Learning Spring Boot

Greg L. Turnquist

Section 3

Creating a File Upload Controller

In this Section, we are going to take a look at…

- Creating a service to store images
- Creating a Spring MVC controller to handle incoming requests
- Uploading an image using cURL

Learning Spring Boot

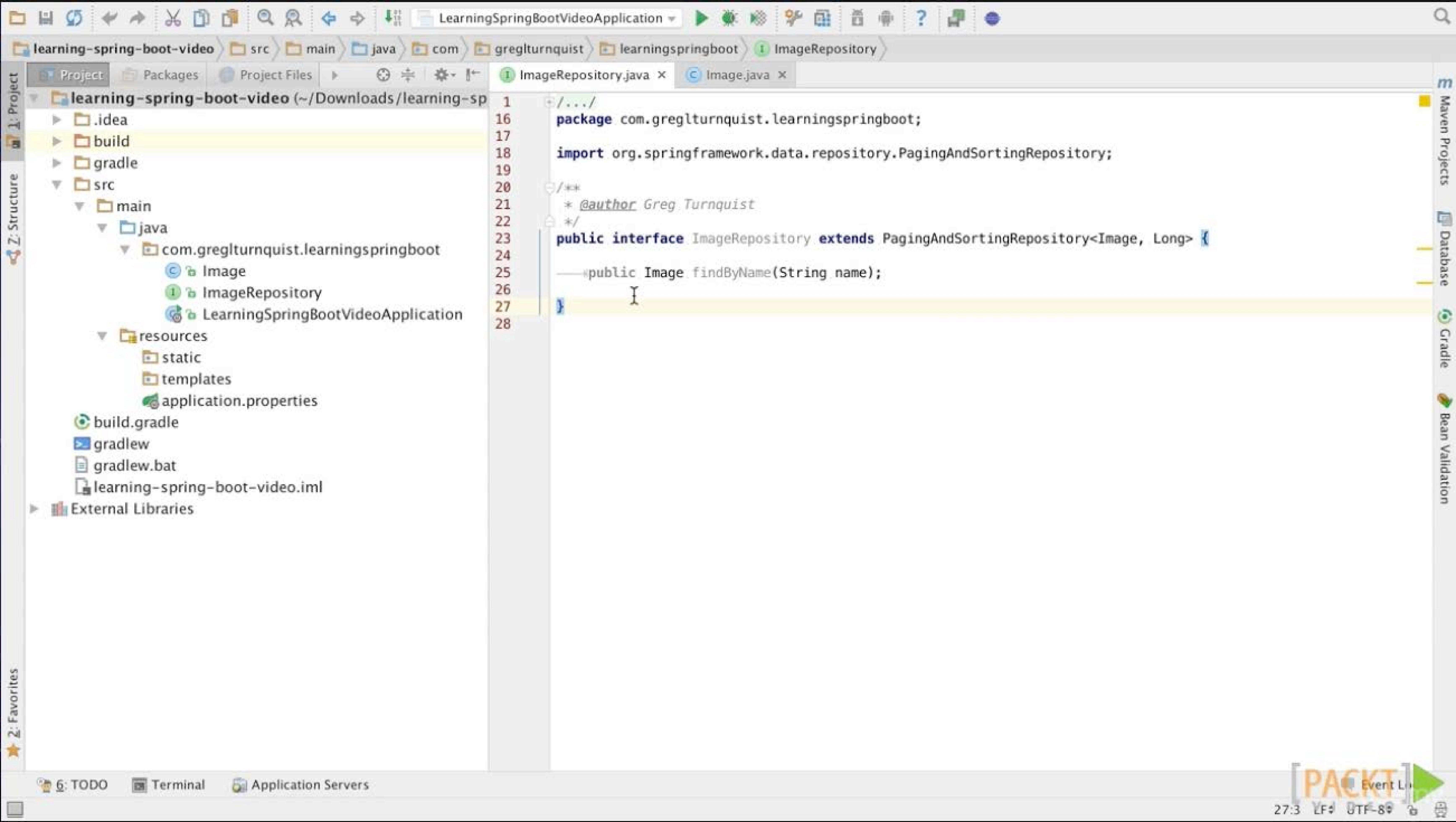
Greg L. Turnquist

Video 3.1

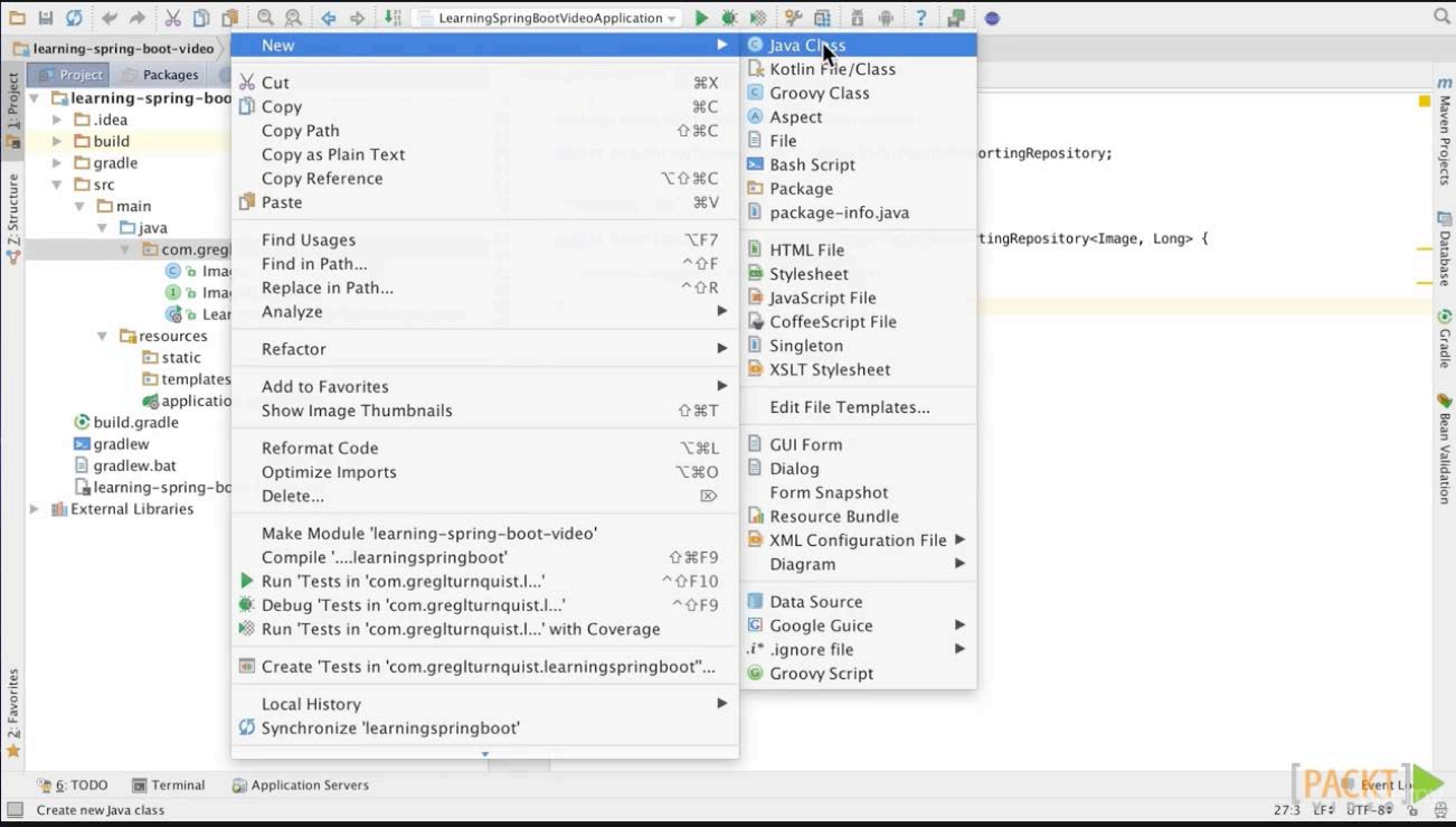
Creating a Service to Store Images

In this Video, we are going to take a look at…

- Creating an auto-wired Spring service
- Adding methods to lookup, store, and delete images
- Pre-load some test data

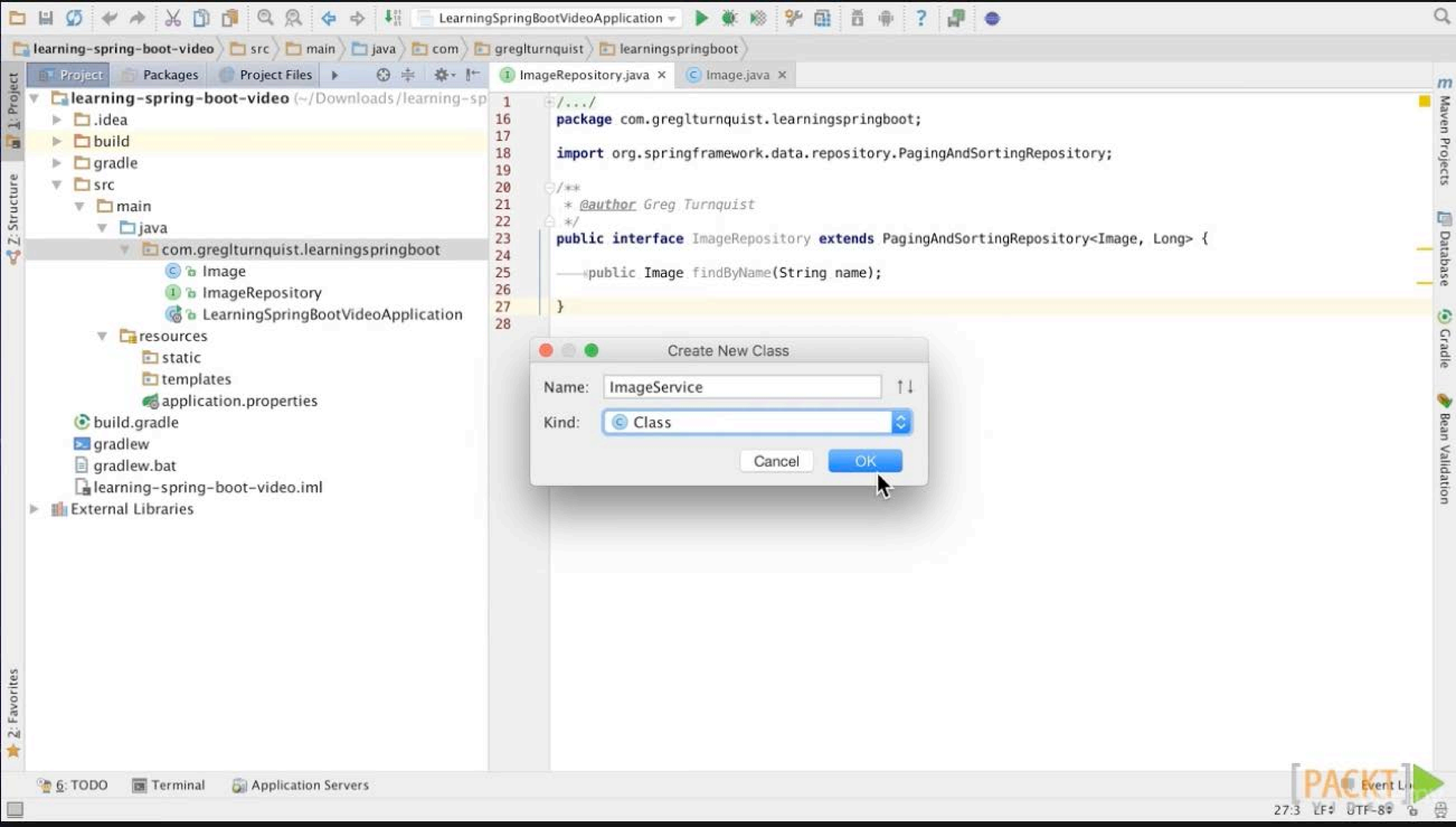


```java
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.data.repository.PagingAndSortingRepository;

/**
 * @author Greg Turnquist
 */
public interface ImageRepository extends PagingAndSortingRepository<Image, Long> {

    public Image findByName(String name);

}
```



```
ortingRepository;

tingRepository<Image, Long> {
```

