

# 학습 내용 정리 - 2월 1주차

박시준

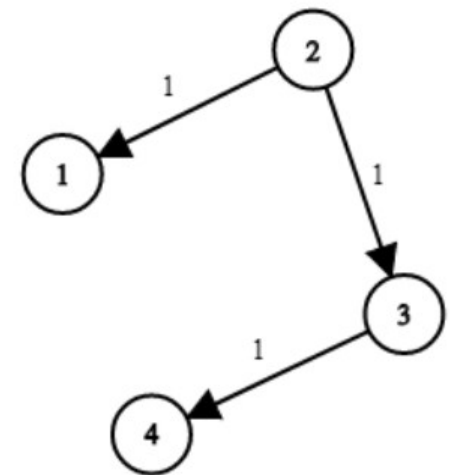
# Table of Contents

- Dijkstra Algorithm
- Filter, Interceptor
- Video DB connect
- Video upload, download
- Block simultaneous access when user login twice

# Dijkstra Algorithm

# [Leetcode] 743. Network Delay Time

- Given a network of nodes:  $n$
- Given array:  $\text{times}[i] = (u, v, w)$ 
  - $u$  is the source node
  - $v$  is the target node
  - $w$  is the time from source to target.
- Given starting node number:  $k$
- Return **the minimum time** for all the nodes to receive the signal.
- If impossible for all the nodes to receive the signal, return -1.



# [Leetcode] 743. Network Delay Time

- Dijkstra Algorithm problem
- Typical graph-greedy algorithm
- Prior knowledge: Graph structure, Priority Queue(Heap structure)

# Dijkstra Algorithm

- Shortest path Algorithm
- Greedy + BFS
- Greedy: Best choice for every node, every moment
- BFS: Breadth first based on distance array
  
- 2 additional data structure needed
  - Queue: to hold next target (to record distance from starting point)
  - Set(hash): to check already visited node

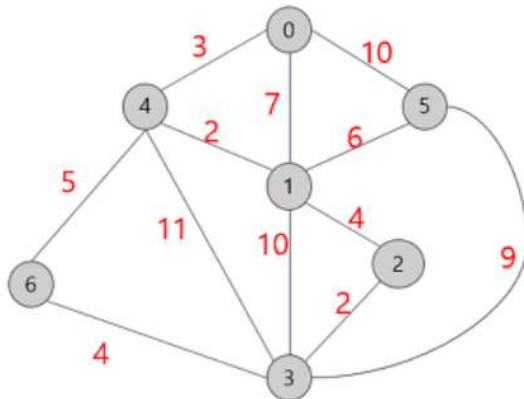
# Dijkstra Algorithm VS Kruskal Algorithm

- Common point: 1. Graph structure 2. Greedy Algorithm

<b>Dijkstra</b>	<b>Kruskal</b>
Given starting point	No starting point
Find distance with minimum cost	Make minimum spanning tree
Use weight to put nodes in priority queue	Use weight to sort
No need to link all the nodes (it depends on problems)	Every node has to be linked

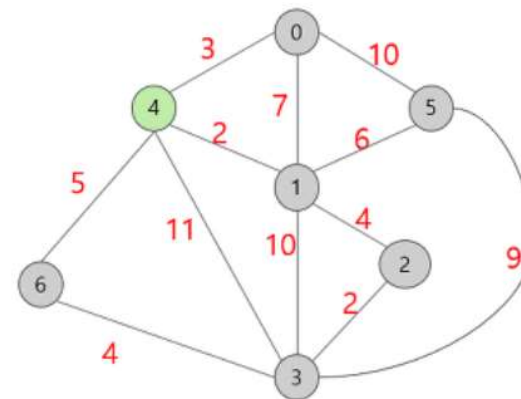
# Dijkstra Algorithm

- Start with given starting point
- Put starting points into visited array
- Search the next node that has smallest distance from original starting point except already visited node.



$S = \{0\}$

	0	1	2	3	4	5	6
distance[ ] =	0	7	$\infty$	$\infty$	3	10	$\infty$



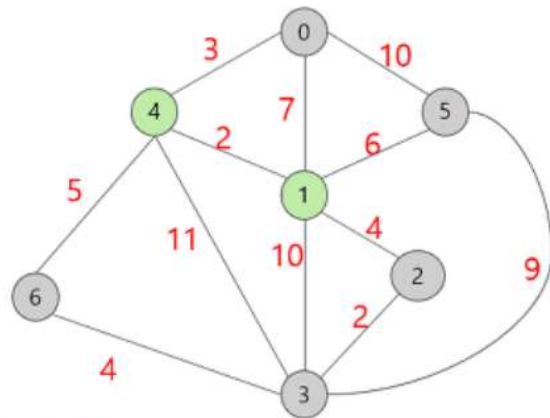
$S = \{0, 4\}$

	0	1	2	3	4	5	6
distance[ ] =	0	5	$\infty$	14	3	10	8

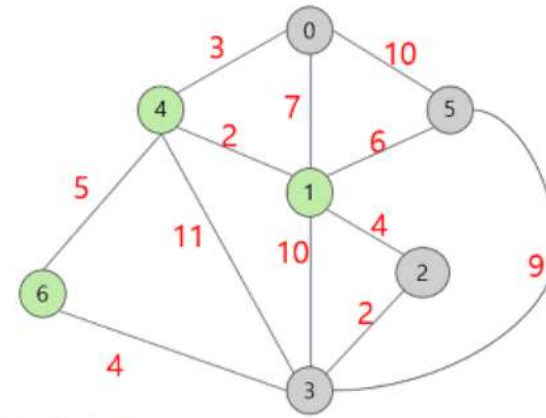


# Dijkstra Algorithm

- It needs 'visited' array (or set or hash)
- It needs 'distance' queue.
- Priority queue is faster than normal queue


$$S = \{0, 4, 1\}$$

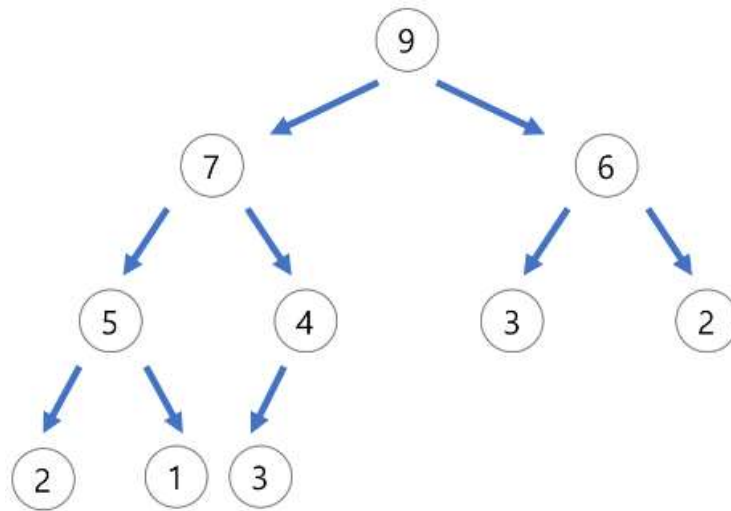
	0	1	2	3	4	5	6
distance[ ] =	0	5	9	14	3	10	8


$$S = \{0, 4, 1, 6\}$$

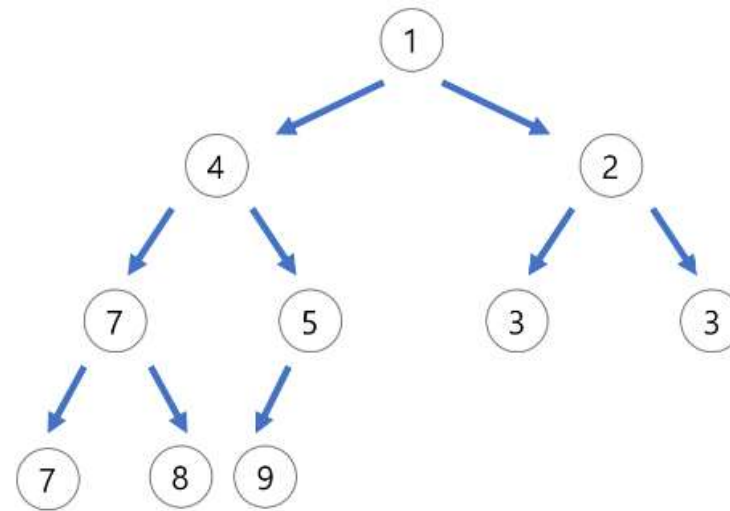
	0	1	2	3	4	5	6
distance[ ] =	0	5	9	12	3	10	8

