Google App Engine의 DataStore의 Blob Type을 활용해 파일 업로드하기

Google App Engine(이하 GAE)의 spin up time 문제도 해결했기 때문에 주말 동안 탄력을 받아 그 동안 미뤄두고 있었던 이미지와 파일 업로드가 가능하도록 구현하기 위한 작업을 시작했다.

처음에는 Blobstore Java API Overview 문서를 참고해 GAE의 BlobStoreService API를 활용하는 방법으로 접근했다. 이 문서에서 제공하는 샘플 예제를 활용해 테스트도 해보니 생각보다 쉽게 해결할 수 있었다. 그런데 BlobStoreService API를 활용할 경우 업로드한 파일의 메타 정보를 알 방법이 없었다. 이 때는 BlobStoreService 에 대하여 잘 모르는 상태였기 때문에 이 같은 판단을 했다. 다음 글에서 BlobStoreService API를 활용해 업로드한 파일의 메타 정보를 추출하는 방법을 방법에 대하여 살펴보도록 하겠다 .

그래서 GAE에서 이미지, 파일을 업로드하기 위한 새로운 방법이 없는지 찾던 중 GAE의 DataStore에 Blob Type을 제공하고 있으며, GAE의 파일 업로드를 Spring MVC에서 파일을 업로드할 때와 같은 방법으로 개발이 가능하도록 지원하는 gmultipart라는 라이브러리가 존재하는 것을 확인하고 적용하기 시작했다. 사실 gmultipart 라이브러리는 더 이상 추가적인 개발이 되지 않고 있는 상황이라 찜찜했지만 일단 GAE의 Blob Type 활용이 가능한지를 검토하기 위해 이
기반으로 개발을 시작했다.

gmultipart 라이브러리를 활용한 샘플 예제는 gmultipart에 올라와 있는 예제를 그대로 활용했다. 일단은 테스트하는 것이 목적이었기 때문에 최대한 단순하게 테스트를 진행했다.

```
[...   ...]

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

import com.google.appengine.api.datastore.Blob;

@Controller
public class FileController {
  @Autowired
  private AttachedFileDao attachedFileDao;

  @RequestMapping(value = "/save", method = RequestMethod.POST)
  public String addComment(CommentForm commentForm) throws IOException {
    MultipartFile[] files = commentForm.getFiles();

    AttachedFile attachedFile = null;
    MultipartFile file = files[0];
    Blob attachment = new Blob(file.getBytes());
    attachedFile = new AttachedFile(ServiceType.diaries, 1L,
file.getOriginalFilename(), file.getSize(),
            attachment, DateTimeUtils.now());
    attachedFileDao.put(attachedFile);

    return "redirect:/show/file/" + attachedFile.getId();
  }
}
```

위 예제에서 볼 수 있는 바와 같이 gmultipart
api를 활용할 경우 지금까지 Spring 프레임워크에서 사용하던 방법과 같이 Multipart를 활용해 업로드한 파일 정보를 추출하는 것이 가능하다.

```
import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.Id;

import com.google.appengine.api.datastore.Blob;
import com.googlecode.objectify.annotation.Unindexed;

@Entity
public class AttachedFile {
  @Id
  private Long id;
  private ServiceType serviceType;
  private Long contentsId;
  @Unindexed
  private String originalFileName;
  @Unindexed
  private Long size;
  @Unindexed
  private Blob attachment;
  private Date createdDate;

  public AttachedFile() {
  }

  public AttachedFile(ServiceType serviceType, Long contentsId, String
originalFileName,
    Long size, Blob attachment, Date createdDate) {
    this.serviceType = serviceType;
    this.contentsId = contentsId;
    this.originalFileName = originalFileName;
    this.size = size;
    this.attachment = attachment;
    this.createdDate = createdDate;
  }

  [...   ...]
}
```

이와 같이 업로드한 파일 데이터는 DataStore의 Blob Type을 활용해 저장하는 것이 가능하다. 위 AttachedFile은 Objectify 프레임워크를 활용해 GAE의 DataSource에 저장하기 위한 모델 클래스이다.

단 GAE DataStore의 Blob Type은 1MB의 데이터까지만 저장할 수 있다는 한계가 있다. 1MB 이상의 파일을 저장하고 싶다면 BlobStoreService를 활용해야 한다. BlobStoreService를 활용하는 방법은 다음 문서에서 다루도록 하겠다.