Create new Java class

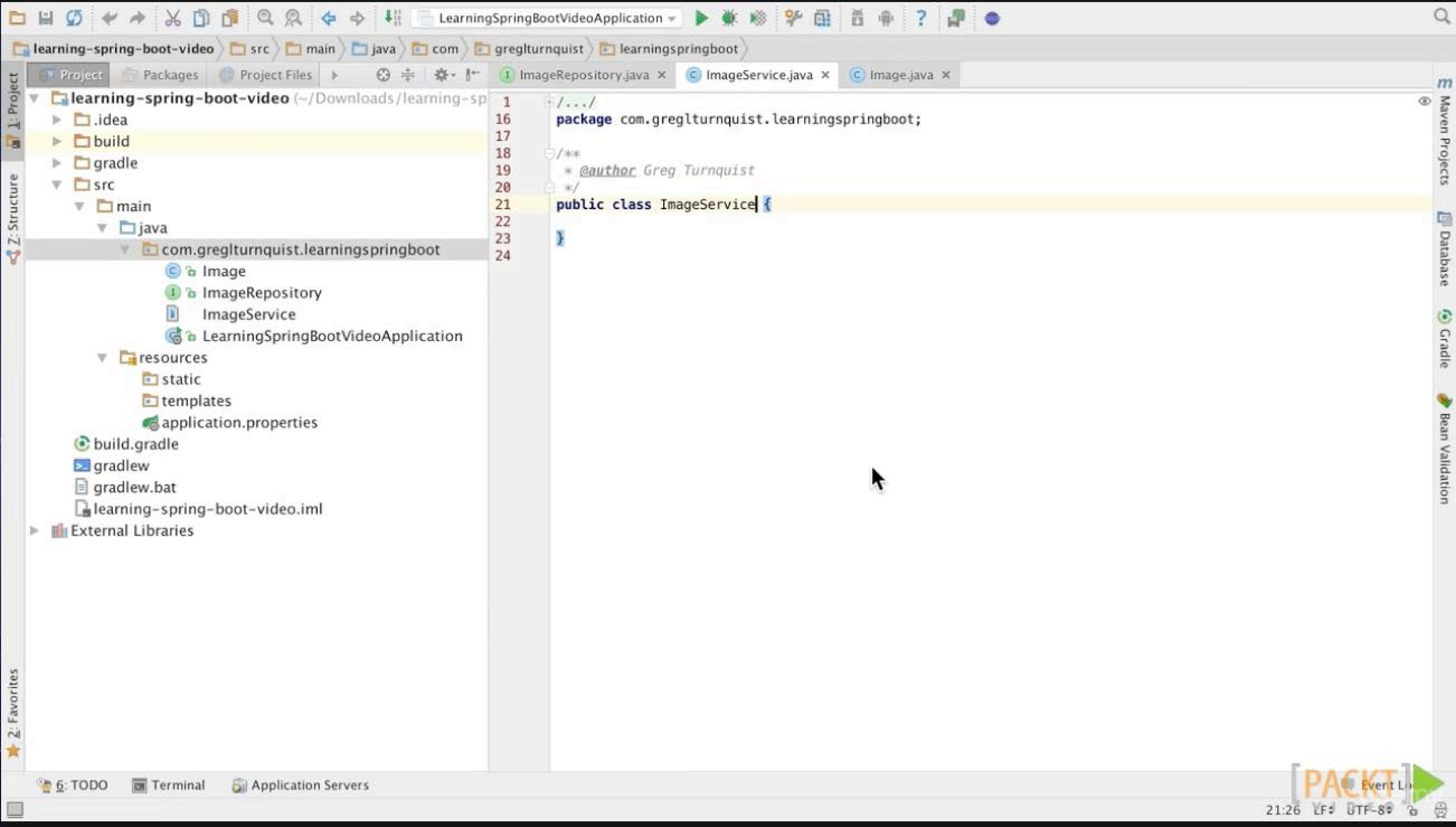


```java
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.data.repository.PagingAndSortingRepository;

/**
 * @author Greg Turnquist
 */
public interface ImageRepository extends PagingAndSortingRepository<Image, Long> {

    public Image findByName(String name);

}
```

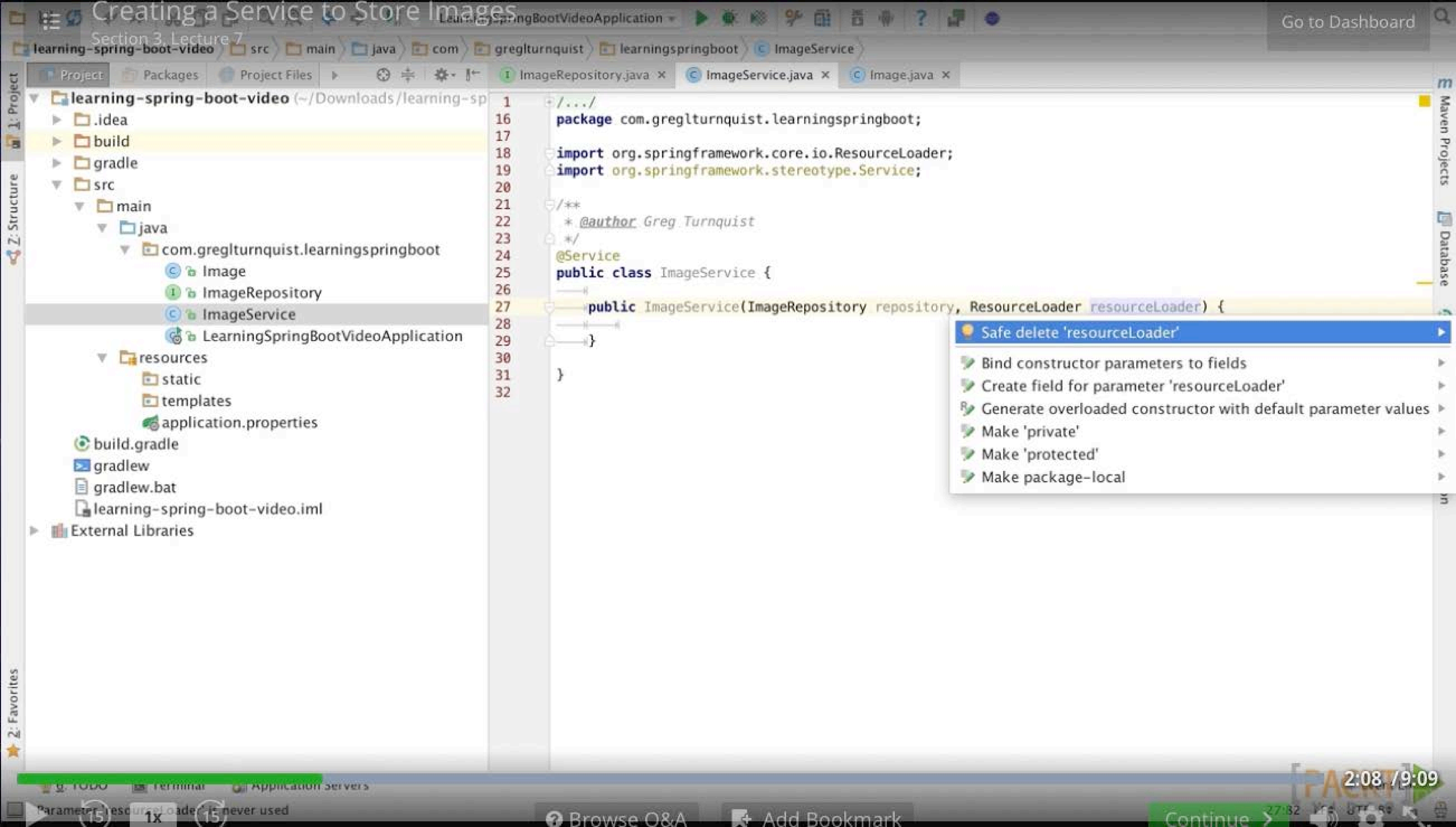
Create New Class

Name: ImageService

Kind: Class



```
/.../
package com.greglturnquist.learningspringboot;

/**
 * @author Greg Turnquist
 */
public class ImageService {

}
```

