

## Chapter 15

# Water Namespace: Dividing the Web

### IN THIS CHAPTER

- ◆ Creating and Using a Water Namespace
- ◆ Creating and Using an XML Namespace

A NAMESPACE IS A USEFUL DESIGN PATTERN for managing names. Each namespace can have a different owner who is responsible for making sure all the names in the namespace are unique. For any large system, namespaces are essential for managing complexity and providing a unique identifier for objects. Water Namespace supports using Water Path for traversing a namespace as well as the XML Namespace standard from the W3C.

## Using Water Path for Namespaces

The following example defines two classes named `my_company` and `partner_A`. Each class has three fields named `customer`, `purchase_order`, and `line_item`. The class is a namespace for the field keys of the class.

```
<defclass my_company
  customer
  purchase_order
  line_item
/>
<defclass partner_A
  customer
  purchase_order
  line_item
/>
```

A Water Path can be used to distinguish between a customer of `my_company` and a customer of `partner_A`. In the following example, the `customer` is referenced through a namespace.

```
my_company.customer
partner_A.customer
```



A namespace is a space or collection for names.

Every name within a namespace is unique, but the same name could exist in any number of namespaces.

---



Every Water object can play the role of a namespace because every object has a unique collection of fields. Within an object, each field has a unique key.

In Water it is common to have two objects where each object has a method of the same name. The appropriate method is called because the method name is looked up in a namespace. The namespace is just the field of the object.

---

The following examples use a namespace in the call name to create an instance of `my_company.customer`:

```
<my_company.customer name="MDP" />
```

The namespace might be accessed through several namespaces. Use a Water Path to access an object through multiple namespaces.

```
my_company.usa.massachusetts.cambridge.customer
```

A long path gets awkward to read and use – particularly when it is used multiple times. To create a shortcut for the namespace, set a local variable.

```
<set cam=my_company.usa.massachusetts.cambridge/>
```

Now you can use that local variable to refer to the full Water Path.

```
cam.customer
```

Water namespaces can also be used in field values. The following example is an instance of `thing` that has a field `customer_type` with a value of `cam.customer`:

```
<thing customer_type=cam.customer/>
```

Using the full namespace would look like the following:

```
<thing customer_type=my_company.usa.massachusetts.cambridge.customer />
```

## Using XML Namespaces

XML Namespace was added in XML 1.1 to provide a way to distinguish between names in different XML standards. Competing standards, for example, might both define `customer`. If `customer` is used in some XML document, there needs to be a way to tell which `customer` is being used. An XML Namespace is uniquely identified by using a Uniform Resource Identifier (URI).

```
"http://www.my_company.com/namespace/public"
```

A Water Path uses the URI string as the field key for accessing objects defined in that URI namespace.

```
thing."http://www.my_company.com/namespace/public".customer
```



Water allows the key of a field to be quoted when inside a Water Path. For example, `this."foo".customer` is equivalent to `this.foo.customer`. This makes it easier to use a field key that contains special characters or spaces. A URI often contains characters such as a slash, colon, or dot, so quotes are used to treat the entire URI string as a field key.

Using a full namespace URI every time can get very cumbersome. XML Namespaces have a special syntax for creating an alias that represents the full namespace. That alias is used in a prefix. The prefix is the namespace alias followed by a colon. An object name is after the prefix.

```
namespace_alias:object_name
```

To refer to the `customer` object in the `cam` namespace, use

```
cam:customer
```

The alias is defined by using a special argument named `xmlns`. The `xmlns` argument is in the form

```
xmlns:namespace_alias="full_namespace_uri"
```

The following code sets the `cam` alias to the URI `http://www.my_company.com/namespace/public`.

```
<thing xmlns:cam="http://www.my_company.com/namespace/public">  
  <cam:customer name="MDP"/>  
</thing>
```

In the content, the `cam` alias is used in the prefix `cam:` before `customer`. The alias is available to every expression that appears within the top-level expression. The alias can also be used as a prefix for the call in which the alias is defined.

```
<cam:customer xmlns:cam="http://www.my_company.com/namespace/public"
              name="MDP"
/>
```

## Using an Attribute Namespace

The XML Namespace standard says that a namespace can also be applied to individual arguments of a call – more commonly known as attributes of an XML element. The namespace alias and a colon is a prefix on the attribute key.

```
<thing cam:customer="MDP"/>
```

The preceding example uses `cam` as the prefix for the `customer` argument. It's not clear why you would use a namespace prefix on an attribute, but the XML standard supports it. I recommend that you avoid the use of a namespace prefix on an argument. Water will not error, however, when reading XML that uses a prefix. Water considers the namespace prefix as part of the field key, "`cam:customer`", in the preceding example.

## Summary

Namespaces are a convenient way to manage collections of names. A Water object is a namespace because each field key in an object is unique and represents a name in a namespace. A Water Namespace can be defined and accessed using the dot notation of a Water Path, or the colon syntax of XML Namespace.