# HTTPS



This page describes Tapestry's mechanism for automatically switching between HTTP and HTTPS URLs. With the recent trend to have all web sites use HTTPS, you will likely want to disable this behavior. To do so, set the tapestry.secure-enabled configuration symbol to *false* (counter-intuitively).

By default, Tapestry assumes your application will be primarily deployed as a standard web application, using HTTP (not HTTPS) as the primary protocol.

Many applications will need to have some of their pages secured: only accessible via HTTPS. This could be a login page, or a product ordering wizard, or administrative pages.

All that is necessary to mark a page as secure is to add the @Secure annotation to the page class:

```
@Secure
public class ProcessOrder
{
  . . .
}
```

When a page is marked as secure, Tapestry will ensure that access to that page uses HTTPS. All links to the page will use the "https" protocol.

If an attempt is made to access a secure page using a non-secure request (a normal HTTP request), Tapestry will send an HTTPS redirect to the client.

Links to non-secure pages from a secure page will do the reverse: a complete URL with an "http" protocol will be used. In other words, Tapestry manages the transition from insecure to secure and back again.

Links to other (secure) pages *and to assets* will be based on relative URLs and, therefore, secure.

The rationale behind using secure links to assets from secure pages is that it prevents the client web browser from reporting a mixed security level.

## Securing Multiple Pages

Rather than placing an @Secure annotation on individual pages, it is possible to enable https URL redirecting for entire folders of pages. All pages in or beneath the folder will be secured.

This is accomplished by making a contribution to the MetaDataLocator service configuration. For example, to secure all pages in the "admin" folder:

**AppModule.java (partial)**

```
public void contributeMetaDataLocator(MappedConfiguration<String,String> configuration)
{
    configuration.add("admin:" + MetaDataConstants.SECURE_PAGE, "true");
}
```

Here "admin" is the folder name, and the colon is a separator between the folder name and the the meta data key. SECURE_PAGE is a public constant for value "tapestry.secure-page";

When Tapestry is determining if a page is secure or not, it starts by checking for the @Secure annotation, then it consults the MetaDataLocator service.

If you want to make your entire application secure:

**AppModule.java (partial)**

```
public void contributeMetaDataLocator(MappedConfiguration<String,String> configuration)
{
    configuration.add(MetaDataConstants.SECURE_PAGE, "true");
}
```

With no colon, the meta data applies to the entire application (including any component libraries used in the application).

## Base URL Support

When Tapestry switches back and forth between secure and unsecure mode, it must create a full URL (rather than a relative URL) that identifies the protocol, server host name and perhaps even a port number.

### Related Articles

- HTTPS
- Security FAQ
- Security

That can be a stumbling point, especially the server host name. In a cluster, behind a fire wall, the server host name available to Tapestry, via the `HttpServletRequest.getServerName()` method, is often *not* the server name the client web browser sees ... instead it is the name of the internal server behind the firewall. The firewall server has the correct name from the web browser's point of view.

Because of this, Tapestry includes a hook to allow you to override how these default URLs are created: the BaseURLSource service.

The default implementation is based on just the getServerName() method; it's often not the correct choice even for development.

Fortunately, it is very easy to override this implementation. Here's an example of an override that uses the default port numbers that the Jetty servlet container uses for normal HTTP (port 8080) and for secure HTTPS (port 8443):

**AppModule.java (partial)**

```
public static void contributeServiceOverride(MappedConfiguration<Class,Object> configuration)
{
    BaseURLSource source = new BaseURLSource()
    {
        public String getBaseURL(boolean secure)
        {
            String protocol = secure ? "https" : "http";

            int port = secure ? 8443 : 8080;

            return String.format("%s://localhost:%d", protocol, port);
        }
    };

    configuration.add(BaseURLSource.class, source);
}
```

This override is hardcoded to generate URLs for localhost; as such you might use it for development but certainly not in production.

## Development Mode

When working in development mode, the Secure annotation is ignored. This is controlled by the tapestry.secure-enabled configuration symbol.

## Application Server Configuration

Setting up HTTPS support varies from application server to application server.

- Jetty:
  - Versions 7, 8 or 9
- Tomcat:
  - Tomcat 7
  - Version 6.0
  - Version 5.5