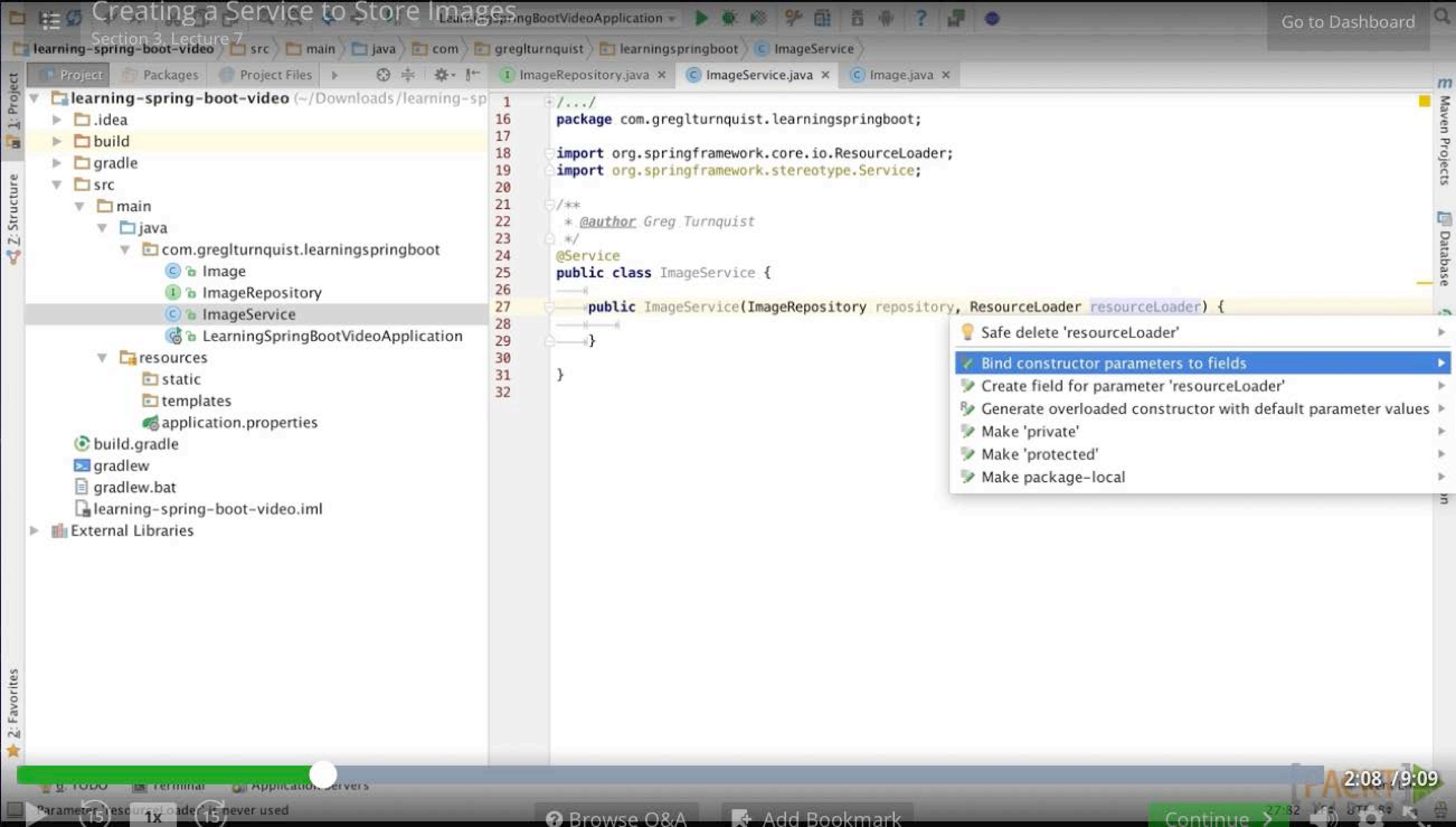


Creating a Service to Store Images

Section 3, Lecture 7

```
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

    }

}
```

Parameter 'resourceLoader' is never used

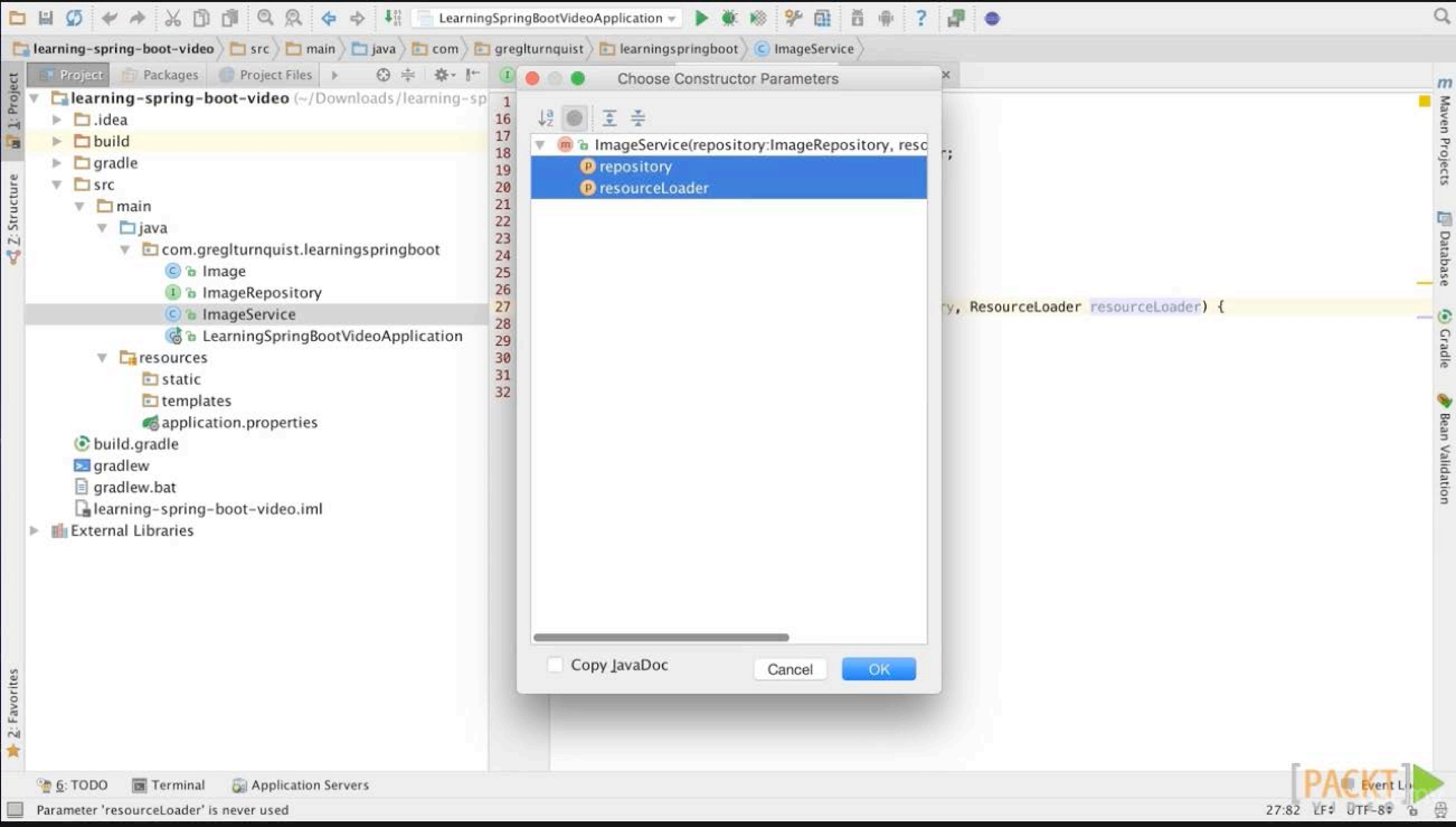


Creating a Service to Store Images

Section 3, Lecture 7

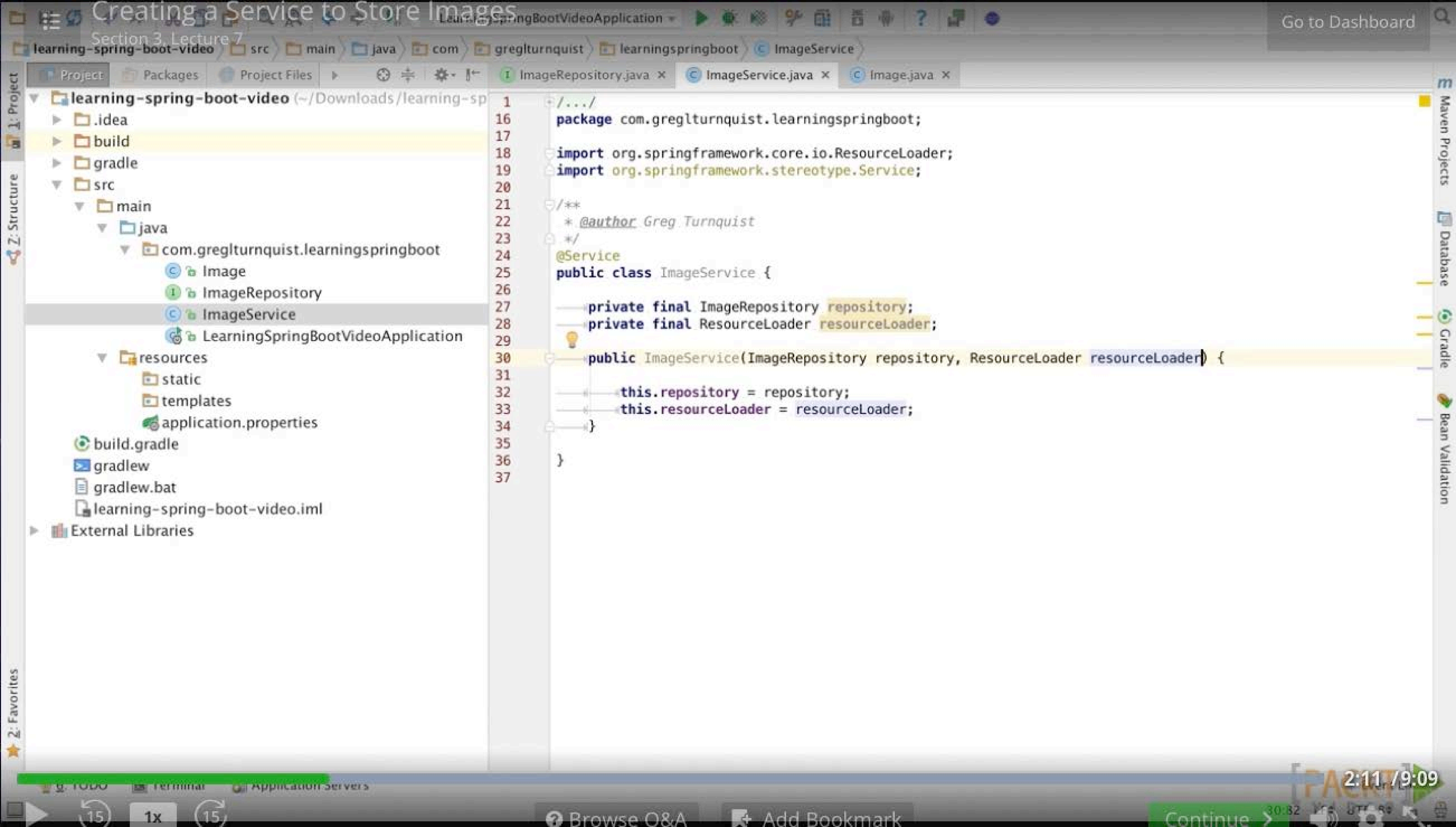
```
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

    }

}
```

