example client-side web service projects, rename it accordingly (by editing the `pom.xml` and `.project` files), replace the code inside its original `src` folder with your own code.
Remember that you will need to Maven-build first to generate the classes for calling the service, and then you can write your code using them, and then you can compile again to actually build the fat jar.

- Now create code that actually calls the service by using these classes

- Now create code that actually calls the service by using these classes:
 - Make a program whose `main` method takes three command line parameters (the stuff in `String[] args`): the city name, province, and country

- Now create code that actually calls the service by using these classes:
 - Make a program whose `main` method takes three command line parameters (the stuff in `String[] args`): the city name, province, and country
 - This program should fill in a `Location` record with this information and then call your web service using the automatically generated service client stub code

- Now create code that actually calls the service by using these classes:
 - Make a program whose `main` method takes three command line parameters (the stuff in `String[] args`): the city name, province, and country
 - This program should fill in a `Location` record with this information and then call your web service using the automatically generated service client stub code
 - It should print the information about the received weather record to the standard output (`System.out`)

- Now create code that actually calls the service by using these classes:
 - Make a program whose `main` method takes three command line parameters (the stuff in `String[] args`): the city name, province, and country
 - This program should fill in a `Location` record with this information and then call your web service using the automatically generated service client stub code
 - It should print the information about the received weather record to the standard output (`System.out`)
- Modify your Maven `pom.xml` to build a “fat jar” with all the required libraries and classes inside *and* to run your main class

Submit the following things:

- your `zip`-compressed folder with both Eclipse projects (client and server side)
- the `aar` with the service
- the fat `jar` of the client

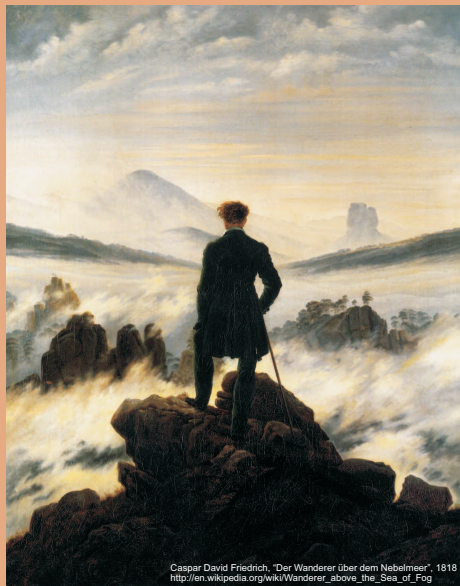
- See the lecture and the documentation of the examples repository!

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfu.edu.cn
<http://www.it-weise.de>

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China



Caspar David Friedrich, "Der Wanderer über dem Nebelmeer", 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog



1. James Gosling, William Nelson Joy, Guy Lewis Steele Jr., and Gilad Bracha. *The Java™ Language Specification*. The Java Series. Upper Saddle River, NJ, USA: Prentice Hall International Inc., Santa Clara, CA, USA: Sun Microsystems Press (SMP), and Reading, MA, USA: Addison-Wesley Professional, 3rd edition, May 2005. ISBN 0-321-24678-0 and 978-0321246783. URL <http://java.sun.com/docs/books/jls/>.
2. James Gosling and Henry McGilton. The java language environment – a white paper. Technical report, Santa Clara, CA, USA: Sun Microsystems, Inc., May 1996. URL <http://java.sun.com/docs/white/langenv/>.
3. Santa Clara, CA, USA: Sun Microsystems, Inc. *Java™ 2 Platform Standard Edition 5.0 – API Specification*, October 19, 2010.
4. Herbert Schildt. *Java 2: A Beginner's Guide*. Essential Skills for First-Time Programmers. Maidenhead, England, UK: McGraw-Hill Ltd., 2002. ISBN 0072225130 and 9780072225136. URL <http://books.google.de/books?id=YWDJJGYaLG4C>.
5. Deepal Jayasinghe. *Quickstart Apache Axis2*. Birmingham, UK: Packt Publishing Limited, 2008. ISBN 1847192866 and 9781847192868. URL <http://books.google.de/books?id=wVdyNwAACAAJ>.
6. Kent Ka lok Tong. *Developing Web Services with Apache Axis2*. Charleston, SC, USA: BookSurge, 2008. ISBN 9993792918 and 9789993792918. URL <http://books.google.de/books?id=0150PwAACAAJ>.
7. *Apache Axis2/Java*. Forest Hill, MD, USA: Apache Software Foundation, 2012. URL <http://axis.apache.org/axis2/java/core/>.
8. Dave Raggett, Arnaud Le Hors, and Ciel J. H. Jacobs. *HTML 4.01 Specification*. W3C Recommendation. MIT/CSAIL (USA), ERCIM (France), Keio University (Japan): World Wide Web Consortium (W3C), December 24, 1999. URL <http://www.w3.org/TR/1999/REC-html401-19991224>.
9. Murray Altheim and Shane McCarron. *XHTML™ 1.1 – Module-based XHTML – Second Edition*. W3C Recommendation. MIT/CSAIL (USA), ERCIM (France), Keio University (Japan): World Wide Web Consortium (W3C), November 23, 2010. URL <http://www.w3.org/TR/2010/REC-xhtml11-20101123>.