Zipping Multiple Files in Salesforce Using Compression.ZipWriter

Overview

Salesforce developers often face scenarios where multiple files must be bundled and shared as a single downloadable package—just like zipping folders on your computer. Thankfully, Salesforce offers a native utility: `Compression.ZipWriter`, which allows you to programmatically create `.zip` files within Apex.

In this blog post, we'll explore:

- Why and when to zip files in Salesforce
- How `Compression.ZipWriter` works
- A practical example using `ContentVersion`
- Tips for large-scale and batch-safe implementations

Why Use Zipping in Apex?

Here are a few common business cases:

- Combining multiple invoice PDFs into a zip for monthly customer reporting
- Packaging student records or course materials for e-learning portals
- Archiving project documents into one downloadable bundle

Salesforce doesn't support folders inside the Files object, so zipping provides users with a clean and organized way to download multiple documents at once.

Meet Compression.ZipWriter

`Compression.ZipWriter` is a built-in Apex class that allows you to:

- Create a `.zip` archive in memory
- Add multiple file entries (with names and binary data)
- Output a final `Blob` that can be used to attach to records or emails

This enables Apex developers to replicate standard file compression behavior inside Salesforce natively.

Example: Zipping Uploaded Files by Project

Let's say users upload files (stored as `ContentVersion` records) to a custom `Project__c` record. We want to gather all related files and zip them into one archive that can be emailed or stored.

Apex Code Example

```
public class ProjectFileZipper {
    public static Blob zipProjectFiles(Id projectId) {
        List<ContentVersion> versions = [
            SELECT Id, Title, PathOnClient, VersionData
            FROM ContentVersion
            WHERE FirstPublishLocationId = :projectId
        ];
        Compression.ZipWriter zipWriter = new Compression.ZipWriter();
        for (ContentVersion cv : versions) {
            String fileName = String.isNotBlank(cv.PathOnClient) ?
cv.PathOnClient : cv.Title;
            zipWriter.addEntry(fileName, cv.VersionData);
        }
        return zipWriter.getArchive();
    }
}
```

Sending the Zip as an Email Attachment

You can use Salesforce's email service to send the generated zip file.

```
Blob zipBlob = ProjectFileZipper.zipProjectFiles(projectId);

Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
email.setToAddresses(new String[] { 'user@example.com' });
email.setSubject('Your Project Files');
email.setPlainTextBody('Attached is the zip archive of your files.');

Messaging.EmailFileAttachment attachment = new
Messaging.EmailFileAttachment();
attachment.setFileName('ProjectDocs.zip');
attachment.setBody(zipBlob);
attachment.setContentType('application/zip');

email.setFileAttachments(new Messaging.EmailFileAttachment[] { attachment
});

Messaging.sendEmail(new Messaging.SingleEmailMessage[] { email });
```

Best Practices

- **Batch-Friendly:** Move zipping logic to the `finish()` method in a batch job to avoid heap size limits.
- **File Size Limit:** Keep attachments under 25 MB to prevent email delivery failures.
- **Error Handling:** Always check for null `VersionData` to avoid malformed entries.
- **File Naming:** Clean file names to avoid unsafe characters like slashes or colons.

Where You Can Use This

- Internal portals and dashboards
- Automated export tools for community or customer users
- Scheduled archiving jobs for compliance reporting
- Button-triggered document bundles in LWC or Aura components

---

Conclusion

With `Compression.ZipWriter`, Salesforce provides a powerful yet simple tool for packaging multiple files into a single archive. Whether you're delivering grouped reports, bundling related content, or enhancing your automation workflows, zipping files with Apex can dramatically improve both usability and performance.