# Priority Queue (Heap data structure)

- We use min heap for Dijkstra
- Min heap: child is bigger than parent (root is smallest)



-최대 힙(max heap)-



-최소 힙(min heap)-

# [Leetcode] 743. Network Delay Time

## • Implementation

```
class Solution:
    def networkDelayTime(self, times: List[List[int]], n: int, k: int) ->
        dic = {}
        visited = {}
        for i in range(1, n + 1):
            dic[i] = []
        for start, end, time in times:
            temp = dic[start]
            temp.append((end, time))
            dic[start] = temp

        heap = [(math.inf, 0), (0, k)]

        while len(heap) > 1:
            root = dequeue(heap)
            if not root:
                continue
            culmulative = root[0]
            node = root[1]
            if node not in visited:
                visited[node] = culmulative
                for end, time in dic[node]:
                    new_time = culmulative + time
                    heap = enqueue(heap, (new_time, end))
```

```
        if len(visited) == n:
            return max(visited.values())
        return -1

    def enqueue(heap, value):
        heap.append(value)
        i = len(heap) - 1
        while i > 1:
            if heap[i // 2][0] < heap[i][0]:
                break
            swap(heap, i // 2, i)
            i = i // 2
        return heap
```

# [Leetcode] 743. Network Delay Time

- Implementation

```
def dequeue(heap):
    if len(heap) == 1:
        return None
    root = heap[1]
    heap[1] = heap[len(heap) - 1]
    heap.pop()
    i = 1
    while i * 2 <= len(heap) - 1:
        if (i * 2 + 1) > len(heap) - 1:
            if heap[i][0] < heap[i * 2][0]:
                break
            swap(heap, i, i * 2)
            i = i * 2
            continue
        if heap[i][0] < heap[i * 2][0] and heap[i][0] < heap[i * 2 + 1][0]:
            break
        elif heap[i * 2][0] < heap[i * 2 + 1][0]:
            swap(heap, i, i * 2)
            i = i * 2
        else:
            swap(heap, i, i * 2 + 1)
            i = i * 2 + 1
    return root
```

```
def swap(heap, i, j):
    heap[i], heap[j] = heap[j], heap[i]
```

# Q. Module을 적극적으로 사용해야 할까?

- Too many code to implement myself
- There is convenient module in Python
  - heapq
  - defaultdict

```
from heapq import *
class Solution:
    def networkDelayTime(self, times: List[List[int]], n: int, k: int) ->
        graph = collections.defaultdict(list)
        for u, v, w in times:
            graph[u].append((v, w))

        Q = [(0, k)]
        dist = collections.defaultdict(int)

        while Q:
            time, node = heapq.heappop(Q)
            if node not in dist:
                dist[node] = time
                for v, w in graph[node]:
                    alt = time + w
                    heapq.heappush(Q, (alt, v))

        if len(dist) == n:
            return max(dist.values())

        return -1
```

# 763. Partition Labels

# [Leetcode] 763. Partition Labels

- Two pointer Algorithm, Greedy Algorithm, Hash Table

## 763. Partition Labels

Hint 

Medium



9.1K

341



Amazon



Facebook



Uber



You are given a string `s`. We want to partition the string into as many parts as possible so that each letter appears in at most one part.

Note that the partition is done so that after concatenating all the parts in order, the resultant string should be `s`.

Return a list of integers representing the size of these parts.

# [Leetcode] 763. Partition Labels

- Not easy to understand, actually very confused.



fadi17

Can someone please explain me this question :(

↑ 19 ↓ Show 1 Replies ↩ Reply ↻ Share ...



farhan786

Can someone explain the problem ??? I don't understand it!

↑ 5 ↓ ↩ Reply



RowidaNurElDin

Problem Explanation:

Given a string, divide it into partitions where:

- characters at each partition DOES NOT appear in any other partition then return the length of each one.

Good Luck!

↑ 3 ↓ ↩ Reply ↻ Share ...



# [Leetcode] 763. Partition Labels

- Why couldn't I solve this?  
⇒ I'm good at handling two pointer.  
⇒ Not good at handling hash table(dictionary)
- Hash table can provide the position(index) of specific character. (ex: last index, first index)

```
class Solution:
    def partitionLabels(self, s: str) -> List[int]:
        answer = []
        last = {}
        for i in range(len(s)):
            last[s[i]] = i

        left = 0
        right = 0

        for i in range(len(s)):
            right = max(right, last[s[i]])
            if i == right:
                answer.append(right - left + 1)
                left = i + 1

        return answer
```

테스트 프로젝트

# 서비스 개요 및 기능

- 영상을 등록하는 어플리케이션
- 사용자는 로그인할 수 있음
- 홈 화면에서 작업 중인 영상 목록을 볼 수 있음
- 영상 목록에서 영상을 누르면 영상 화면으로 갈 수 있음

# 비즈니스 요구사항 정리

- 데이터

- 1) 고객 (id, 이름, email, password, 멤버십 유무) **ok**
- 2) 영상 (id, 등록날짜) **ok**

- 기본 기능

- 1) 로그인 **ok**
- 2) 파일 등록
- 3) 파일 목록 조회
- 4) 파일 편집

Filter, Inteceptor

# Filter? Interceptor?

- Common function: to block the user who doesn't have a right to access some URL.
- Servlet – Filter
- Spring – Interceptor

**<Call order>**

**HTTP -> WAS -> Filter -> Servlet -> Interceptor -> Controller**

# Filter – Servlet

- Filter is basically provided by servlet.
- we can use it by doing 'implements'

```
public class MyFilter implements Filter{
```

- Override method 'doFilter' and put all the function in it

```
public void doFilter(ServletRequest req, ServletResponse resp,  
    FilterChain chain) throws IOException, ServletException {
```

# Filter Registration

- FilterRegistrationBean

```
@Bean
public FilterRegistrationBean<FirstFilter> firstFilter(){
    FilterRegistrationBean<FirstFilter> registrationBean = new FilterRegistrationBean<>();
    registrationBean.setFilter(new FirstFilter());
    registrationBean.addUrlPatterns("/user/*");
    registrationBean.setOrder(1);
    registrationBean.setName("first-filter");
    return registrationBean;
}
```



# Interceptor - Spring

- Interceptor
- Served by Spring
- Can control URL very effectively

```
import org.springframework.web.servlet.HandlerInterceptor;
```

**<Call order>**

**HTTP -> WAS -> Filter -> Servlet -> Interceptor -> Controller**

# Interceptor - Spring

- Implements HandlerInterceptor and override method that you need

```
@Slf4j
public class LoginCheckInterceptor implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
        String requestURI = request.getRequestURI();
        log.info("Certification interceptor activated {}", requestURI);

        HttpSession session = request.getSession();
```

- There are three types of method
  - **preHandle**: Before controller called, mainly treated
  - postHandle: After controller called, Not activated when an exception occurs
  - afterCompletion: After rendering view, Always activated

# Interceptor Registration

- Make new config class and implements "WebMvcConfigurer".