Parameter 'resourceLoader' is never used

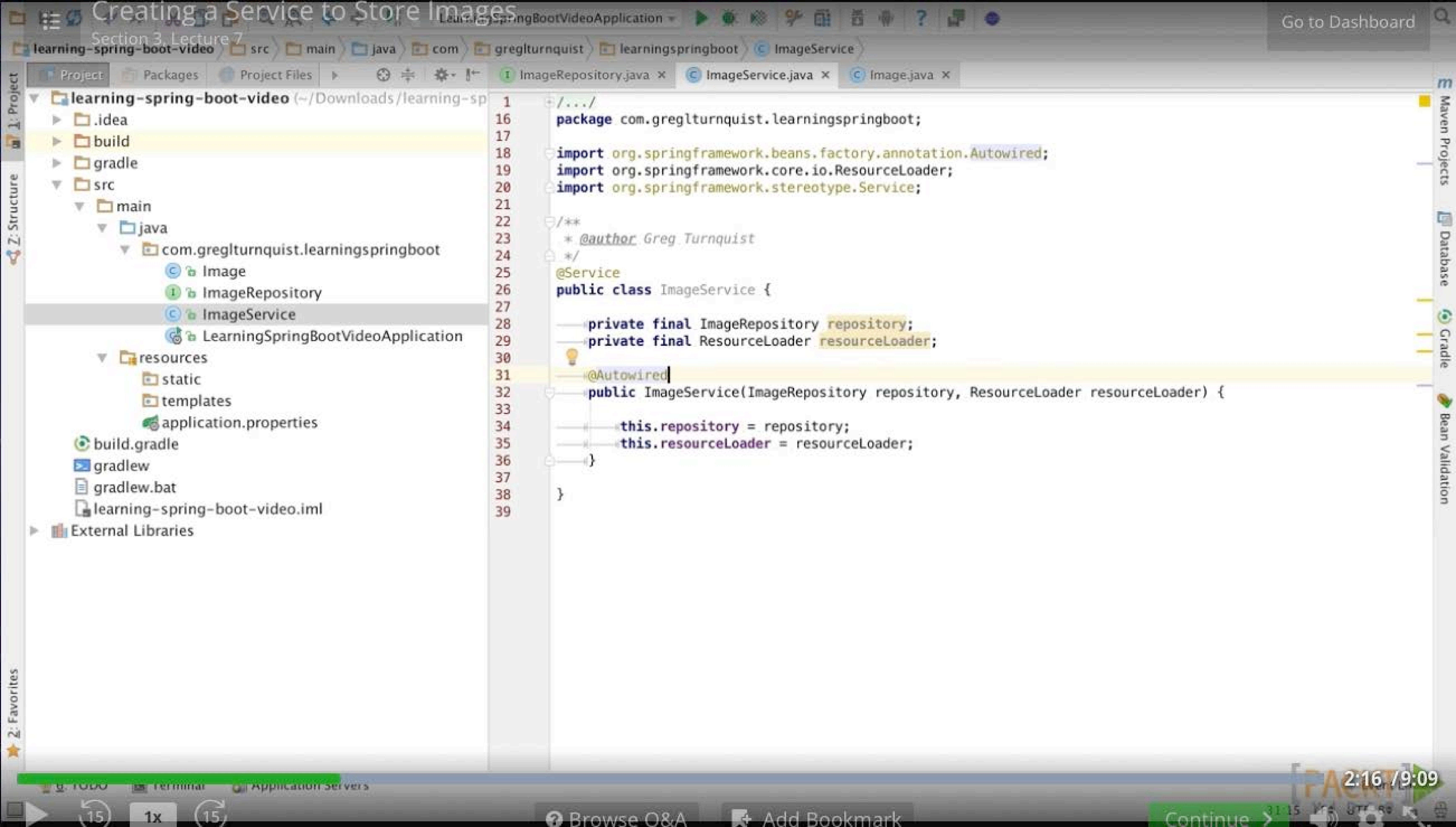


```
;
ry, ResourceLoader resourceLoader) {
```

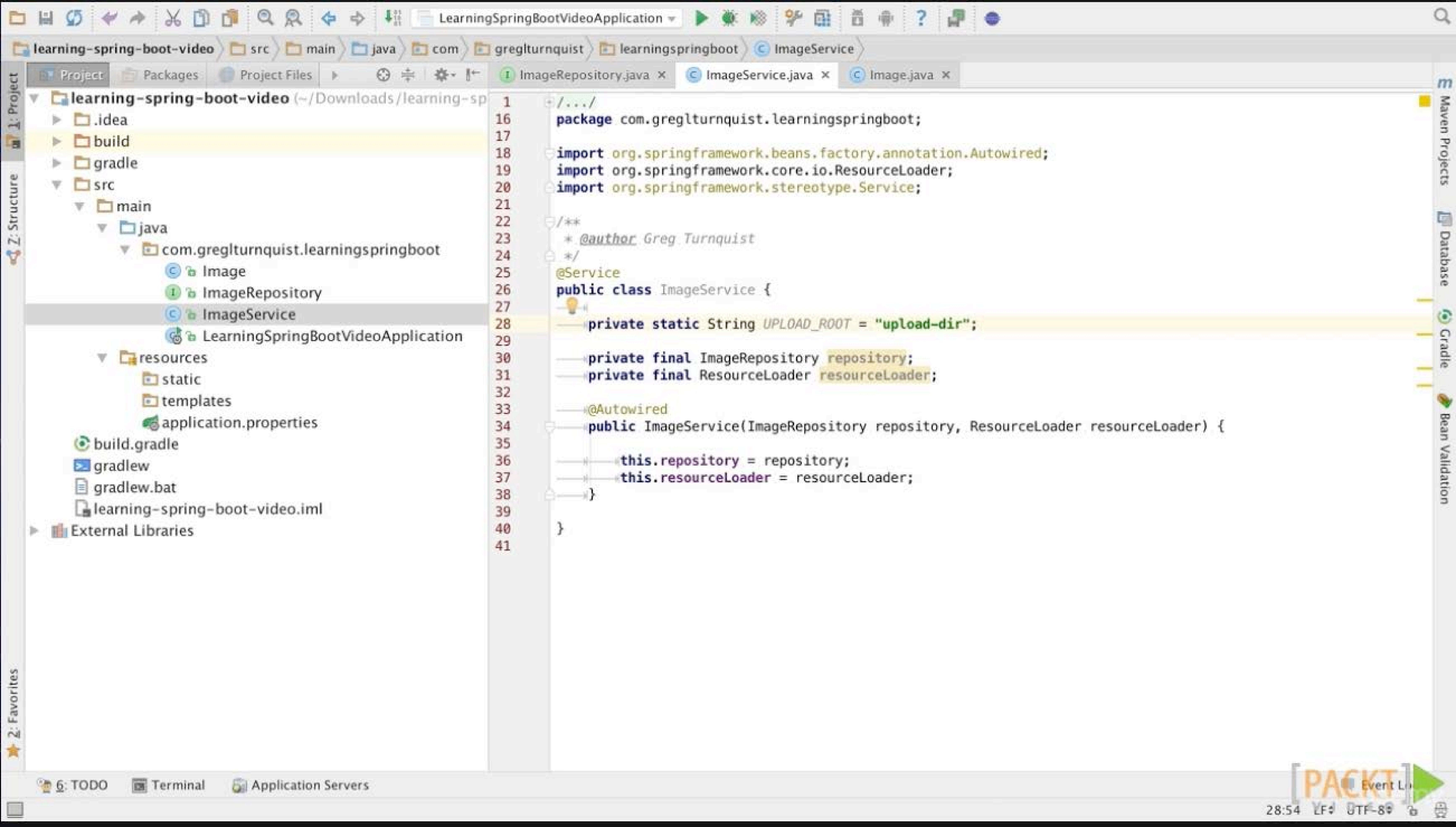
Parameter 'resourceLoader' is never used



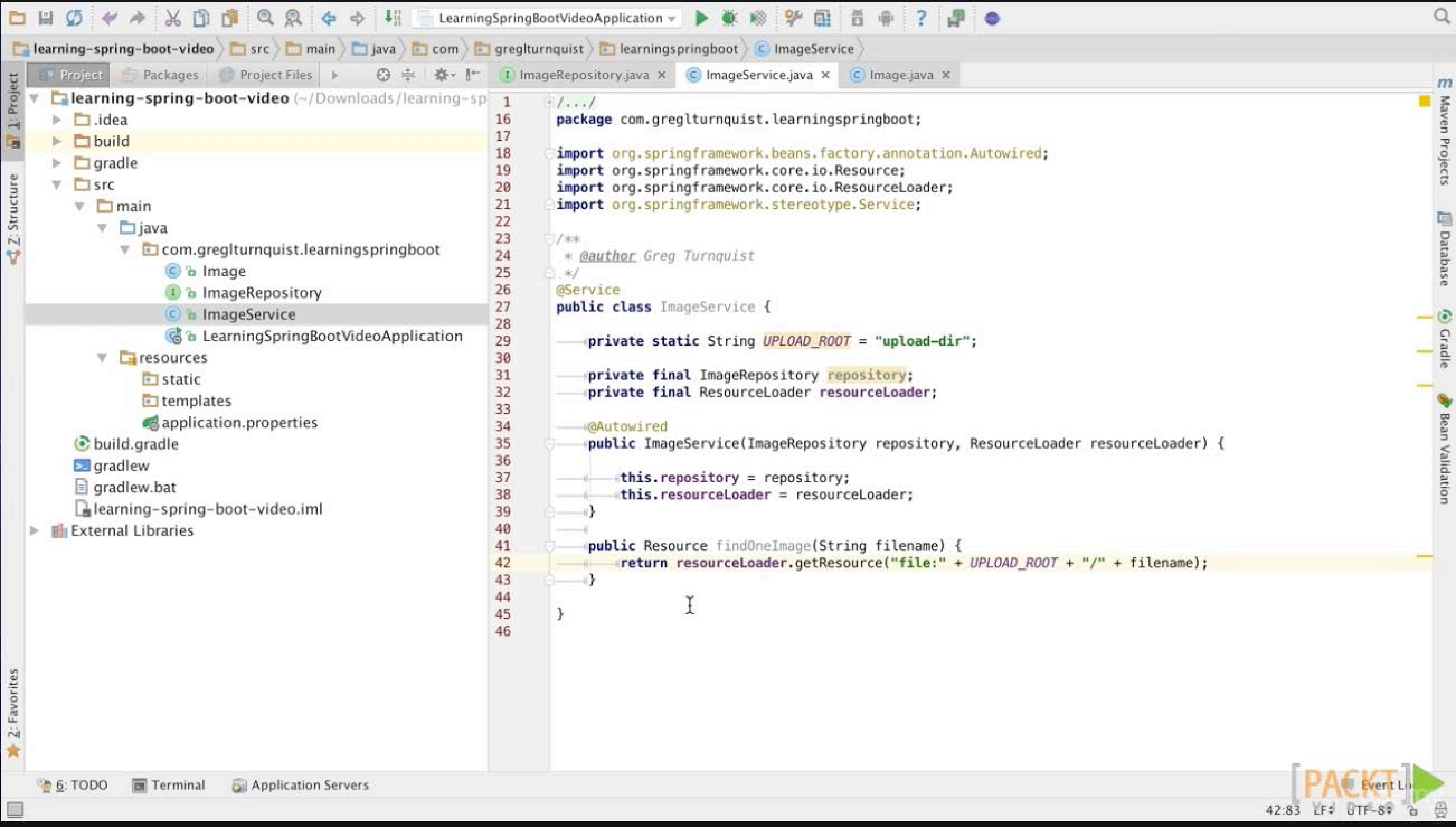
```java
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

}
```



