Merging extension points

In order to allow implementers to add custom behaviour with minimal effort, the Sync2 module includes an extendable interface along with several objects useful with dealing with merging results.

To implement your own merging behaviour, you have to create a new class which will implement the MergeBehaviour interface.

MergeBehaviour

```
package org.openmrs.module.fhir.api.merge;

/**
 * <h1>MergeBehaviour</h1>
 * Reflects merging behaviour for type T.
 * <p><b>Note:</b> use Object.class as a type T for more generic merging behaviour.</p>
 *
 * @see <a href="https://issues.openmrs.org/browse/SYNCT-243">SYNCT-243</a>
 * @since 1.15.0
 */
public interface MergeBehaviour<T> {
        /**
         * <p>Resolves merge conflicts for entities' synchronization.</p>
         *
         * @param local represents local version of an entity
         * @param foreign represents foreign version of an entity
         * @param clazz represents specific class which extends T
         * @return returns a result which depends on implemented merging behaviour
         */
        MergeResult<T> resolveDiff(Class<? extends T> clazz, T local, T foreign);
}
```

MergeResult is an abstract object and it is extended by either MergeSuccess or MergeConflict. The MergeConflict represents a situation when MergeBehaviour couldn't merge objects automatically. The MergeSuccess represents situations when:
- local object overrides another,
- foreign object overrides another,
- local and foreign objects are merged,
- we do not have to change any objects.

An example implementation might look like this:

An example implementation

```
package org.openmrs.module.fhir.api.merge;

import org.openmrs.BaseOpenmrsData;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class TestMergeBehaviour implements MergeBehaviour<BaseOpenmrsData> {

   @Override
   public MergeResult<BaseOpenmrsData> resolveDiff(Class<? extends BaseOpenmrsData> clazz,
                                       BaseOpenmrsData local, BaseOpenmrsData foreign) {
      MergeResult<BaseOpenmrsData> result;
      BaseOpenmrsData merged;

      if (local.getDateChanged().after(foreign.getDateChanged())) {
         merged = deepClone(foreign);
         result = new MergeSuccess<>(clazz, local, foreign, merged, true, false);
      } else {
         result = new MergeConflict<>(clazz, local, foreign);
      }

      return result;
   }

   private BaseOpenmrsData deepClone(BaseOpenmrsData org) {
      BaseOpenmrsData clone = null;
      try {
         ByteArrayOutputStream baOUT = new ByteArrayOutputStream();
         ObjectOutputStream oOUT = new ObjectOutputStream(baOUT);
         oOUT.writeObject(org);

         ByteArrayInputStream baIN = new ByteArrayInputStream(baOUT.toByteArray());
         ObjectInputStream oIN = new ObjectInputStream(baIN);
         clone = (BaseOpenmrsData) oIN.readObject();
      } catch (IOException | ClassNotFoundException e) {
         e.printStackTrace();
      }
      return clone;
   }
}
```

All the code mentioned above is placed in source and test org.openmrs.module.fhir.api.merge package along with the usage example.