

## Chapter 12

# Water Web: Accessing Remote Resources – WSDL

### IN THIS CHAPTER

- ◆ Creating and calling Water Web resources
- ◆ Specifying a Water Web Contract
- ◆ Specifying a Protocol

THERE ARE MANY DIFFERENT TYPES OF RESOURCES (documents, images, services, and so on) that are accessed via protocols such as HTTP, FTP, and SMTP. Remote resources on the Web become local resources in Water Web. A connection to a remote Web resource is an instance of `web`, called a Web resource, and it represents the ability to call that remote Web resource. A Water Web object has all the information necessary to generate a WSDL document. WSDL is an XML standard for how to describe the interface and protocol of a remote resource. A Water Web call is a remote procedure call that occurs over an internal or external network – even the same machine on a network port.



A resource represents the permission (access rights and capabilities) to use some remote resource. A resource can be created from a base resource and a URI/URL.

## Creating and Calling Water Web Resources

The following example sets `current_temperature_boston` to a Web resource. The second line requests the Web resource.

```
<set current_temperature_boston=  
  www.<web "http://weather9.com/current/temperature?city=boston"/>
```

```

/>
current_temperature_boston → "60"

```

The result is "60" because 60 degrees is the current temperature in Boston and this service returns a simple string with the temperature. Please note that the URI in this example does not point to a live Web service.



A *Web service* is a service that uses Web protocols. A service is an entity that supplies something of value.

---



A *Web request* or *Web call* is the use of a Web service.

---

The following example is similar to the preceding example, but instead of specifying the `city` argument in the URI of the resource, the argument is passed when calling the resource. I do not know of any languages other than Water that make it so easy to make and use a Web service. Water Web makes Web services as easy to call as standard Water methods.

```

<set current_temperature=
  www.<web "http://weather9.com/current/temperature"/>
/>
<current_temperature city="boston"/> → "60"

```

Defining and calling a Web resource follows the following structure:

```

<set web_resource=
  www.<web a_uri contract/>
/>
<web_resource arg_1=value_1 arg_N=value_N/>

```

`www` is an object that represents the capability to make a remote call over the Web. That capability (or resource) is named `www`. A new Web resource can be created from the `www` object.



The “`www`” resource is very similar to the “`filesystem`” resource. The object with the capability to access the filesystem is called `filesystem`. There are two specialized resources for `filesystem` called `file` and `folder`. Creating instances of those objects will create an instance of a `filesystem` resource. The `web` resource has a similar structure. `www` is the object with the capability for creating remote resources. An instance of `web` is a resource for accessing a specific remote Web resource.

If a Web resource does not take any arguments or if the arguments have already been given in the URI, then getting the `content` field of a resource will make a remote request.

```
current_temperature_boston.content → "60"
```

Requesting `content` will not add a query, but any query in the original URI of the resource will be sent.

## Specifying a Water Web Contract

A `web` resource can have an associated contract in addition to a URI. This makes it possible to check the names of arguments and their type before making the remote call. The return value will also be type-checked if the contract has a `return_type`.



See Chapter 5 on Water Contract for a full description of a Water contract.

```
<defclass web
  a_uri=required
  contract=optional
/>
```

The following `web` resource has a simple contract defined by using `defmethod`. The resource will only be called if the `city` argument is a string. No other arguments are allowed.

```
www.<web
  <uri "http://weather.com/current/temperature"/>
```

```
contract=<defmethod city=required=string/>
/>
```

## Specifying the Protocol

The protocol used to interact with the remote Web resource is given in the URI. In the following example, `http` is the name of the protocol.

```
www.<web <uri "http://weather.com/" /> />
```

A method that handles that protocol is looked up in the `web` class and the protocol handler method is called every time a request is made to the remote resource. If the protocol method is not found, an error is returned.



*A protocol is the process used to access a resource.*

---

New protocols can be defined with Water Protocol (Chapter 13) and the protocol can be given in the URI. A custom protocol could be created named “`http_9921_water`” that uses HTTP over port 9921 and always returns a Water object. The protocol would be specified as in the following example.

```
www.<web <uri "http_9921_water://weather.com/" /> />
```

The code for implementing the custom protocol could itself be a Web service. The possibilities are endless.

## Summary

To create a Web resource, simply create an instance of `web` with a URI. To request the value of that resource, simply reference the `web` instance. To call the resource with arguments, simply call the web resource as you would any other Water method. If the web resource has a contract, the arguments will be checked against the contract before making the remote call.

Water represents a fundamental change in how Web services are created and used. Water lets you “program the Web” where every remote resource has the feel of a local variable or method. The simplification that Water brings to Web services is significant.