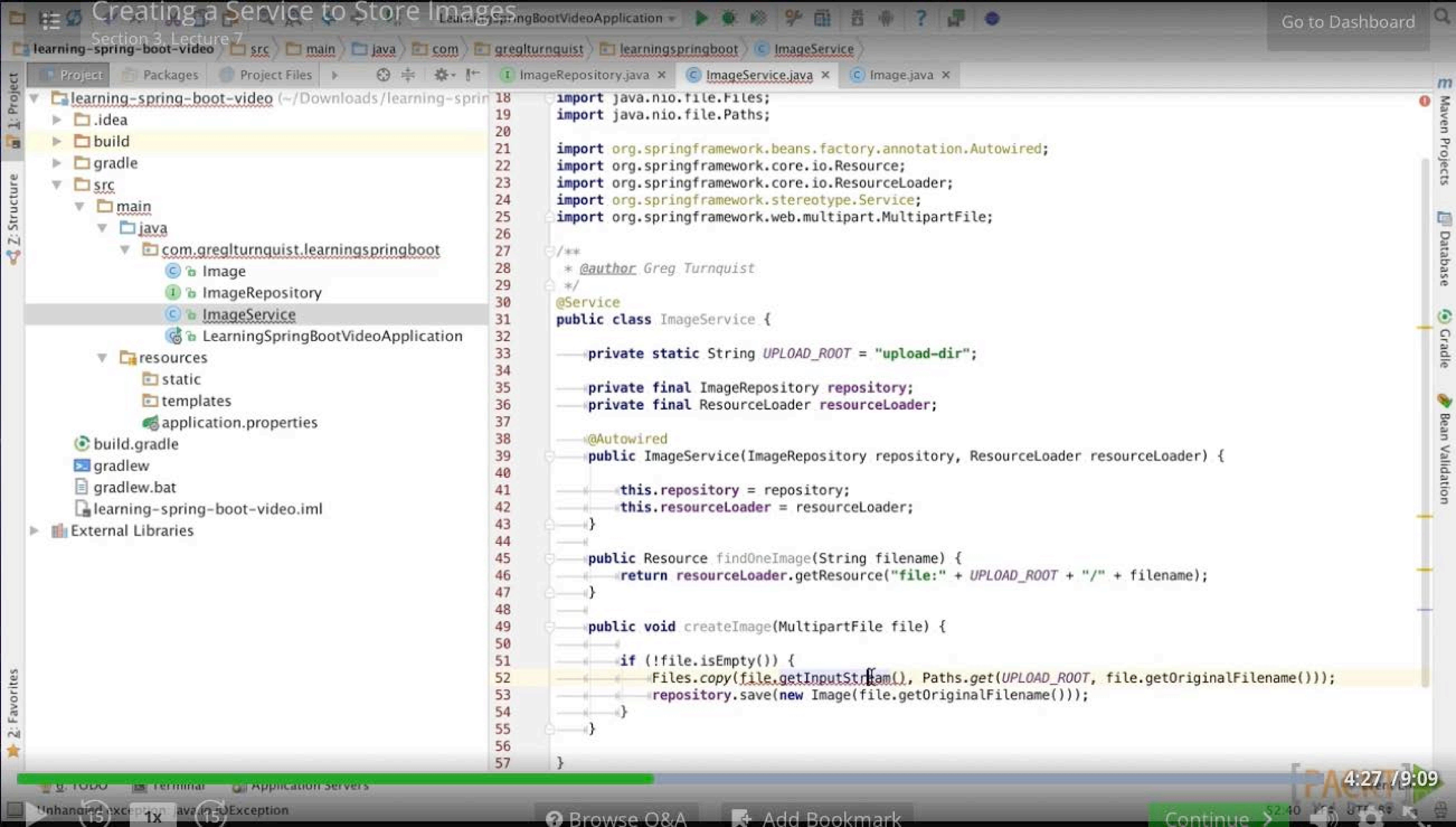
```java
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

}
```



```java
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

}
```



```java
/.../
package com.greglturnquist.learningspringboot;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

}
```

