This `web.xml` file defines the `ch17` servlet of the class `DispatcherServlet` that maps to all requests to `*.html` or `*.tile`.

■ **Note** We usually create a mapping to `*.html` because it is a recognized extension and easily fools search engines into thinking that the page is not dynamically generated.

Using Handler Mappings

How does our web application know which servlet (controller implementation) to invoke on a specific request? This is where Spring handler mappings kick in. In a few easy steps, you can configure URL mappings to Spring controllers. All you need to do is edit the Spring application context file.

Spring uses `HandlerMapping` implementations to identify the controller to invoke and provides three implementations of `HandlerMapping`, as shown in Table 17-1.

Table 17-1. *HandlerMapping Implementations*

HandlerMapping	Description
`BeanNameUrlHandlerMapping`	The bean name is identified by the URL. If the URL were `/product/index.html`, the controller bean ID that handles this mapping would have to be set to `/product/index.html`. This mapping is useful for small applications, as it does not support wildcards in the requests.
`SimpleUrlHandlerMapping`	This handler mapping allows you to specify in the requests (using full names and wildcards) which controller is going to handle the request.
`ControllerClassNameHandlerMapping`	This handler mapping is part of the convenience over configuration approach introduced with Spring 2.5. It automatically generates URL paths from the class names of the controllers. This implementation is covered in more detail later in this chapter.

All three `HandlerMapping` implementations extend the `AbstractHandlerMapping` base class and share the following properties:

- `interceptors`: This property indicates the list of interceptors to use. `HandlerInterceptors` are discussed in the next section.
- `defaultHandler`: This property specifies the default handler to use when this handler mapping does not result in a matching handler.
- `order`: Based on the value of the `order` property (see the `org.springframework.core.Ordered` interface), Spring will sort all handler mappings available in the context and apply the first matching handler.
- `alwaysUseFullPath`: If this property is set to `true`, Spring will use the full path within the current servlet context to find an appropriate handler. If this property is set to `false` (the default), the path within the current servlet mapping will be used. For example, if a servlet is mapped using `/testing/*` and the `alwaysUseFullPath` property is set to `true`, `/testing/viewPage.html` would be used, whereas if the property is set to `false`, `/viewPage.html` would be used.
- `urlPathHelper`: Using this property, you can tweak the `UrlPathHelper` used when inspecting URLs. Normally, you shouldn't have to change the default value.

- `urlDecode`: The default value for this property is `false`. The `HttpServletRequest` returns request URLs and URIs that are *not* decoded. If you do want them to be decoded before a `HandlerMapping` uses them to find an appropriate handler, you have to set this to `true` (which requires JDK 1.4). The decoding method uses either the encoding specified by the request or the default ISO-8859-1 encoding scheme.
- `lazyInitHandlers`: This allows for lazy initialization of *singleton* handlers (prototype handlers are always lazily initialized). The default value is `false`.

■ **Note** The last four properties are only available to subclasses of `org.springframework.web.servlet.handler.AbstractUrlHandlerMapping`.

We will start with the example of `BeanNameUrlHandlerMapping`. This is the simple `HandlerMapping` implementation that maps controller bean IDs to the servlet URLs. This `HandlerMapping` implementation is used by default if no `HandlerMapping` is defined in the Spring context files. Listing 17-2 shows an example of the `BeanNameUrlHandlerMapping` configuration, without the actual `BeanNameUrlHandlerMapping` bean (`DispatcherServlet` will instantiate it by default, if no other `HandlerMapping` has been configured).

Listing 17-2. *BeanNameUrlHandlerMapping Configuration*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
...>

<beans>
    <bean name="/index.html.form" ➥
class=" com.apress.prospring.ch17.web.IndexController "/>
</beans>
```

`SimpleUrlHandlerMapping` offers more flexibility in the request mappings. You can configure the mapping as key/value properties in the `publicUrlMapping` bean. In Listing 17-3, you can see a simple example of a Spring application context file containing the handler mapping configuration.

Listing 17-3. *SimpleUrlHandlerMapping Definitions*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
...>

<beans>
    <bean id="publicUrlMapping"
        class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
                <property name="mappings">
            <value>
                /index.html=indexController
                /product/index.html=productController
                /product/view.html=productController
                /product/edit.html=productFormController
            </value>
        </property>
    </bean>
</beans>
```