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new LogInterceptor())
            .order(1)
            .addPathPatterns("/**")
            .excludePathPatterns("/css/**", "/&.ico", "/error");
    }
}
```

- And override "addInterceptors".
- No need to register 'Bean'.

# Interceptor vs Filter

- Filter

```
//@Bean
public FilterRegistrationBean loginCheckFilter() {
    FilterRegistrationBean<Filter> filterRegistrationBean = new FilterRegistrationBean<>();
    filterRegistrationBean.setFilter(new LoginCheckFilter());
    filterRegistrationBean.setOrder(2);
    filterRegistrationBean.addUrlPatterns( ...urlPatterns: "/*");
    return filterRegistrationBean;
}
```

```
private static final String[] whitelist = {"/", "/members/add", "/login", "/logout", "/css/*"};
```

```
private boolean isLoginCheckPath(String requestURI){
    return !PatternMatchUtils.simpleMatch(whitelist, requestURI);
}
```

# Interceptor vs Filter

- Interceptor

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new LoginCheckInterceptor())
            .order(1)
            .addPathPatterns("/**")
            .excludePathPatterns("/", "/members/add", "/login", "logout", "/css/**", "&.ico", "/error");
    }
}
```

- Interceptor is more convenient

# Interceptor 적용하기

- Override preHandle from HandlerInterceptor
- Use session from HttpServletRequest and check if exist

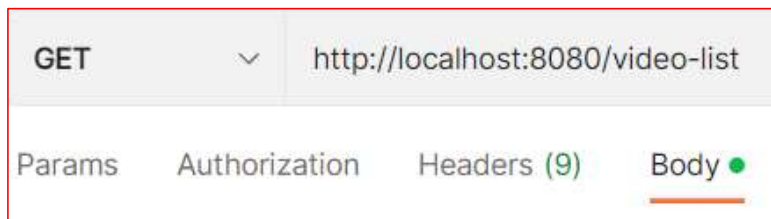
```
@Slf4j
public class LoginCheckInterceptor implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
        HttpSession session = request.getSession();
        if (session == null || session.getAttribute("loginMember") == null){
            log.info("non-certified access");
            return false;
        }
        return true;
    }
}
```

# Test: Access to video list

- Try access to video-list
- Spring Interceptor blocked the access

<postman>



<controller>

```
@ResponseStatus(HttpStatus.OK)
@GetMapping("/video-list")
public String videoList(){
    return "video";
}
```

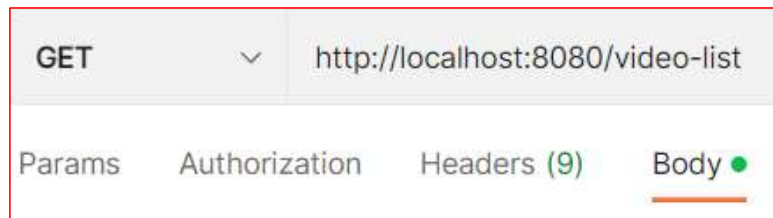
<console>

```
o-8080-exec-8] t.r.interceptor.LoginCheckInterceptor : non-certified access
o-8080-exec-1] t.r.interceptor.LoginCheckInterceptor : non-certified access
```

# Test: After login

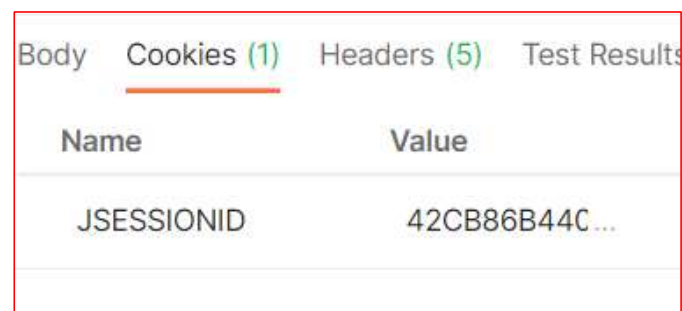
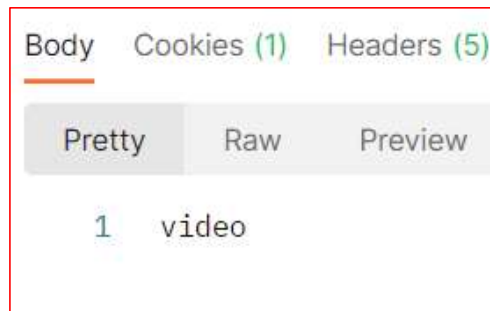
- Try access to video-list after login
- Access success. It has session id.

<postman>



<controller>

```
@ResponseStatus(HttpStatus.OK)
@GetMapping("/video-list")
public String videoList(){
    return "video";
}
```





Video connected to DB

# Video entity connected to DB

- Connect it to DB by using Spring Data Jpa

```
public interface SpringDataJpaVideoRepository extends JpaRepository<Video, Long> {  
  
    List<Video> findByUsername (String username);  
  
}
```

```
@Slf4j  
@Repository  
@Primary  
@Transactional  
public class SpringDataVideoRepository implements VideoRepository {  
  
    private final SpringDataJpaVideoRepository repository;  
  
    public SpringDataVideoRepository(SpringDataJpaVideoRepository repository) { this.repository = repository; }  
  
    @Override  
    public void make(Video video) {  
        repository.save(video);  
    }  
  
    @Override  
    public Optional<Video> findById(Long id) {  
        return repository.findById(id);  
    }  
}
```

# Stream? List?

- Video data model (I used to make it as Stream)

```
public interface VideoRepository {  
    void make(Video video);  
    Optional<Video> findById(Long id);  
    Stream<Video> findByUsername(String userName);  
}
```

```
import java.util.List;  
import java.util.stream.Stream;
```

```
public interface VideoRepository {  
    void make(Video video);  
    Optional<Video> findById(Long id);  
    List<Video> findByUsername(String userName);  
}
```

# Stream? List?

- What is difference between using List and Stream?

```
@Override
public Stream<Video> findByUsername(String userName) {
    return store.values().stream().filter(video -> video.getUserName().equals(userName));
}
```

```
public interface SpringDataJpaVideoRepository extends JpaRepository<Video, Long> {

    Stream<Video> findByUsername (String username);
}
```

VS

```
@Override
public List<Video> findByUsername(String userName) {
    return store.values().stream().filter(video -> video.getUserName().equals(userName)).toList();
}
```

```
public interface SpringDataJpaVideoRepository extends JpaRepository<Video, Long> {

    List<Video> findByUsername (String username);
}
```

# Difference between List and Stream

- Stream is data structure that provide a set of data that matches on condition
- Data can not be added to or deleted from Stream data structure.
- Stream data can be consumed only once

```
List<String> title = Arrays.asList("kim", "seo", "hae");  
Stream<String> s = title.stream();  
s.forEach(System.out::println);  
s.forEach(System.out::println); // java.lang.IllegalStateException : 스트림이 이미 소비되었거나 닫힘
```

- Iterated internally
- Normally used for iteration

```
List<String> names = menu.stream()  
                        .map(Dish::getName)  
                        .collect(toList());
```

# Difference between List and Stream

- List is one of the data type in Collection
- Collection is data structure that is used to store the data
- Should be externally iterated

```
List<String> names = new ArrayList<>();  
for (Dish dish : menu) {  
    names.add(dish.getName());  
}
```

# Difference between Collection and Stream

Collections	Streams
Collections are mainly used to store and group the data.	Streams are mainly used to perform operations on data.
You can add or remove elements from collections.	You can't add or remove elements from streams.
Collections have to be iterated externally.	Streams are internally iterated.
Collections can be traversed multiple times.	Streams are traversable only once.
Collections are eagerly constructed.	Streams are lazily constructed.
Ex : <b>List</b> , Set, Map...	Ex : filtering, mapping, matching...