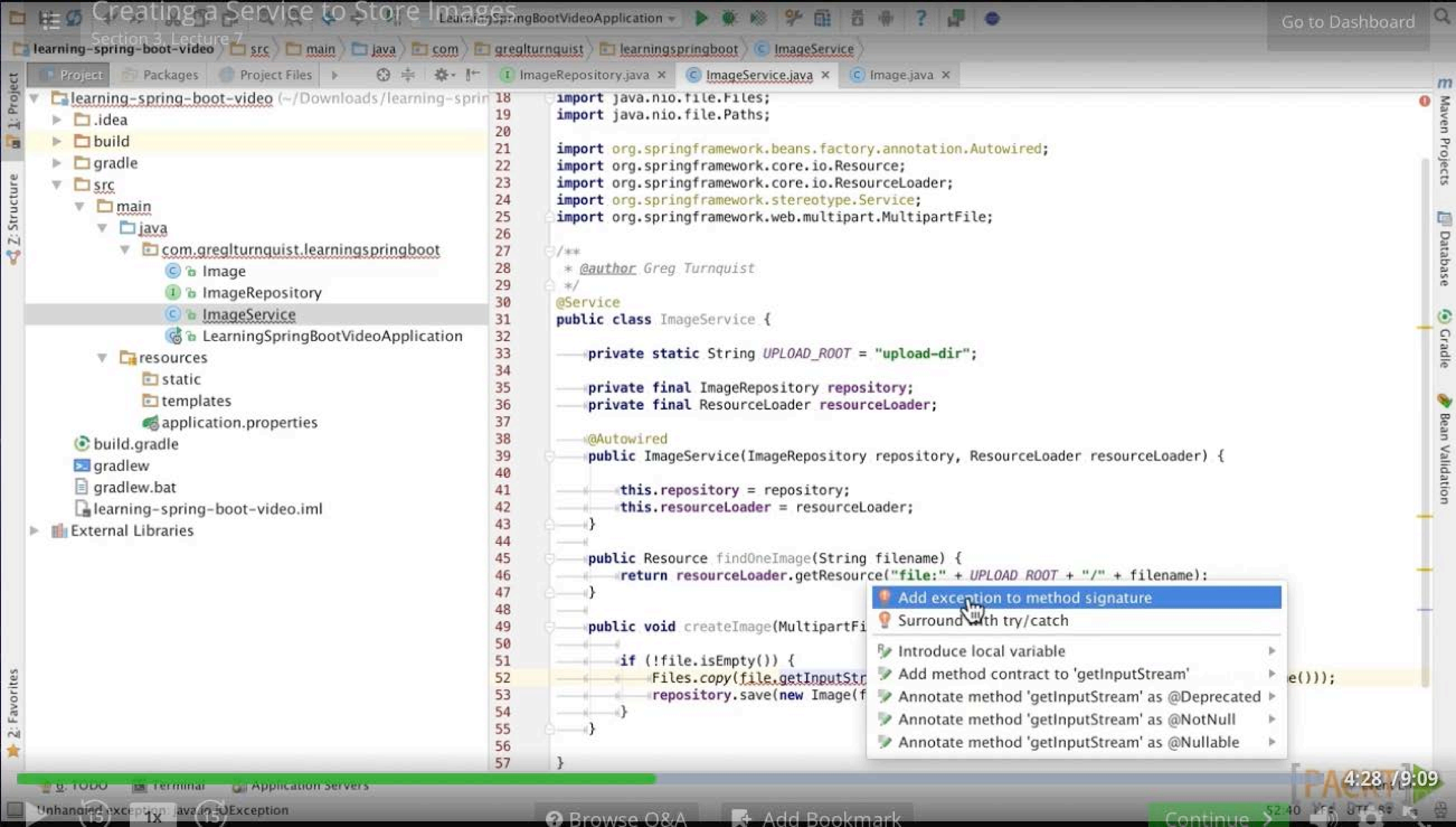


Creating a Service to Store Images

Section 3, Lecture 7

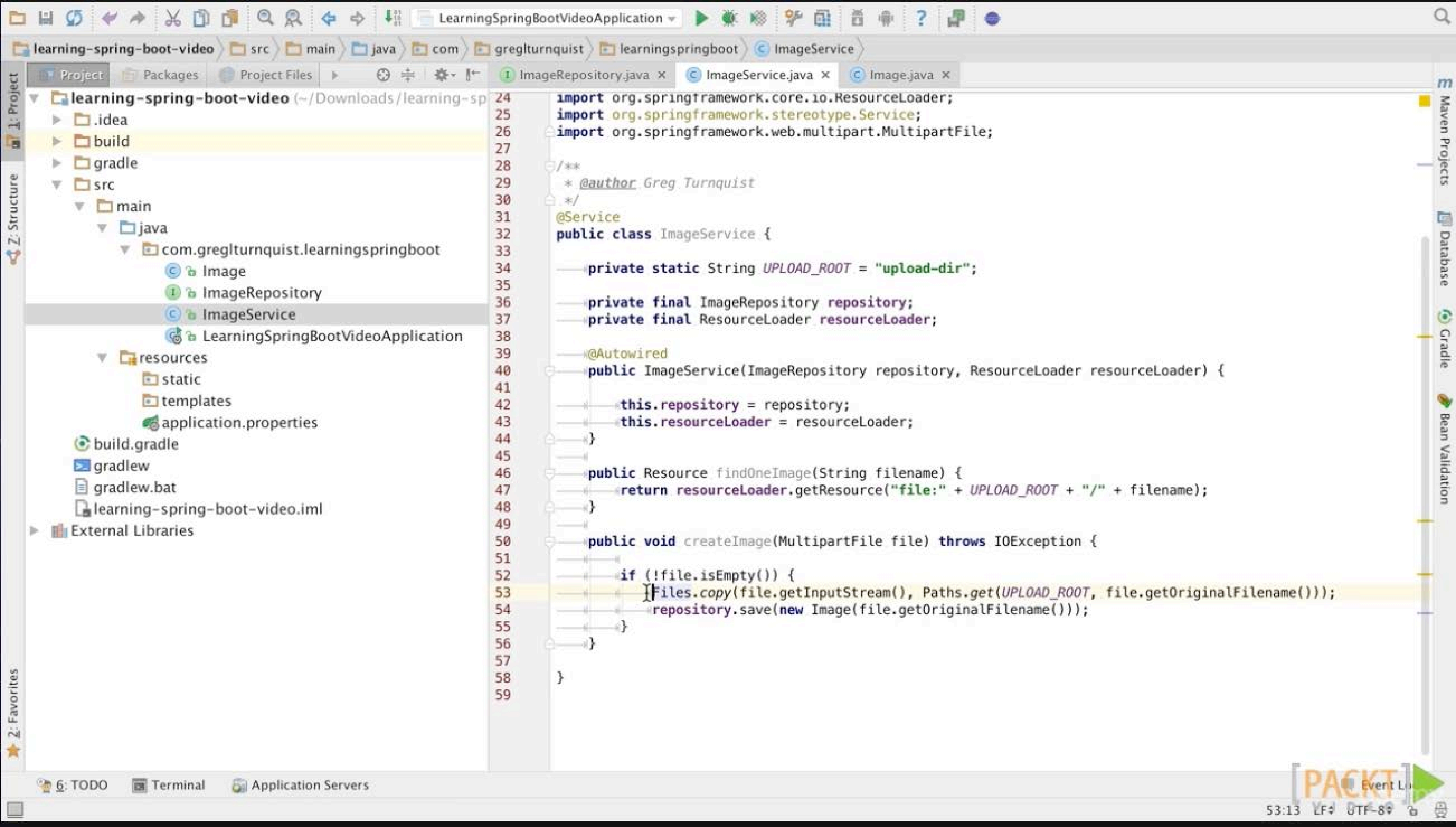
```java
import java.nio.file.Files;
import java.nio.file.Paths;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

    public void createImage(MultipartFile file) {

        if (!file.isEmpty()) {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_ROOT, file.getOriginalFilename()));
            repository.save(new Image(file.getOriginalFilename()));
        }
    }

}
```

Unhandled exception: java.io.IOException

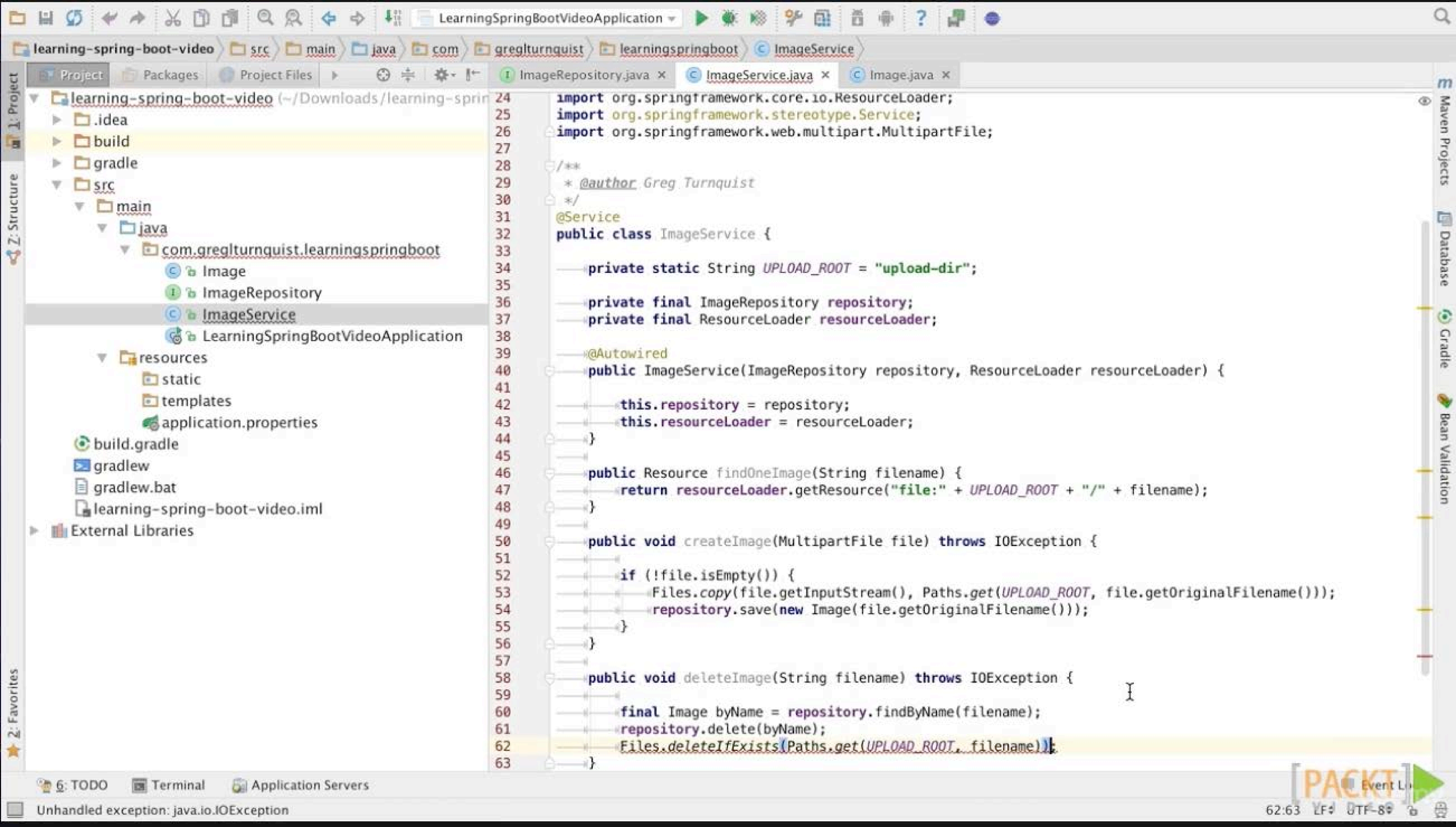


```java
import java.nio.file.Files;
import java.nio.file.Paths;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

    public void createImage(MultipartFi

        if (!file.isEmpty()) {
            Files.copy(file.getInputStr                                                e()));
            repository.save(new Image(f
        }
    }

}
```

Unhandled exception: java.io.IOException

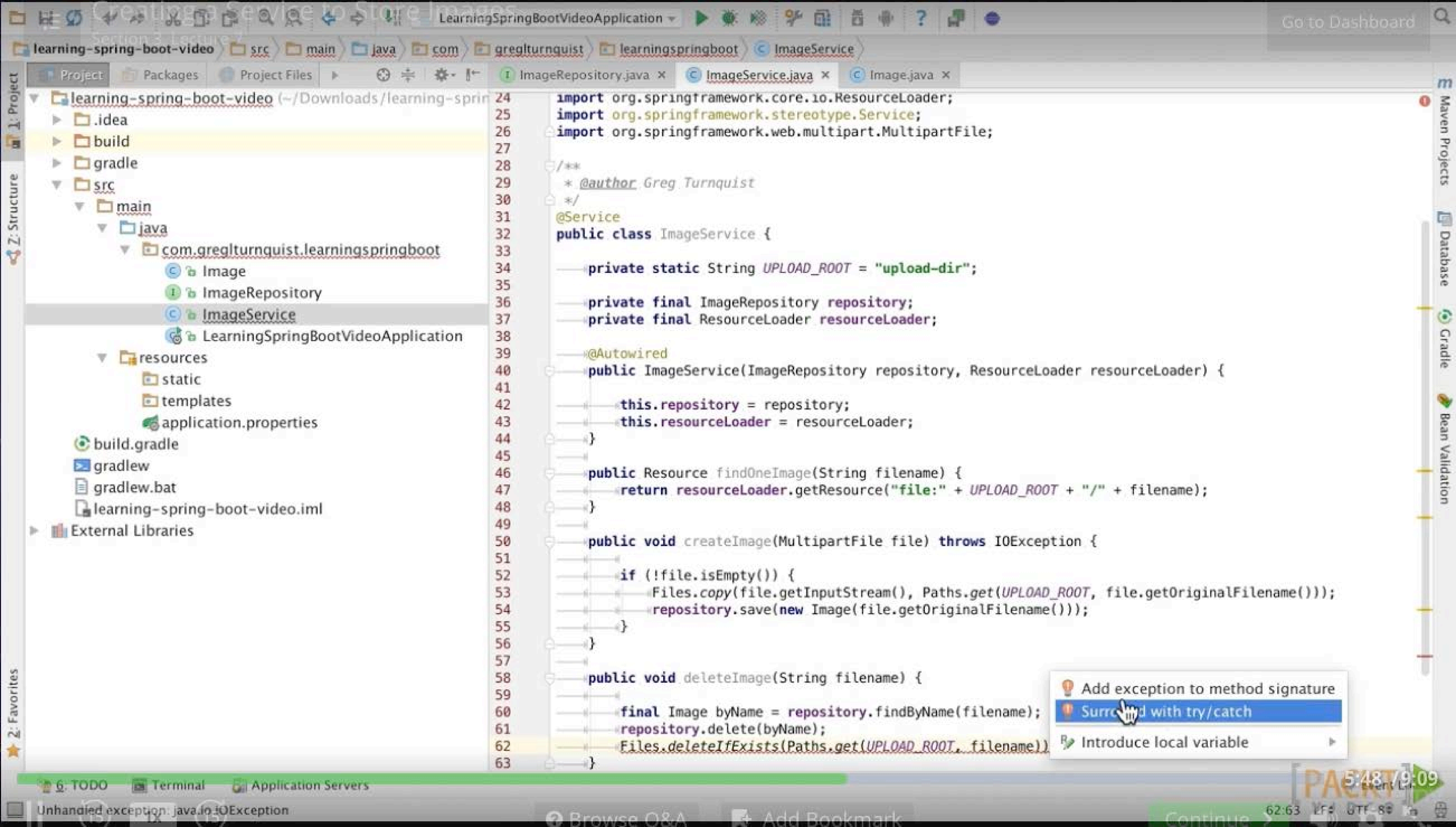


```java
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

    public void createImage(MultipartFile file) throws IOException {

        if (!file.isEmpty()) {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_ROOT, file.getOriginalFilename()));
            repository.save(new Image(file.getOriginalFilename()));
        }
    }

}
```



