Tapestry5HowToControlAccess

Note: the new cleaner way to control access is to use ComponentRequestFilter instead of Dispatcher. You can see the sources of chenillekit-access for example, partically Component{{`Request}}Access`Filter.java

---

In the articles Tapestry5HowToCreateADispatcher and Tapestry5HowToCreateADispatcher2, Chris Lewis has explained how to create a dispatcher, get it into the pipeline, and use it to check user rights against the url requested. What i need in my project is similar but i would rather check the rights of my users against an information stored in the page requested, say an annotation !

Page rights as an annotation

Our first task is to create a @Private annotation that we will put on pages where the authentication of the user is required. Our annotation will be associated to pages and will have no parameters. So, its implementation will be as simple as :

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Private
{
}
```

User rights in the user session

The state of the user will be stored in his session. To keep it simple, the status of the user will be "logged in" or "logged out". So the interface of the user session will display a function to get the user state :

```
public interface UserSession
{
  ...
  public abstract boolean isUserLoggedIn();
  ...
}
```

The problem

With those implementations in hand, what we have to do is to get the Tapestry page associated with the request of the dispatcher and check on this page the presence of the @Private annotation.

The solution

We will just have to parse the path of the request and check it against the Tapestry ComponentClassResolver service (like in the Tapestry PageRenderDispatcher) to find the page name. Having the page name, we can get the Tapestry page from the Tapestry RequestPageCache. After we get the Tapestry Page, we can check the presence of our @Private annotation :

```
public class AccessController implements Dispatcher
{
  private final static String LOGIN_PAGE = "/login";

  private ApplicationStateManager asm;
  private UserSession newUserSession;
  private final ComponentClassResolver resolver;
  private final ComponentSource componentSource;

  /**
   * Receive all the services needed as constructor arguments. When we bind this
   * service, T5 IoC will provide all the services !
   */
  public AccessController(ApplicationStateManager asm, ComponentClassResolver resolver, ComponentSource
componentSource, UserSession newUserSession)
  {
    this.asm = asm;
    this.newUserSession = newUserSession;
    this.resolver = resolver;
    this.componentSource = componentSource;
  }
```

```
  public boolean dispatch(Request request, Response response) throws IOException
  {
    /* We need to get the Tapestry page requested by the user.
     * So we parse the path extracted from the request
     */
    String path = request.getPath();
    if (path.equals("")) return false;

    int nextslashx = path.length();
    String pageName;

    while (true)
    {
      pageName = path.substring(1, nextslashx);
      if (!pageName.endsWith("/") && resolver.isPageName(pageName)) break;
      nextslashx = path.lastIndexOf('/', nextslashx - 1);
      if (nextslashx <= 1) return false;
    }
    return checkAccess(pageName, request, response);
  }

  /**
   * Check the rights of the user for the page requested
   */
  public boolean checkAccess(String pageName, Request request, Response response) {

    boolean canAccess = true;

    /* Is the requested page private ? */
    Component page = componentSource.getPage(pageName);
    boolean privatePage = page.getClass().getAnnotation( Private.class ) != null;

    if (privatePage)
    {
      canAccess = false;
      /* Is the user already authentified ? */
      if(asm.exists(UserSessionImpl.class))
      {
        UserSessionImpl userSession = asm.get(UserSessionImpl.class);
        canAccess = userSession.isUserLoggedIn();
      }
    }

    /* This page can't be requested by a non authentified user => we redirect him on the signon page */
    if(!canAccess) {
      response.sendRedirect(request.getContextPath() + LOGIN_PAGE);
      return true; //Make sure to leave the chain
    }

    return false;
  }
}
```

Conclusion

We have made the choice to use an annotation to set the page status but after we get the Tapestry page in the dispatcher, any other information of the page can be checked. An interesting evolution would be to keep the initial request in the user session to be able to redirect the user on it after he has logged in ... Any suggestions, corrections, or other ideas on this document are welcome. Feel free to contact me on sdecleire@cariboo-networks.com (Stephane Decleire).