# Video connected to DB

```
public interface SpringDataJpaVideoRepository extends JpaRepository<Video, Long> {  
  
    List<Video> findByUsername (String userName);  
}
```

```
@Override  
public List<Video> findByUsername(String username) {  
    return repository.findByUsername(username);  
}
```



# [warning] Raw use of 'List'

- Without generic, There was warning related to "Raw type"
- List is data structure who has parameter of data type
- 'Raw type' means Generic type without type parameter
- This is not error, just warning.
- But most people said "Never use raw type because it eliminate every advantage of generic."
- Compile error is best error.  
(Actually, it wasn't an error)

```
public List findVideoForUser(String username){  
    return videoRepository.findByUsername(username);  
}
```

Raw use of parameterized class 'List'

```
java.util  
public interface List<E>  
    extends java.util.Collection<E>
```

An ordered collection (also known as a *sequence*).  
The user of this interface has precise control over  
where in the list each element is inserted. The  
user can access elements by their integer index  
(position in the list), and search for elements in

# [warning] Raw use of 'List'

- Add generic
- Problem solved.

```
public List<Video> findVideos(String username){  
    return videoRepository.findByUsername(username);  
}
```

# Code - Video Controller

- Video list for each user
- Video add
- Replace this String 'name' to name of original file later.

```
@Getter
@Setter
public class VideoForm {
    private String name;
}
```

```
@ResponseStatus(HttpStatus.OK)
@GetMapping("/video-list")
public List<Video> videoList(HttpServletRequest request){
    HttpSession session = request.getSession();
    Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);
    List<Video> videoList = videoService.findVideos(loginMember.getName());
    for (Video video: videoList){
        log.info("video={}", video);
    }
    return videoList;
}
```

```
@PostMapping("/video/add")
public void videoAdd(@RequestBody VideoForm data, HttpServletRequest request)
{
    HttpSession session = request.getSession();
    Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);
    videoService.saveVideo(data.getName(), loginMember.getName());
    response.sendRedirect(location: "/video-list");
}
```

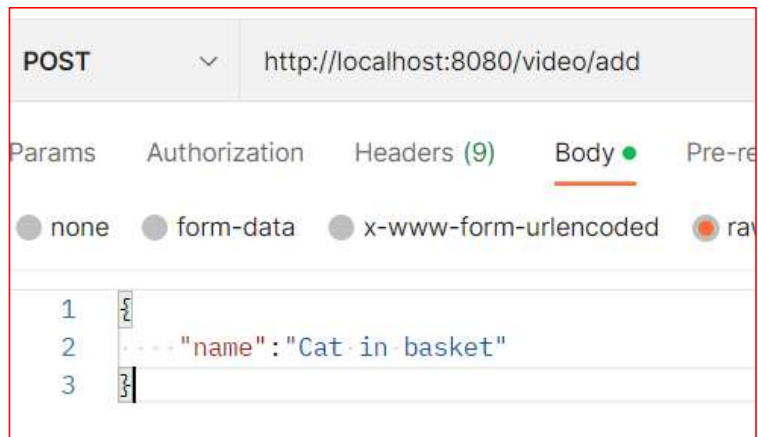
# Code - Video Service

- Video list
- Video add
- Use 'LocalDatetime' for Date data

```
public List<Video> findVideos(String username){  
    return videoRepository.findByUsername(username);  
}  
  
public void saveVideo(String name, String username){  
    Video video = new Video();  
    video.setName(name);  
    video.setDate(LocalDate.now());  
    video.setUsername(username);  
    videoRepository.save(video);  
}
```

# Add video function

- Video data added in H2



```
SELECT * FROM VIDEO;
```

ID	NAME	DATE	USERNAME
1	Cat in basket	2023-02-05 19:35:18.941014	season

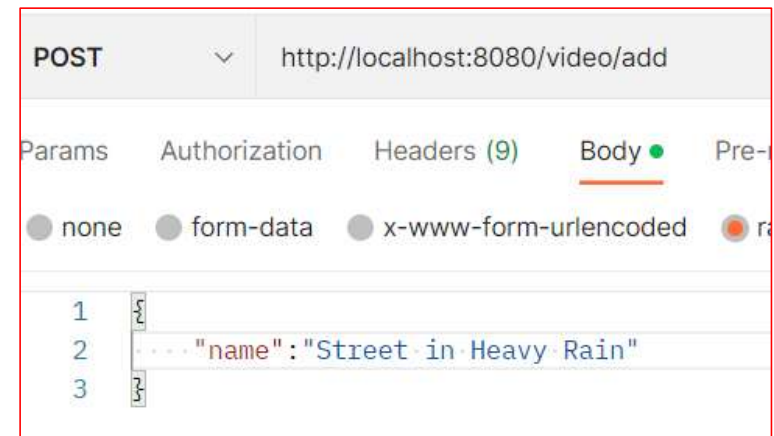
(1 row, 1 ms)

# Show videos list function

- Redirect to “/video-list” after successful add

```
Member loginMember = (Member)session.getAttribute("loginMember");  
videoService.saveVideo(data.getName(), loginMember);  
response.sendRedirect("/video-list");
```

```
@GetMapping("/video-list")  
public List<Video> videoList(HttpServletRequest request, HttpSession session) {  
    Member loginMember = (Member)session.getAttribute("loginMember");  
    List<Video> videoList = videoService.getVideoList(loginMember);  
    for (Video video: videoList) {  
        log.info("video={}", video);  
    }  
    return videoList;  
}
```



# Redirect and send back

- Redirect to login page when non-login user accesses “/video-list”

<preHandle of Interceptor>

```
if (session == null || session.getAttribute( name: "loginMember") == null){  
    log.info("non-certified access");  
    response.sendRedirect( location: "/login?redirect=" + requestURI);  
    return false;  
}
```

- Send back to URL the user originally wanted to access after login.

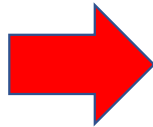
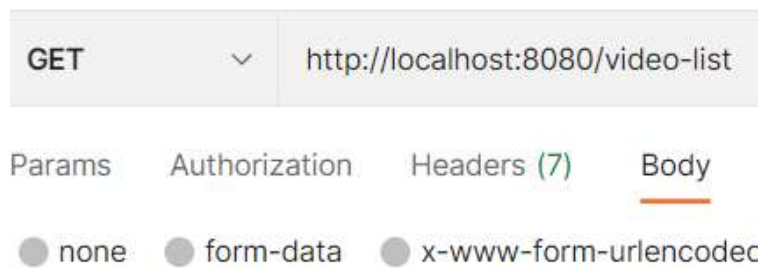
```
@PostMapping("/login")  
public void login(  
    @RequestBody LoginForm data,  
    @RequestParam(defaultValue = "/" ) String redirectURL,  
    HttpServletRequest request,  
    HttpServletResponse response
```

```
session.setAttribute(Const.LOGIN_MEMBE  
response.sendRedirect(redirectURL)
```



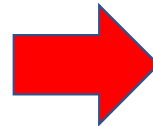
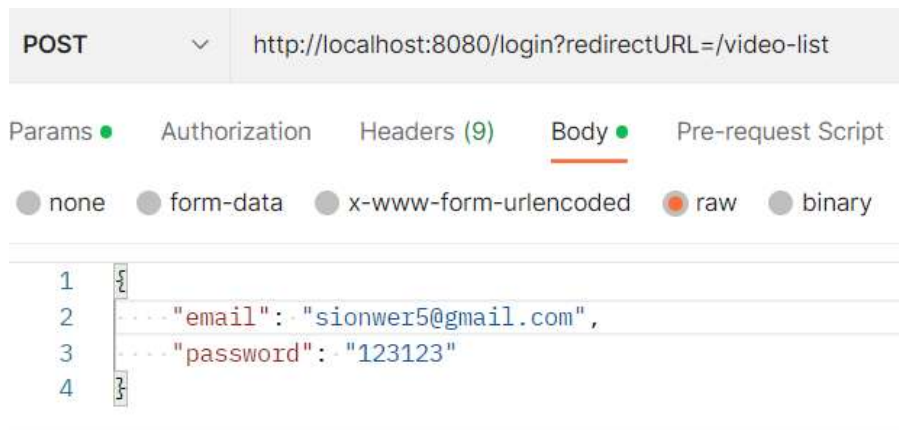
# Redirect and send back

- Redirect to login page



certification activated.  
non-certified access

- Send back to URL





# Video Upload, Download

# HTML data transmission method

- application/x-www-form-urlencoded
  - HTML default, it sends letter through HTML Body
- multipart/form-data
  - It sends multiple part of data. It can send letter or file binary
- application/json
  - Usually used to get and post data in method of REST API

# Multipart and MultipartFile

- Spring provide the interface 'MultipartFile' with @RequestParam to process a file of multiple part of HTTP body conveniently.
- But this is the method in case of Form-data.

```
@Controller
public class TestController {
    // file upload 처리 핸들러
    @ResponseBody
    @PostMapping("/file")
    public void upload(@RequestParam("file") MultipartFile file) {
        System.out.println("file name : " + file.getName());
    }
}
```