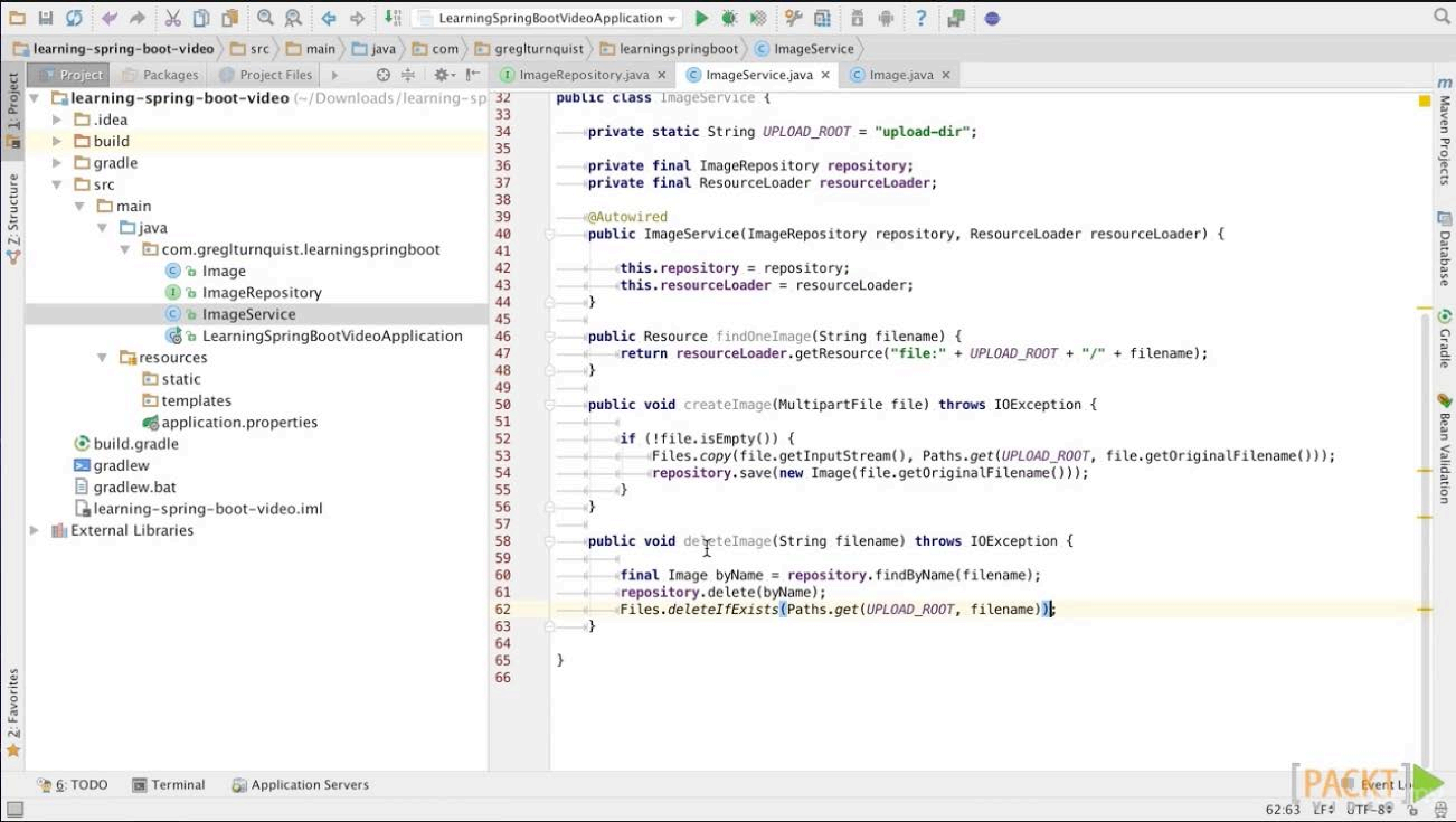
```java
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

    public void createImage(MultipartFile file) throws IOException {

        if (!file.isEmpty()) {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_ROOT, file.getOriginalFilename()));
            repository.save(new Image(file.getOriginalFilename()));
        }
    }

    public void deleteImage(String filename) throws IOException {

        final Image byName = repository.findByName(filename);
        repository.delete(byName);
        Files.deleteIfExists(Paths.get(UPLOAD_ROOT, filename));
    }
```

Unhandled exception: java.io.IOException

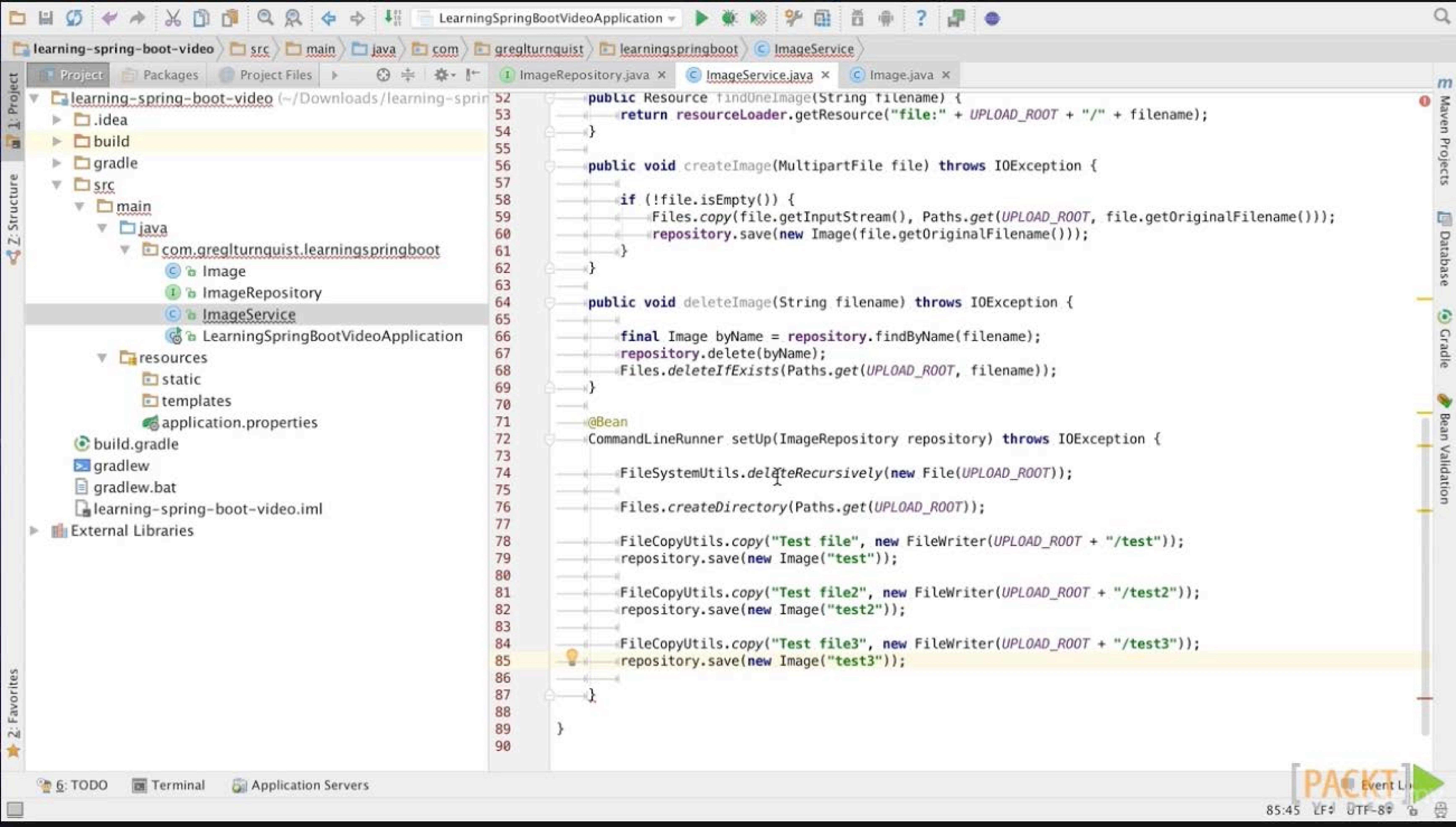


```java
import org.springframework.core.io.ResourceLoader;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

/**
 * @author Greg Turnquist
 */
@Service
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

    public void createImage(MultipartFile file) throws IOException {

        if (!file.isEmpty()) {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_ROOT, file.getOriginalFilename()));
            repository.save(new Image(file.getOriginalFilename()));
        }
    }

    public void deleteImage(String filename) {

        final Image byName = repository.findByName(filename);
        repository.delete(byName);
        Files.deleteIfExists(Paths.get(UPLOAD_ROOT, filename))
    }
```

