Emmanouil Gkatziouras

Got Code?

Behavioural Design Patterns: Iterator

OCTOBER 29, 2018 ~ EMMANOUIL GKATZIOURAS

The iterator pattern is one of the most used patterns from the behavioural ones. Most of the times you use it without even noticing it. Supposing you have a container with elements and you want to traverse them. Iterating the elements might differ based on the container, retrieval method etc. By using the iterator pattern we can decouple the algorithms used by the containers, and the retrieval methods and abstract the iteration of the elements contained.

For example when it comes to retrieving paged data from a rest endpoint the iterator can help you to abstract it. Therefore you won't expose the user with any information on how you retrieve the next batch of data.

I will make a new iterator interface similar the one provided by the java language.

```
1  package com.gkatzioura.design.behavioural.iterator;
2
3  public interface Iterator<T> {
4
5      boolean hasNext();
6
7      T next();
8
9  }
```

We will use the github jobs (https://jobs.github.com/api) api since it is open to issue rest queries and search for jobs.
The first page is page zero.

https://jobs.github.com/positions.json?page=0 (https://jobs.github.com/positions.json?page=0).

We shall create a simple object which would contain the id, title and company information.

```java
package com.gkatzioura.design.behavioural.iterator;

import java.util.UUID;

public class GitHubJob {

    private final UUID id;
    private final String company;
    private final String title;

    public GitHubJob(UUID id, String company, String title) {
        this.id = id;
        this.company = company;
        this.title = title;
    }

    public UUID getId() {
        return id;
    }

    public String getCompany() {
        return company;
    }

    public String getTitle() {
        return title;
    }
}
```

Our first step would be to create a repository witch shall fetch the data of the page specified.

```java
package com.gkatzioura.design.behavioural.iterator;

import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

import org.apache.commons.io.IOUtils;
import org.json.JSONArray;
import org.json.JSONObject;

public class GitHubJobsRepository {

    public static final String GITHUB_JOB_API = "https://jobs.github

    public List<GitHubJob> fetch(int page) throws Exception {

        List<GitHubJob> gitHubJobs = new ArrayList<>();

        URL url = new URL(GITHUB_JOB_API+page);
        HttpURLConnection httpConnection = (HttpURLConnection) url.o
        String response = IOUtils.toString(httpConnection.getInputSt
        JSONArray jsonArray = new JSONArray(response);

        for(int i=0;i<jsonArray.length();i++) {
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            GitHubJob gitHubJob = new GitHubJob(
                    UUID.fromString(jsonObject.getString("id")),
                    jsonObject.getString("company"),
                    jsonObject.getString("title"));
            gitHubJobs.add(gitHubJob);
        }

        return gitHubJobs;
    }

}
```

What’s great with this api is that if you ask for a page that does not exists you get an empty json object, thus asking for a non existing page won’t give us an exception like 404, therefore no need for error handling for this case.

Now the interesting part is the iterator.

```java
package com.gkatzioura.design.behavioural.iterator;

import java.util.ArrayList;
import java.util.List;

public class GitHubIterator implements Iterator<GitHubJob> {

    private List<GitHubJob> currentJobsPage = new ArrayList<>();
    private int page = 0;

    private final GitHubJobsRepository gitHubJobsRepository;

    public GitHubIterator(GitHubJobsRepository gitHubJobsRepository) {
        this.gitHubJobsRepository = gitHubJobsRepository;
    }

    @Override
    public boolean hasNext() {
        fetchPageIfNeeded();
        return currentJobsPage.size()>0;
    }

    @Override
    public GitHubJob next() {
        fetchPageIfNeeded();

        if(currentJobsPage.size()==0) {
            return null;
        }

        return currentJobsPage.remove(0);
    }

    private void fetchPageIfNeeded() {
        if(page == -1) {
            return;
        }

        if(currentJobsPage==null||currentJobsPage.size()==0) {
            try {
                currentJobsPage = gitHubJobsRepository.fetch(page);
                if(currentJobsPage.size()==0) {
                    page = -1;
                } else {
                    page++;
                }
            } catch (Exception e) {
                throw new RuntimeException();
            }
        }
    }
}
```

The iterator shall contain a page request from github. For each element requested one item shall be removed from the page. Once the page gets empty a new page shall be requested.

Whether the user asks for another element or if another element exists the code shall check if the current page is empty and if yes it will request for the next page. If we got an empty page this mean that there should not be any extra request to the iterator.

The next step would be to add a method to the repository which shall give back the github iterator.

```
1  public class GitHubJobsRepository {
2
3  ...
4
5      public Iterator<GitHubJob> iterator() {
6          return new GitHubIterator(this);
7      }
8
9  ...
10 }
```

So let us sum up and iterate over all the entries.

```
1  package com.gkatzioura.design.behavioural.iterator;
2
3  public class IteratorExample {
4
5      public static void main(String[] args) throws Exception {
6          GitHubJobsRepository gitHubJobsRepository = new GitHubJobsRe
7
8          Iterator<GitHubJob> gitHubJobIterator = gitHubJobsRepository
9
10         while (gitHubJobIterator.hasNext()) {
11             GitHubJob gitHubJob = gitHubJobIterator.next();
12             System.out.println(String.format(" id: %s title: %s compa
13         }
14     }
15
16 }
```

You can find the source code on github (https://github.com/gkatzioura/DesignPatterns/tree/master/src/main/java/com/gkatzioura/design/behavioural/iterator).

It is also worth referencing patterns we saw previously such as the interpreter (https://egkatzioura.com/2018/10/23/behavioural-design-patterns-interpreter/), chain of responsibility (https://egkatzioura.com/2018/10/08/behavioural-design-patterns-chain-of-responsibility/) and the command (https://egkatzioura.com/2018/10/15/behavioural-design-patterns-command/) pattern.

POSTED IN DESIGN PATTERNS, JAVA, SOFTWARE ENGINEERING

Published by Emmanouil Gkatziouras

View all posts by Emmanouil Gkatziouras

One thought on “Behavioural Design Patterns: Iterator”

1. Pingback: Behavioural Design Patterns: Mediator – Emmanouil Gkatziouras