# HttpMediaTypeNotSupportedException

- @RequestParam for MultipartFile
- @RequestBody for VideoForm

POST http://localhost:8080/video/upload

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	CONTENT TYPE
<input checked="" type="checkbox"/>	file	One-Finger-512.png x	Auto
<input checked="" type="checkbox"/>	videoForm	{"name": "created"}	application/json
	Key	Value	Auto

```
@PostMapping("/video/add")
public void videoAdd(
    @RequestParam MultipartFile file,
    @RequestBody VideoForm videoForm,
    HttpServletRequest request, HttpServletResponse response) throws IOException
```

```
[org.springframework.web.HttpMediaTypeNotSupportedException:
```

```
Content-Type 'multipart/form-data;boundary=-----214474703808139517035829;charset=UTF-8' is not supported]
```

# HttpMessageNotReadableException

- @RequestParam for MultipartFile
- @RequestBody for String

```
@PostMapping("/video/upload")
public void videoUpload(
    @RequestParam MultipartFile file,
    @RequestBody String videoName,
    HttpServletRequest request, HttpServletResponse response) throws
```

```
.HttpMessageNotReadableException: Required request body is missing:
```

POST ⌵ http://localhost:8080/video/upload

Params Authorization Headers (9) **Body** ● Pre-request Script Tests

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ Gr

	KEY	VALUE
<input checked="" type="checkbox"/>	file	One-Finger-512.png <span>×</span>
<input checked="" type="checkbox"/>	videoName	pigeon

# MethodArgumentConversionNotSupportedException

- @RequestParam for both MultipartFile and VideoForm

```
@PostMapping("/video/add")
public void videoAdd(
    @RequestParam MultipartFile file,
    @RequestParam VideoForm videoForm,
    HttpServletRequest request, HttpServletResponse response) throws IOException {

    HttpSession session = request.getSession();
    Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);
    videoService.saveVideo(videoForm.getName(), loginMember.getName());
}
```

POST http://localhost:8080/video/upload

Params Authorization Headers (9) Body Pre-request Script Tests

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> file	One-Finger-512.png x
<input checked="" type="checkbox"/> videoForm	{"name":"pig"}

```
springframework.web.method.annotation.MethodArgumentConversionNotSupportedException:
```

```
Failed to convert value of type 'java.lang.String' to required type 'training.restapi.form.VideoForm';
```

# Form data Body structure

- HTTP Body is divided by boundary when using multipart/form-data
- Use "multipart/form-data" method of HTTP, combine JSON and file data and put them into form-data.
- One part is for JSON
- Other parts are for files

```
POST /save HTTP/1.1
Host: localhost:8080
Content-Type: multipart/form-data; boundary=-----XXX
Content-Length: 10457

-----XXX
Content-Disposition: form-data; name="username"

kim
-----XXX
Content-Disposition: form-data; name="age"

20
-----XXX
Content-Disposition: form-data; name="file1"; filename="intro.png"
Content-Type: image/png

109238a9o0p3eqwokjasd09ou3oirjwoe9u34ouief...
-----XXX--
```

끝에는 -- 추가

# JSON with Form => @RequestPart

- Use "@RequestPart" instead of "@RequestParam"

```
@RestController
@Slf4j
public class TestController {

    @PostMapping(value = "/api/v1/character", consumes = {MediaType.APPLICATION_JSON_VALUE, MediaType.
MULTIPART_FORM_DATA_VALUE})
    public void saveCharacter(@RequestPart CharacterCreateRequest request,
                             @RequestPart MultipartFile imgFile) {
        log.info("이름 : {}, 나이 : {}, 이미지 : {}", request.getAge(), request.getName(), imgFile);
    }
}
```



# Upload Success!

- Use @RequestParam

```
@PostMapping("/video/add")
public void videoAdd(
    @RequestParam MultipartFile file,
    @RequestParam VideoForm videoForm,
    HttpServletRequest request, HttpServletResponse response) throws IOException {

    HttpSession session = request.getSession();
    Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);
    videoService.saveVideo(videoForm.getName(), loginMember.getName());

    if (!file.isEmpty()) {
        String fullPath = fileDir + file.getOriginalFilename();
        file.transferTo(new File(fullPath));
    }

    response.sendRedirect(location: "/video/list");
}
```

# Upload Success!

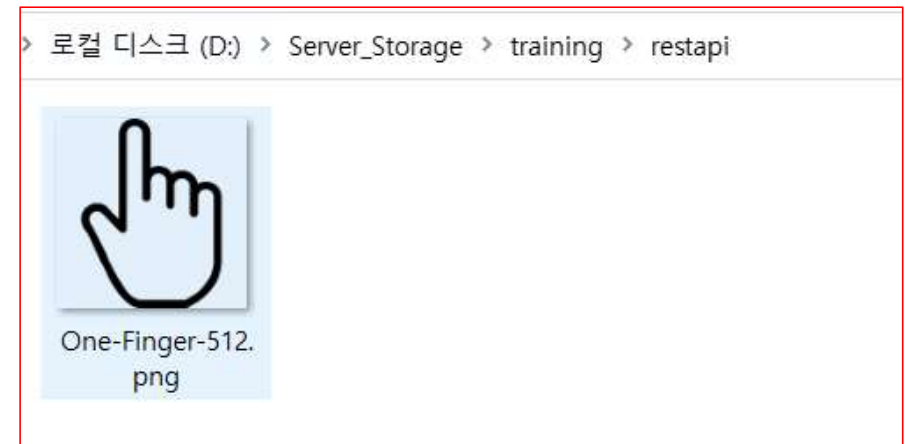
- Server storage has the uploaded file

POST ▼ http://localhost:8080/video/add

Params Authorization Headers (9) Body ● Pre-request Script Tests

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	videoForm	{"name": "Finger"}
<input checked="" type="checkbox"/>	file	One-Finger-512.png ×
	Key	Value



```
{
  "date": "2023-02-06T15:55:25.551089",
  "username": "season"
},
{
  "id": 36,
  "name": "Finger",
  "date": "2023-02-06T17:12:18.215687",
  "username": "season"
}
```

# Upload Success!

- No problem with getting a String data

```
@PostMapping("/video/upload")
public void videoUpload(
    @RequestPart MultipartFile file,
    @RequestPart String videoName,
    HttpServletRequest request, HttpServletResponse res)
```

POST ⌵ http://localhost:8080/video/upload

Params Authorization Headers (9) **Body** ● Pre-request Script Tests

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	file	One-Finger-512.png <span>×</span>
<input checked="" type="checkbox"/>	videoName	pigeon

# Fixed method according to Content-Type

- Stack overflow answer:
- **Just use right annotation following Content-Type**

map HTTP request header `Content-Type`, handle request body.

- `@RequestParam` ← `application/x-www-form-urlencoded`,
- `@RequestBody` ← `application/json`,
- `@RequestPart` ← `multipart/form-data`,

# Fixed method according to Content-Type

- [RequestParam \(Spring Framework 5.1.9.RELEASE API\)](#)

map to query parameters, form data, and parts in multipart requests.

