Content Handler Module Design

Content Handler Module Design

The Content Handler module receives data from Interface modules along with the content type of that data. Using this information it is expected to forward data to an appropriate Processor module that can handle that data. It is also required to have functionality to allow Processor modules to be able to dynamically register and de-register their interest to data of particular content types.

The module is available at: https://github.com/jembi/openmrs-module-shr-contenthandler.

Implementation

The Content Handler module's purpose, at an implementation level, is to:

- Provide generic interfaces to Interface modules for abstracting the operations that can be performed by Processor modules
- Provide functionality for Processor modules to register/deregister content handlers for specific content (MIME) types at runtime
- Provide a content handler for handling unstructured data, which will also be used as a default for any unknown content types.

See the below interfaces; Processor modules can register handlers for specific content types during module startup (or whenever deemed appropriate by the module). The content handlers need to provide functionality to save and retrieve content. Interface handlers can simply then retrieve a registered instance based on the content type of the data they need to handle. There can only be a single handler registered for a specific content type.

The module follows the prototype pattern for content handler instantiation. When registering a handler, a Processor module provides an instance of a handler. Then whenever the getContentHandler method is called, the Content Handler module will clone an instance of the handler for use by the caller.

In the event that there are no handlers registered for a specific type, the module will store the payload as an unstructured "blob" in the OpenMRS database. Any document-type data such as PDFs or image data can be stored this way. Blobs will be saved as observations linked to a new encounter for the specified patient. The Unstructured Storage module provides appropriate obs handlers for this functionality, and therefore the Content Handler module will not handle the implementation. It will simply treat an unstructured payload as a complex observation, but will however provide a code handler instance for use by the getContentHandler method.

Interfaces

Source

ContentHandlerService

```
public interface ContentHandlerService extends OpenmrsService {

        /**
         * Returns a content handler for a specified content type.
         * Will return a default handler for unknown types.
         */
        ContentHandler getContentHandler(String contentType);

        /**
         * Register a content handler for a specified content type.
         */
        void registerContentHandler(String contentType, ContentHandler prototype) throws
AlreadyRegisteredException, InvalidContentTypeException;

        /**
         * Deregister the current handler assigned for the specified content type.
         *
         * This method should be called by processor modules on shutdown.
         */
        void deregisterContentHandler(String contentType);
}
```

Source

ContentHandler

```
public interface ContentHandler {

        /**
         * Parse and store clinical content for the specified patient.
         */
        Encounter saveContent(Patient patient, Provider provider, EncounterRole role, EncounterType
encounterType, Content content);


        /**
         * Retrieve the content associated with the specified encounter uuid.
         */
        Content fetchContent(String encounterUuid);

        /**
         * Retrieve the content associated with the specified encounter id.
         */
        Content fetchContent(int encounterId);


        /**
         * Retrieve a list of formatted encounters for a specified patient.
         */
        List<Content> queryEncounters(Patient patient, Date from, Date to);

        /**
         * Retrieve a list of formatted encounters for a specified patient.
         */
        List<Content> queryEncounters(Patient patient, List<EncounterType> encounterTypes, Date from, Date to);


        /**
         * Create a clone of this handler.
         */
        ContentHandler cloneHandler();
}
```