Resources

(Notes)

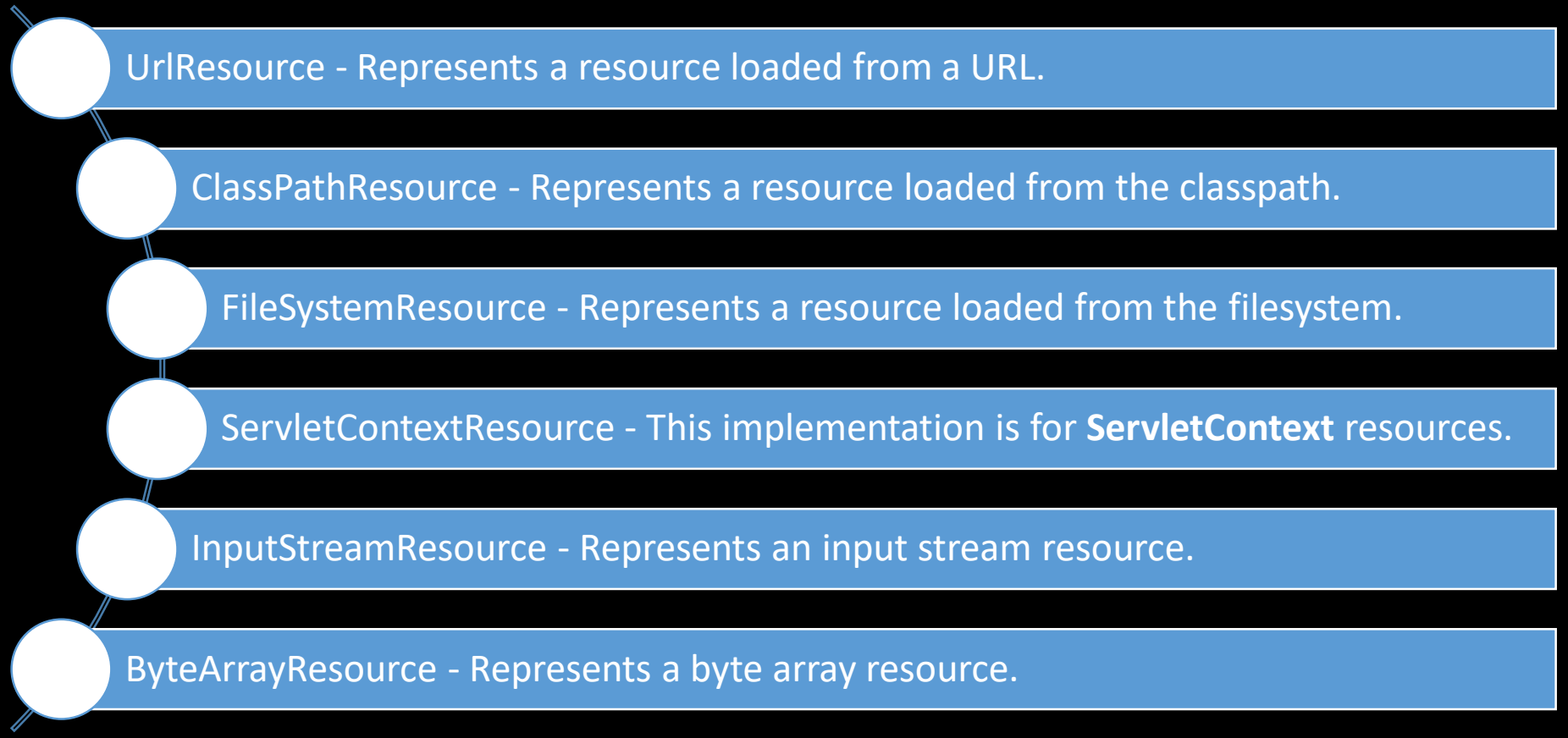
Resource interface

```
public interface Resource extends InputStreamSource {
    boolean exists();
    boolean isOpen();
    URL getURL() throws IOException;
    File getFile() throws IOException;
    Resource createRelative(String relativePath) throws IOException;
    String getFilename();
    String getDescription();
}
```

- *getInputStream()* (From **InputStreamSource** interface): Locates and opens the resource. It returns an **InputStream** for reading from the resource.
- *exists():* Returns a boolean indicating whether this resource actually exists in physical form.
- *isOpen():* Returns a boolean indicating whether this resource represents a handle with an open stream. If true, the

- **InputStream** must be read once only and then closed to avoid resource leaks. It will typically be false for resource implementations, with the exception of **InputStreamResource**.
- *getDescription():* Returns a description for this resource. The description can be used for error output when working with the resource. The description is often the fully qualified file name or the actual URL of the resource.

Resource implementations



Resource prefixes

PREFIX	EXAMPLE	EXPLANATION
classpath:	classpath:com/myapp/config.xml	Loaded from the classpath.
file:	file:///data/config.xml	Loaded as a URL from the filesystem.
http:	https://myserver/logo.png	Loaded as a URL.
(none)	/data/config.xml	Depends on the underlying ApplicationContext.

Using ResourceLoader to Get a Resource

```
@Component("resourceLoaderService")
public class ResourceLoaderService implements ResourceLoaderAware{
 private ResourceLoader resourceLoader;
 public void setResourceLoader(ResourceLoader resourceLoader) {
  this.resourceLoader = resourceLoader;
 }
 public void showResourceDataUsingFilePath() throws IOException {
  Resource resource = resourceLoader.getResource("file:d:/test.txt");
  InputStream in = resource.getInputStream();
  BufferedReader reader = new BufferedReader(new InputStreamReader(in));
  while (true) {
   String line = reader.readLine();
   if (line == null)
    break;
   System.out.println(line);
  }
  reader.close();
 }
}
```

Define the ResourceLoaderService class

Write the main method and using Spring application context get the *ResourceLoaderService* object and call the *showResourceDataUsingFilePath()* using this object

```
@SpringBootApplication
public class Application {
  @SuppressWarnings("resource")
  public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
    ApplicationContext ctx =
new AnnotationConfigApplicationContext("com.example.resource
loaderdemo.service");
    ResourceLoaderService loader =
(ResourceLoaderService)ctx.getBean("resourceLoaderService");
   try {
      loader.showResourceDataUsingFilePath();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```