`RequestParam` is likely to be used with name-value form fields

- [RequestBody \(Spring Framework 5.1.9.RELEASE API\)](#)

bound to the body of the web request. The body of the request is passed through an

**HttpMessageConverter** to resolve the method argument depending on the `content type` of the request. (e.g. JSON, XML)

- [RequestPart \(Spring Framework 5.1.9.RELEASE API\)](#)

used to associate the part of a "`multipart/form-data`" request

`RequestPart` is likely to be used with parts containing more complex content

# Q. Why is this working?

- @RequestParam for both MultipartFile and String

```
@PostMapping("/video/upload")
public void videoUpload(
    @RequestParam MultipartFile file,
    @RequestParam String videoName,
    HttpServletRequest request, HttpServletResponse response) throws IOException {

    HttpSession session = request.getSession();
    Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);
```

POST http://localhost:8080/video/upload

Params Authorization Headers (9) Body Pre-request Script Tests







☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	file	One-Finger-512.png x
<input checked="" type="checkbox"/>	videoName	Endgame

```
select vi_0.id,vi_0.date,vi_0.name,vi_0.username from video vi_0 where vi_0.username=?
video=Video(id=67, name=Endgame.png, date=2023-02-08T07:20:44.767712, username=season)
video=Video(id=68, name=Endgame.png, date=2023-02-08T07:22:27.874733, username=season)
```

# Q. Why is this working?

- Table of experiment

<p>@RequestParam MultipartFile @RequestBody VideoForm</p>  <p>HttpMediaNotSupportedException</p>	<p>@RequestParam MultipartFile @RequestParam VideoForm</p>  <p>MethodArgumentConversionNot SupportedException</p>	<p>@RequestPart MultipartFile @RequestPart VideoForm</p> 
<p>@RequestParam MultipartFile @RequestBody String</p>  <p>HttpMessageNotReadableException</p>	<p>@RequestParam MultipartFile @RequestParam String</p> 	<p>@RequestPart MultipartFile @RequestPart String</p> 

# Download function

- First step: Use path variable to distinguish each file

```
@GetMapping("/video/{id}/download")
public ResponseEntity<Resource> videoDownload(
    @PathVariable Long id,
    HttpServletRequest request,
    HttpServletResponse response) throws IOException {
```



# Download function

- Second step: make 'UrlResource' and put it into body of response.
- Spring framework provides 'UrlResource'
- It is essential to set header as content disposition

```
String videoName = video.get().getName();
UrlResource resource = new UrlResource(path: "file:" + fileDir + videoName);
String encodedName = UriUtils.encode(videoName, StandardCharsets.UTF_8);
String contentDisposition = "attachment; filename=\"" + encodedName + "\"";
return ResponseEntity.ok()
    .header(HttpHeaders.CONTENT_DISPOSITION, ...headerValues: contentDisposition)
    .body(resource);
```

# Content Disposition?

- Web MDN definition

## Content-Disposition

In a regular HTTP response, the `content-disposition` response header is a header indicating if the content is expected to be displayed *inline* in the browser, that is, as a Web page or as part of a Web page, or as an *attachment*, that is downloaded and saved locally.

- if header isn't Content-Disposition, web-browser just render it, doesn't download.

# Download source code

```
@GetMapping("/video/{id}/download")
public ResponseEntity<Resource> videoDownload(
    @PathVariable Long id,
    HttpServletRequest request,
    HttpServletResponse response) throws IOException {

    Optional<Video> video = videoService.findVideoById(id);
    if (video.isEmpty()){
        response.sendRedirect( location: "/video/list");
        return null;
    }
    String videoName = video.get().getName();
    UrlResource resource = new UrlResource( path: "file:" + fileDir + videoName);
    String encodedName = UriUtils.encode(videoName, StandardCharsets.UTF_8);
    String contentDisposition = "attachment; filename=\"" + encodedName + "\"";
    return ResponseEntity.ok()
        .header(HttpHeaders.CONTENT_DISPOSITION, ..headerValues: contentDisposition)
        .body(resource);
}
```

# [Error] FileNotFoundException

- There is no file such as 'Finger'

```
.FileNotFoundException Create breakpoint : URL [file:D:/Server_Storage/training/restapi/Finger] ca  
org.springframework.core.io.AbstractFileResolvingResource.getLength(AbstractFileResolving  
org.springframework.http.converter.ResourceHttpMessageConverter.getLength(ResourceHttp
```



getOriginalFilename() when upload

```
if (!file.isEmpty()) {  
    String fullPath = fileDir + file.getOriginalFilename();  
    file.transferTo(new File(fullPath));  
}
```

# return to upload

- There is no original name in my DB

```
HttpSession session = request.getSession();
Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);
videoService.saveVideo(videoForm.getName(), loginMember.getName());
```

**No Original name!**

```
public void saveVideo(String name, String username){
    Video video = new Video();
    video.setName(name);
    video.setDate(LocalDateTime.now());
    video.setUsername(username);
    videoRepository.save(video);
}
```

SELECT \* FROM VIDEO;

ID	NAME	DATE	USERNAME
1	Cat in basket	2023-02-05 19:35:18.941014	season
2	Street in Heavy Rain	2023-02-05 19:37:11.597609	season
33	Blow shot	2023-02-06 15:28:28.407212	season
34	Blow shot	2023-02-06 15:31:49.826766	season
35	End game	2023-02-06 15:55:25.551089	season
36	Finger	2023-02-06 17:12:18.215687	season

(6 행, 5 ms)


# Throw away the original name

- Wait... No need to get original name.
- Throw away the original name and just use the name user gave!

```
@PostMapping("/video/upload")
public void videoUpload(
    @RequestParam MultipartFile file,
    @RequestParam VideoForm videoForm,
    HttpServletRequest request, HttpServletResponse response) throws IOException {

    HttpSession session = request.getSession();
    Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);

    if (!file.isEmpty()) {
        videoService.saveVideo(videoForm.getName(), loginMember.getName());
        String fullPath = fileDir + videoForm.getName();
        file.transferTo(new File(fullPath));
    }
}
```



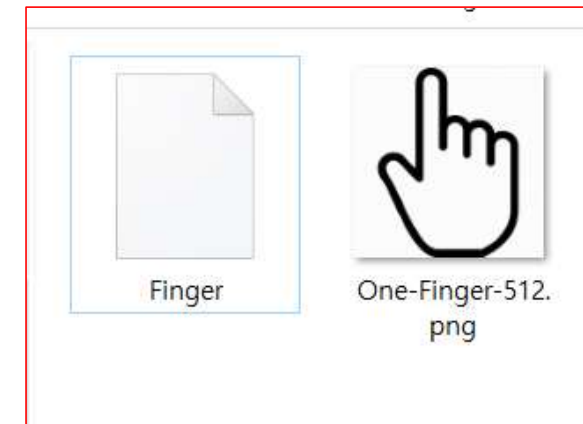


# Download success! But another problem

- There is no extension in filename

GET <http://localhost:8080/video/37/download>

```
1  PNG
2  →
3  █
4  IHDR███-███x███U!IDATx███•███Wu?███):███WuUdY███M███+H 0|███gmpbS███L  S███,███I███
   [e███{f███>███=j███4k███5███'███>|███?███0███;S███<███}E███•b███T███U███.███0███R███b███q███;███=R19███Ç███|███m███&███L[███
5  -███&███&███^███`███_███!
6  {F███f███{i███[███)b███L███<xU███/███n███z███GLL,
7  8_███Gb███†+E███!!███k███
8  {g███g███=K1)███•███^Y███V+u███c███V███5/X†███"███C███:o███+{███%███+███<xE███Xc███•███h^C███~h███Y^███<███
   o███0███F███†x███>F███I#███P'6███P███\███+███@|███7o>███8███#^███<xE███~
9  }JI███x███;em███X███|███•███^███fuaG███y███0███3███{███-███j███!!b5███F███!k
10  {^███*███J███
11  {███8███!███4y███8███0███0███{███uG███†x███-!!███km███+M███8███^███I|███70███g*███Xoc███W†███J███G███T███k███;3███m███<███
12  0|███p███~███†███(███p███;M███Y███?+███u███j███^███_███r███Ub███x███e███q███k███5███0███n███!███A███L███+███&███Z███+███#███
13  ]███|███•███^███
14  /███5███IV)███w███{███e███|███e███_███y███H███|███<x███u███
```



