In JSR 315: Java Servlet 3.0 Specification, `web-fragment.xml` is introduced for pluggability of library jars which are packaged under `WEB-INF/lib`. The content of `web.xml` and `web-fragment.xml` are almost the same. One can define servlets, filters and listeners there. One can also specify `metadata-complete=true` in a given `web-fragment.xml`. In the latter case, the annotation processing of classes in that jar would be skipped. With `web-fragment.xml`, library jars can be self-contained and provide web related metadata information.

The basic differences of `web.xml` and `web-fragment.xml` are summarized in the following table:

	web.xml	**web-fragment.xml**
Location	`WEB-INF` of the war file	`META-INF` directory of JAR file inside WAR file's `WEB-INF/lib`
Ordering related element	`<absolute-ordering>`	`<ordering>`

Ordering of web fragments

If there are more than one web-fragment jars, then one may like to specify the order of processing web-fragment.xml and annotations. This is important. For instance, filters will be executed in the order specified in `web.xml`. Similary for listeners. In Servlet 3.0, `<absolute-ordering>` is introduced in `web.xml` and `<ordering>` is introduced in `web-fragment.xml`. The ordering of web-fragments is specified in the following priority:

- from `<absolute-ordering>` in `web.xml` if it exists
- from `<ordering>` for each `web-fragment.xml` if it exists
- otherwise unspecified

absolute-ordering in web.xml

The `<absolute-ordering>` in `web.xml` provides a way to specify the ordering of loading `web-fragment.xml` and annotation processing of web fragment. For instance,

```
<web-app>
    ...
    <absolute-ordering>
        <name>A</name>
         <others/>
        <name>B</name>
    <absolute-ordering>
</web-app>
```

In the above example, the web fragment A would be processed first and web fragment B would be processed last. Note the name A and B are specified in `name` element of `web-fragment.xml` (see examples below).

ordering in web-fragment.xml

If there is no `<absolute-ordering>` in `web.xml`, then one would look at `<ordering>` in `web-fragment.xml`. The details are described in section 8.2.3 of Servlet 3.0 spec. Let us look at some examples.

- There is only one jar having `<ordering>` in `web-fragment.xml`.

```
    <web-fragment>
        <name>A</name>
        ...
        <ordering>
            <before>
                <others/>
            </before>
    </ordering>
</web-fragment>
```

In this case, web-fragment A would be processed first.

- There are two jars having `<ordering>` in `web-fragment.xml`, namely
  web-fragment A:

  ```
  <web-fragment>
      <name>A</name>
      ...
      <ordering>
          <before>
              <others/>
          </before>
      </ordering>
  </web-fragment>
  ```

  web-fragment B:

  ```
        <web-fragment>
            <name>B</name>
            ...
            <ordering>
                <before>
                    <others/>
                </before>
            </ordering>
        </web-fragment>
  ```

  Both web-fragment A and B would like to be processed first. In this case, one only guarantee that both A and B are processed before other web-fragments. But the ordering of A and B are not determined, that is arbitrary in this case.

- There are two jars having `<ordering>` in `web-fragment.xml`, namely
  web-fragment A:

  ```
  <web-fragment>
      <name>A</name>
      ...
      <ordering>
          <before>
              <others/>
          </before>
      </ordering>
  </web-fragment>
  ```

  web-fragment B:

  ```
        <web-fragment>
            <name>B</name>
  ```

```
    ...
    <ordering>
        <after>
            <name>A</name>
        </after>
        <before>
            <others/>
        </before>
    </ordering>
</web-fragment>
```

In this case, A would be processed first, then followed by B, and then other web-fragments.

If one would like to have a deterministic ordering, then I would recommend to use `absolute-ordering` in `web.xml`.