

Writing Tests

Table of contents

1 Running Tests.....	2
2 Adding Tests to Modules.....	2
3 Testing Usecases.....	2

1. Running Tests

- To run all tests:
`./build.sh test`
- To run all module tests:
`./build.sh modules.test`
- To run the tests of a single module:
`./build.sh module.test -Dmodule=org.myproject.lenya.modules.MyModule`

2. Adding Tests to Modules

Adding unit tests to a module is very simple - you just have to put a Java file in the `{yourmodule}/java/test`. The most convenient way to get started is to extend `AbstractAccessControlTest`. This class provides the infrastructure to create a session and invoke operations on documents.

Here's an example:

`mymodule/java/test/MyModuleTest.java`

The source code:

```
package org.myproject.lenya;

import org.apache.lenya.ac.impl.AbstractAccessControlTest;
import org.apache.lenya.cms.publication.Document;
import org.apache.lenya.cms.publication.DocumentFactory;
import org.apache.lenya.cms.publication.DocumentUtil;
import org.apache.lenya.cms.publication.Publication;
import org.apache.lenya.cms.publication.PublicationUtil;
import org.apache.lenya.cms.repository.RepositoryUtil;
import org.apache.lenya.cms.repository.Session;

public class MetaDataTest extends AbstractAccessControlTest {

    /**
     * Tests my module.
     */
    public void testMyModule() throws Exception {
        Session session = RepositoryUtil.getSession(getManager(), getRequest());
        DocumentFactory factory = DocumentUtil.createDocumentIdentityMap(getManager(), session);

        Publication publication = PublicationUtil.getPublication(getManager(), "test");
        Document doc = factory.get(publication, Publication.AUTHORING_AREA, "/index", "en");

        assertNotNull(doc);
        ...
    }
}
```

3. Testing Usecases

To implement a unit test for a usecase, you can extend the class `AbstractUsecaseTest` and override the following methods:

- `String getUsecaseName()` - return the name of the usecase to test
- `void prepareUsecase()` - setup the initial environment
- `Map getParameters()` - return a map containing the usecase parameters
- `void checkPostconditions()` - check the post conditions after the usecase was executed

Here's an example:

```

package org.apache.lenya.cms.ac.usecases;

import java.util.HashMap;
import java.util.Map;

import org.apache.cocoon.environment.Session;
import org.apache.lenya.ac.AccessControlException;
import org.apache.lenya.ac.Identity;
import org.apache.lenya.ac.User;
import org.apache.lenya.cms.usecase.AbstractUsecaseTest;

/**
 * Login test.
 */
public class LoginTest extends AbstractUsecaseTest {

    protected static final String USER_ID = "lenya";
    protected static final String PASSWORD = "levi";

    protected Map getRequestParameters() {
        return getParameters();
    }

    protected Map getParameters() {
        Map params = new HashMap();
        params.put(Login.USERNAME, USER_ID);
        params.put(Login.PASSWORD, PASSWORD);
        return params;
    }

    protected String getUsecaseName() {
        return "ac.login";
    }

    protected void checkPostconditions() {
        Session session = getRequest().getSession();
        Identity identity = (Identity) session.getAttribute(Identity.class.getName());
        User user = identity.getUser();
        assertNotNull(user);
        assertEquals(user.getId(), USER_ID);
    }

    protected void login() throws AccessControlException {
        getAccessController().setupIdentity(getRequest());
    }
}

```