# Return to upload again

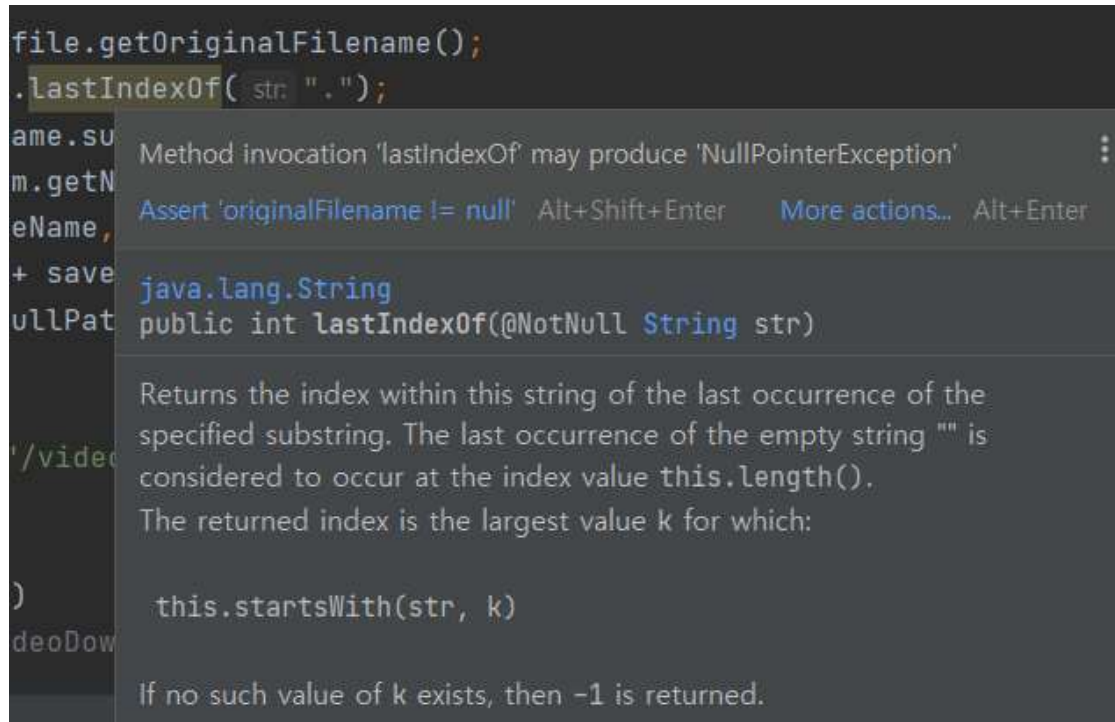
- Save file name including extension
- Use "getOriginalFilename()" again
- Some recommendation related to "NullPointerException" appeared.

```
if (!file.isEmpty()) {  
    String originalFilename = file.getOriginalFilename();  
    int pos = originalFilename.lastIndexOf(".");  
    String ext = originalFilename.substring(pos + 1);  
    String saveName = videoForm.getName() + "." + ext;  
    videoService.saveVideo(saveName, loginMember.getName());  
    String fullPath = fileDir + saveName;  
    file.transferTo(new File(fullPath));  
}
```



# 'lastIndexOf' warning about NullPointerException

- Method invocation 'lastIndexOf' may produce 'NullPointerException'



# 'lastIndexOf' warning about NullPointerException

- Wrap it up with "Objects.requireNonNull"

```
String originalFilename = file.getOriginalFilename();  
int pos = Objects.requireNonNull(originalFilename).lastIndexOf( str: ".");  
String ext = originalFilename.substring(pos + 1);
```

- Objects.requireNonNull: This is null check method. If null goes into parameter of this method, it will throw "NullPointerException"

# Upload fix

- Saved with extension

```
HttpSession session = request.getSession();
Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);

if (!file.isEmpty()) {
    String originalFilename = file.getOriginalFilename();
    int pos = Objects.requireNonNull(originalFilename).lastIndexOf( str: ".");
    String ext = originalFilename.substring(pos + 1);
    String saveName = videoForm.getName() + "." + ext;
    videoService.saveVideo(saveName, loginMember.getName());
    String fullPath = fileDir + saveName;
    file.transferTo(new File(fullPath));
}

response.sendRedirect( location: "/video/list");
```

**Q. Does this code  
should be in Service?**

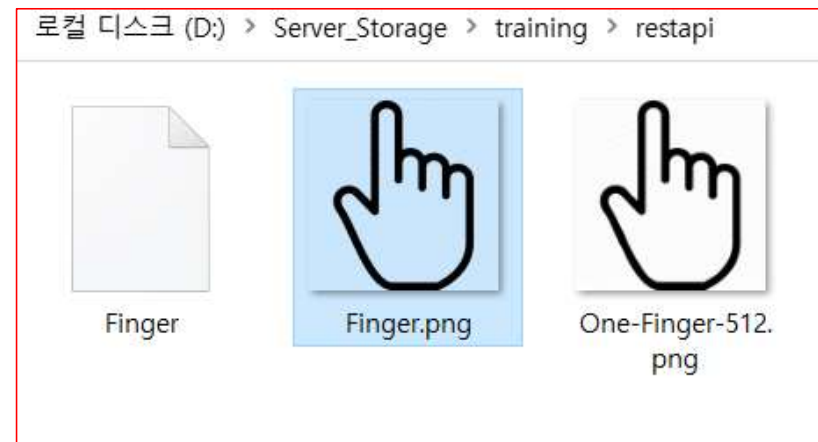
# Download finally complete

- Problem solved
- Successfully upload and download

```
SELECT * FROM VIDEO;
```

ID	NAME	DATE	USERNAME
38	Finger.png	2023-02-06 22:59:34.078298	season

(1 row, 2 ms)



# Video screen page

- If someone click the specific video among the video list, go to the page which shows the video

```
@GetMapping("/{id}")
public ResponseEntity<Resource> videoRender(
    @PathVariable Long id,
    HttpServletResponse response
) throws IOException{
    Optional<Video> video = videoService.findVideoById(id);
    if (video.isEmpty()){
        response.sendRedirect( location: "/video/list");
        return null;
    }
    String videoName = video.get().getName();
    UrlResource resource = new UrlResource( path: "file:" + fileDir + videoName);
    return ResponseEntity.ok().body(resource);
}
```

# Video screen page

- Return the resource without any header configuration

Body Cookies (1) Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1  PNG
2  →
3  ???
4  IHDR???-???x???U↓IDATx???•?Wu??:?W?uU?dY?M??+H 0|???pmpbS???L    'S?,,,?I?'→?
    [e???{f???>?=j?▲?□4k?5? ' ">|?♀???Ŏ;S?<?$?}E?•b?τ?U?.???Ŭ?R?b?♀;???=Rl9???Ç???|▲???mž?&L[?#?ρ
5  ←???&&&?^?'_?!
6  {F?f?{r????[?)b???L<xU???/???n???z???Gll,
7  |
8  |      8_???Gb?↑+E?!!?k?
9  |      {g?ǔ?=Kl)?•$^Y???V?u???c???V!???ō/X+?"???C?:o????+{?%?+|▲<xE?□???Xc?•?h^C???~h?Y^?<?□D,?
10 |      o?Ŏ?F?↑x????>F?r#??P'6????\?+@|????7o>□???ž?#^?<xE?~
11 |      {JI??x;em?ōX?||?•$^▲????fuaG????y????-?3?{???-???j!!b5?F!!k
12 |      {^?×?J?
13 |      {?ž?▲?yž????□???žf?uG?↑x????-!!????km?←M?ž?^?I|????70?g*????Xoc?w+?J???G?τ?k???;3?m?<?b?
```

# 비즈니스 요구사항

- 데이터

- 1) 고객 (id, 이름, email, password, 멤버십 유무) **ok**
- 2) 영상 (id, 등록날짜) **ok**

- 기본 기능

- 1) 로그인 **ok**
- 2) 파일 등록 **ok**
- 3) 파일 목록 조회 **ok**
- 4) 파일 상세 **ok**

Block simultaneous access  
(invalidate existing user)



# Invalidate existing session already used

- User is not allowed to login twice with one account.
- Blocking simultaneous access of login.
- Invalidate existing session when user login again with same account from different device or different IP address.

## 2 way to block simultaneous login

1. By using Spring framework function "Spring-Security"
2. By using Servlet interface "HttpSessionListner"  
(HttpSessionBindingListener)

# HttpSessionListener

- One of the interface of "Servlet"
- kind of event listener.
- It is listening the event of HttpSession
- When specific HttpSession method called, its method will be activated

```
package jakarta.servlet.http;

import java.util.EventListener;

Implementations of this interface are notified of changes to the list of active sessions in
To receive notification events, the implementation class must be configured in the depl
for the web application.

