Pimp My DSpace

A Case Study in Extending DSpace
Customizing DSpace

- Altering the config files
- Making changes to JSPs
- Creating a “Plug-in”
- Editing the source directly
What We’ll Cover

• Changing JSPs
• Custom Authentication Plug-In
• Statistics
User Interface

- [dspace-source]/jsp
- don’t edit these files
- [dspace-source]/jsp/local
- create new files and directories with the same names as in /jsp and they will be replaced during the build process
User Interface

- Changing the site layout.
  - in jsp/layout/
  - cp jsp/layout/*.jsp jsp/layout/local/
  - changes are:
    - 1) portable
    - 2) self-documenting
DSpace JSPs

• DSpace Custom Tag Library
• Text Strings in [dspace]/config/
  Message.properties from [dspace-source]/
  config/language-packs/
Custom Authentication

• CSTA wants members to use their member number during log-in so they can automatically determine their status and grant privileges

• Let’s make a Plug-In
Plug-Ins

- Source files are in [dspace-source]/src
- default dspace package:
  - org.dspace.*
  - src/org/dspace
- create a new package for custom code
  - package csta;
  - src/csta
Custom Authentication

- Considerations
  - Users will log in with username as email and special password
  - There is a url the given the password as a GET parameter will return a string containing the ACTIVE, EXPIRED or UNKOWN
Considerations

- Email:
  - doesn’t matter, since the get url doesn’t need it
  - However, we’re going to need something that issues a GET request and parses the response.
Solution

- **CstaValidator** - handles HTTP communication with CSTA server
- **CstaAuthServlet** - display login page and accept login credentials
- **CstaAuthentication** - the ‘plug-in’ that gets called by CstaAuthServlet and uses CstaValidator to validate login credentials
CstaValidator

httpURLConnection

https://xxx.xxx?client_no=PASSWORD
(GET parameter)

Csta Validator

External Server

ACTIVE, EXPIRED, or UNKNOWN
public class CstaValidator {

    //passes password to askCstaServer
    //then parses response with string.contains()
    public static boolean isCstaMember(String password)

    //creates HttpURLConnection urlc with member_number appended as GET
    //parameter, then calls urlc.connect();
    //reads the returned IO object into a string and returns it
    private static String askCstaServer(String member_number)
}

CstaAuthServlet

AuthServlet:
doDSGet
  Show Login

doDSPost
  Pass credentials to
  AuthenticationMethod

CstaAuthentication
authenticate
return a status int

Context, Login, Password

status integer
Context?

- All interactions to a database occur within a context--a user with credentials.
- Database interactions will be queued until context.commit() or context.close() is called.
- context.commit() keeps the context open.
org.dspace.core.Context

/** Database connection */
private Connection connection;

/** Current user - null means anonymous access */
private EPerson currentUser;

/** Extra log info */
private String extraLogInfo;

/** Indicates whether authorisation subsystem should be ignored */
private boolean ignoreAuth;

/** Object cache for this context */
private Map objectCache;

/** Group IDs of special groups user is a member of */
private List specialGroups;
```java
int status = AuthenticationManager.authenticate(context, email, password, null, request);

if (status == AuthenticationMethod.SUCCESS)
{
   // Logged in OK.
   Authenticate.loggedIn(context, request, context.getCurrentUser());

   log.info(LogManager.getHeader(context, "login", "type=explicit"));

   // resume previous request
   Authenticate.resumeInterruptedRequest(request, response);

   return;
}
else if (status == AuthenticationMethod.CERT_REQUIRED)
   jsp = "/error/require-certificate.jsp";
else
   jsp = "/login/incorrect.jsp";

// If we reach here, supplied email/password was duff.
log.info(LogManager.getHeader(context, "failed_login",
    "email=" + email + ", result=" + String.valueOf(status)));
JSPManager.showJSP(request, response, jsp);
```
Register the Servlet

- registering servlets
- [dspace-source]/etc/dspace-web.xml

```xml
<servlet>
  <servlet-name>csta-login</servlet-name>
  <servlet-class>csta.CstaAuthServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>csta-login</servlet-name>
  <url-pattern>/csta-login</url-pattern>
</servlet-mapping>
```
Authentication

- in [dspace]/config/dspace.cfg

```
####### Stackable Authentication Methods #######
# Stack of authentication methods
# (See org.dspace.eperson.AuthenticationManager)
plugin.sequence.org.dspace.eperson.AuthenticationMethod = \
csta.CstaAuthentication
```
public interface AuthenticationMethod {
    public int authenticate(Context context,
                            String username,
                            String password,
                            String realm,
                            HttpServletRequest request)
            throws SQLException;
}
CstaAuthentication

• Logging in as a CSTA member creates a new account
• 1) Determine membership status
• 2) Try to log the person in normally
• 3) If they are a CSTA member, create a new account with appropriate privileges.
Privileges

• Let DSpace do it!

• Created a group that all the CSTA members will belong to. Make sure they get added to this group on creation.
CstaAuthentication

```java
EPerson auth = EPerson.findByEmail(context, "christopher.continanza@gmail.com");
context.setCurrentUser(auth);

//create new Eperson
EPerson e = EPerson.create(context);
Group group = Group.find(context, CSTA_MEMBERS_ID); //"CSTA MEMBERS"

e.setEmail(username);
e.setCanLogIn(true);
e.setFirstName("New");
e.setLastName("User" + e.getID());
e.setPassword(password);
e.update();
group.addMember(e);
group.update();
context.commit();

currentUser.set(EUser(e));
return SUCCESS;
```
Extensions

- Restricting Access once you’re expired
  - 1) use the lookup to block access
  - 2) use group privileges to not allow members to change their number
Statistics

- Java program that processes the logs
- Run by Perl scripts
  - which need customization
  - no automation-- must be added as a cron job
Statistics

- There are six Perl scripts in [dspace]/bin:
  - stat-initial
  - stat-general
  - stat-monthly
  - stat-report-initial
  - stat-report-general
  - stat-report-monthly
Order Counts

- Must run stat-initial once, then stat-* to generate specific reports
- Same with stat-report-initial and stat-report series
- Run them every time you’d like new statistics
Statistics

- Customize those scripts (each one):

```python
# Details used
$out_prefix = "dspace-log-general-";
$out_suffix = "dat";
$dsrun = "/dspace/bin/dsrun"
$out_directory = "/dspace/log/"
```
Statistics

- Customize the config file
  `[dspace]/config/dstat.cfg`

```bash
# the log directory to be analysed
dspace.log=[dspace]/log

# the name and url of the service being reported on
host.name=CSTA Test Site
host.url=http://localhost:8080/
```
What we’ve learned

• Making changes to Config Files
• Making Changes to JSPs
• Making Plug-Ins
• That’s all folks...