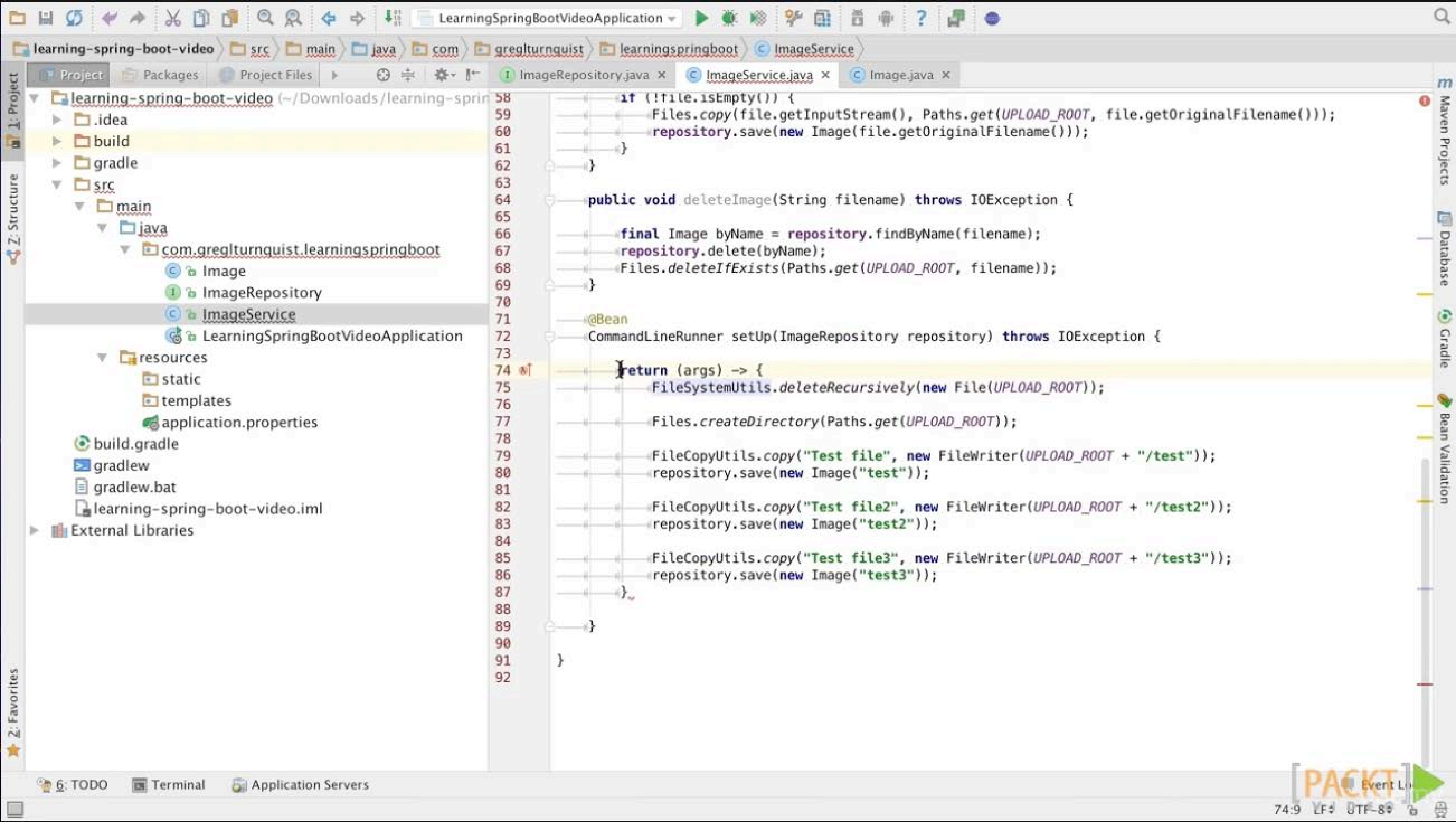
Unhandled exception: java.io.IOException



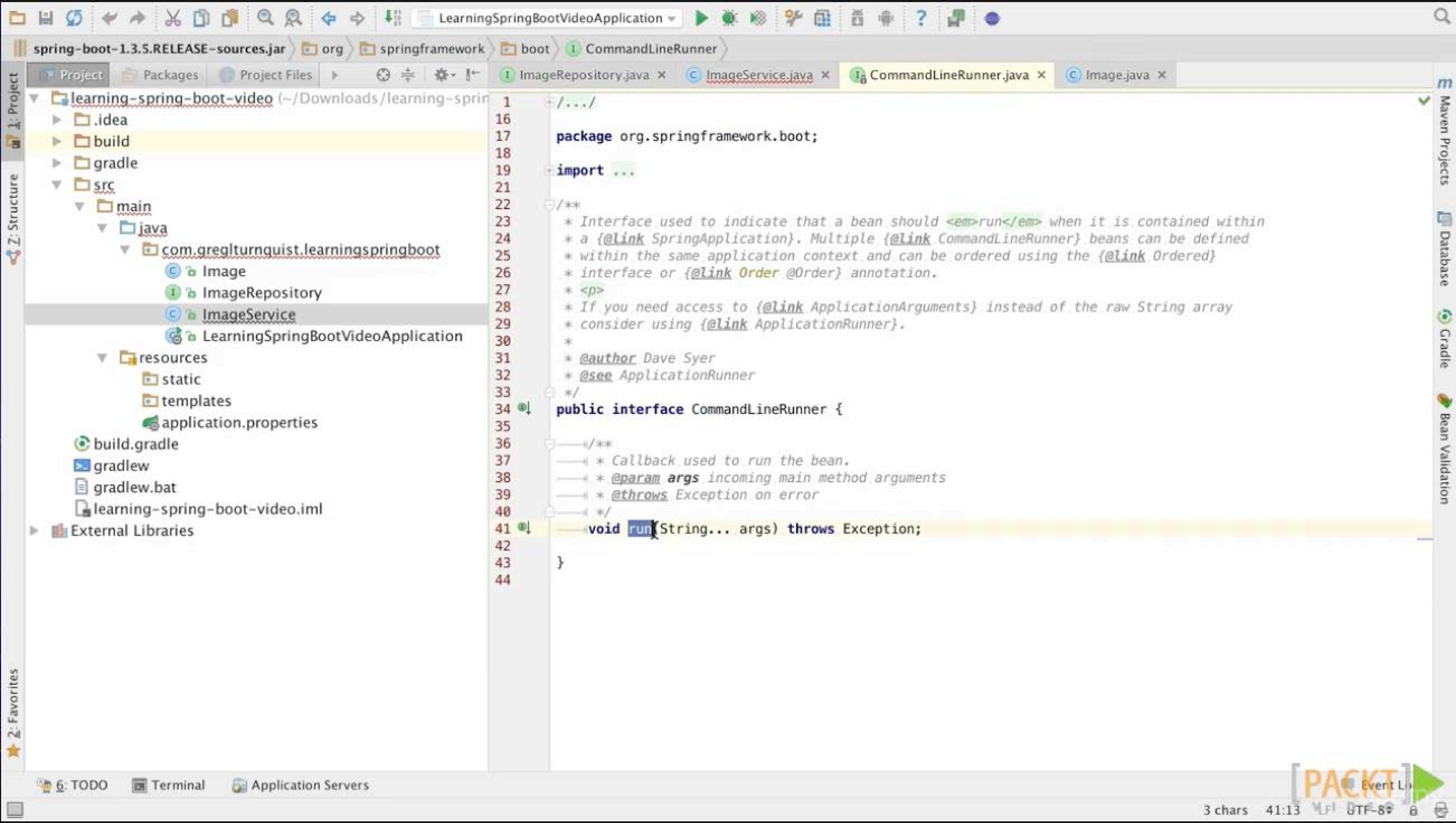
```java
public class ImageService {

    private static String UPLOAD_ROOT = "upload-dir";

    private final ImageRepository repository;
    private final ResourceLoader resourceLoader;

    @Autowired
    public ImageService(ImageRepository repository, ResourceLoader resourceLoader) {

        this.repository = repository;
        this.resourceLoader = resourceLoader;
    }

    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

    public void createImage(MultipartFile file) throws IOException {

        if (!file.isEmpty()) {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_ROOT, file.getOriginalFilename()));
            repository.save(new Image(file.getOriginalFilename()));
        }
    }

    public void deleteImage(String filename) throws IOException {

        final Image byName = repository.findByName(filename);
        repository.delete(byName);
        Files.deleteIfExists(Paths.get(UPLOAD_ROOT, filename));
    }

}
```



```java
    public Resource findOneImage(String filename) {
        return resourceLoader.getResource("file:" + UPLOAD_ROOT + "/" + filename);
    }

    public void createImage(MultipartFile file) throws IOException {

        if (!file.isEmpty()) {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_ROOT, file.getOriginalFilename()));
            repository.save(new Image(file.getOriginalFilename()));
        }
    }

    public void deleteImage(String filename) throws IOException {

        final Image byName = repository.findByName(filename);
        repository.delete(byName);
        Files.deleteIfExists(Paths.get(UPLOAD_ROOT, filename));
    }

    @Bean
    CommandLineRunner setUp(ImageRepository repository) throws IOException {

        FileSystemUtils.deleteRecursively(new File(UPLOAD_ROOT));

        Files.createDirectory(Paths.get(UPLOAD_ROOT));

        FileCopyUtils.copy("Test file", new FileWriter(UPLOAD_ROOT + "/test"));
        repository.save(new Image("test"));

        FileCopyUtils.copy("Test file2", new FileWriter(UPLOAD_ROOT + "/test2"));
        repository.save(new Image("test2"));

        FileCopyUtils.copy("Test file3", new FileWriter(UPLOAD_ROOT + "/test3"));
        repository.save(new Image("test3"));

    }

}
```



```java
        if (!file.isEmpty()) {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_ROOT, file.getOriginalFilename()));
            repository.save(new Image(file.getOriginalFilename()));
        }
    }

    public void deleteImage(String filename) throws IOException {

        final Image byName = repository.findByName(filename);
        repository.delete(byName);
        Files.deleteIfExists(Paths.get(UPLOAD_ROOT, filename));
    }

    @Bean
    CommandLineRunner setUp(ImageRepository repository) throws IOException {

        return (args) -> {
            FileSystemUtils.deleteRecursively(new File(UPLOAD_ROOT));

            Files.createDirectory(Paths.get(UPLOAD_ROOT));

            FileCopyUtils.copy("Test file", new FileWriter(UPLOAD_ROOT + "/test"));
            repository.save(new Image("test"));

            FileCopyUtils.copy("Test file2", new FileWriter(UPLOAD_ROOT + "/test2"));
            repository.save(new Image("test2"));

            FileCopyUtils.copy("Test file3", new FileWriter(UPLOAD_ROOT + "/test3"));
            repository.save(new Image("test3"));
        }

    }

}
```



```java
/.../

package org.springframework.boot;

import ...

/**
 * Interface used to indicate that a bean should <em>run</em> when it is contained within
 * a {@link SpringApplication}. Multiple {@link CommandLineRunner} beans can be defined
 * within the same application context and can be ordered using the {@link Ordered}
 * interface or {@link Order @Order} annotation.
 * <p>
 * If you need access to {@link ApplicationArguments} instead of the raw String array
 * consider using {@link ApplicationRunner}.
 *
 * @author Dave Syer
 * @see ApplicationRunner
 */
public interface CommandLineRunner {

    /**
     * Callback used to run the bean.
     * @param args incoming main method arguments
     * @throws Exception on error
     */
    void run(String... args) throws Exception;

}
```

Next Video

Creating a Spring MVC Controller to Handle Incoming Requests