Since: Servlet 2.3
See Also: HttpSessionEvent

public interface HttpSessionListener extends EventListener {

    Notification that a session was created. The default implementation is a NO-OP.
    Params: se – the notification event

    public default void sessionCreated(HttpSessionEvent se) {}

    Notification that a session is about to be invalidated. The default implementation is
    Params: se – the notification event

    public default void sessionDestroyed(HttpSessionEvent se) {
    }
}
```

# Wait... What is event and listener?

- Event: 마우스 클릭, 키보드 입력, 클라이언트로부터의 HTTP 요청, 특정 객체의 생성, 소멸, 웹어플리케이션 시작, 웹어플리케이션 종료 등등
  - 1.**Servlet context-level (application-level) event:** This event concerns resources or states held at the appliance servlet context object's extent.
  - 2.**Session-level event:** It involves resources or states associated with a sequence of requests from a single user session; in other words, it is associated with the HTTP session object.
- Listener: 이벤트가 발생되기를 기다렸다가 발생시 실행되는 메서드나 함수. 또는 메서드를 가진 객체

# Example of implementation of HttpSessionListener

- Beadung example
- The session listener will be triggered when the session is created and destroyed

```
HttpSession session = request.getSession();
```

create

destory

```
session.invalidate();
```

```
public class SessionListenerWithMetrics implements HttpSessionListener {  
  
    private final AtomicInteger activeSessions;  
  
    public SessionListenerWithMetrics() {  
        super();  
  
        activeSessions = new AtomicInteger();  
    }  
  
    public int getTotalActiveSession() {  
        return activeSessions.get();  
    }  
  
    public void sessionCreated(final HttpSessionEvent event) {  
        activeSessions.incrementAndGet();  
    }  
  
    public void sessionDestroyed(final HttpSessionEvent event) {  
        activeSessions.decrementAndGet();  
    }  
}
```

# Other types of listener related to session

- HttpSessionListener – triggered when session is created or destroyed
- HttpSessionAttributeListener – triggered when session attribute get new value or replace it or remove it
- HttpSessionBindingListener – triggered when object is put into session attribute
- HttpSessionActivationListener – triggered when session get to be active or passive

# HttpSessionAttribute vs HttpSessionBinding

- HttpSessionAttributeListener : 능동적 입장의 리스너. Session에 아무 attribute가 들어오고 나올 때 작동된다. Web.xml의 listener 태그를 설정하던가, 클래스 파일에 @WebListener를 설정해야함.
- HttpSessionBindingListener: 수동적 입장의 리스너. 특정 객체가 들어오고 나올 때 작동된다. 별도의 설정 필요 없으나, 해당 클래스의 객체를 직접 구현해서 세션에 넣어야 한다.
- Binding Listener와 Activation Listener는 컨테이너에 등록할 필요 X

# Where is the session stored?

## 1. 톰캣 내장메모리 session 사용

: session은 톰캣의 내장 메모리에 저장되므로 서버를 재시작할 때마다 세션이 초기화된다는 특징이 있다. 보통 1대의 WAS를 사용하는 프로젝트에서 주로 사용하는 방식이다. (2대 이상의 WAS를 사용할 경우 추가적인 설정 필요)

## 2. DBMS에 session 저장

: 여러 WAS에서 공용으로 세션을 사용할 수 있다. 로그인/로그아웃 때마다 DB I/O가 발생하여 성능에 영향을 준다는 단점이 있다. 로그인/로그아웃 요청이 많지 않은 프로젝트에서 주로 사용된다.

## 3. Redis, Elastic cache등의 메모리 DB에 session 저장

: 가장 많이 사용되는 방식이다. 실제 서비스로 배포하려면 embedded redis 방식이 아닌 외부 메모리 서버를 구축하여 사용해야 한다.



# Implements HttpSessionListener

- Collect session id in static memory using hashmap data structure.

```
@WebListener
public class WebSessionListener implements HttpSessionListener {
    private static final Map<String, HttpSession> sessions = new Co

    @Override
    public void sessionCreated(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        String sessionId = session.getId();
        log.info("sessionId={}", sessionId);
        sessions.put(sessionId, session);
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        String sessionId = session.getId();
        if (sessionId != null){
            log.info("session invalidate");
            sessions.remove(sessionId);
        }
    }
}
```

# [Error] No log... why?

- No log for sessionCreated.
- Maybe container couldn't find the sessionListener

```
@WebListener  
public class WebSessionListener  
    private static final Map<Str
```

```
@Override  
public void sessionCreated(HttpSessionEvent se) {  
    HttpSession session = se.getSession();  
    String sessionId = session.getId();  
    log.info("sessionId={}", sessionId);  
    sessions.put(sessionId, session);  
}
```

```
: email=sionwer5@gmail.com, password=123123  
: select m1_0.id,m1_0.email,m1_0.membership,m1_0.na  
: login success
```

# [Error] No log... why?

- Use @Component instead of @WebListener
- @WebListener doesn't contain @Component. That is servlet annotation, handled by Servlet Container.

```
@Slf4j
@Component
public class WebSessionListener implements HttpSessionListener {
    private static final Map<String, HttpSession> sessions = new HashMap<>();

    @Override
    public void sessionCreated(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        String sessionId = session.getId();
        log.info("sessionId={}", sessionId);
        sessions.put(sessionId, session);
    }
}
```

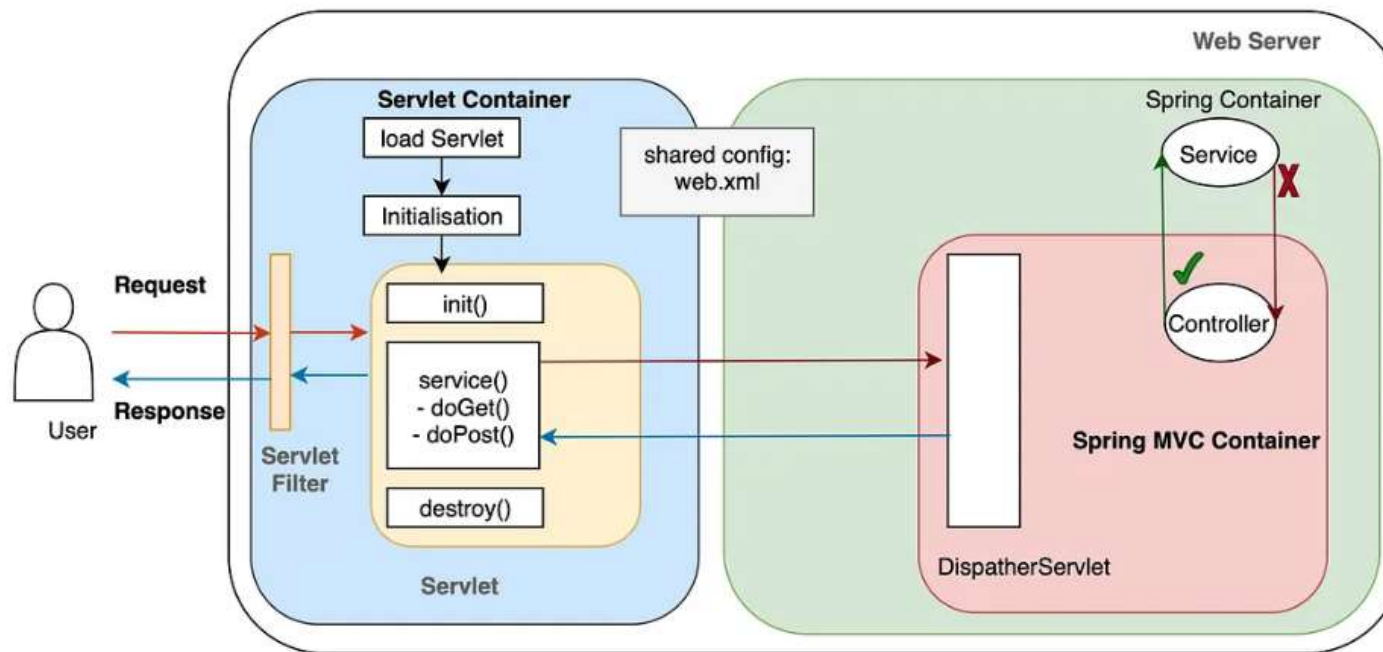
```
: email=signme@gmail.com, password=123456
: select m1_0.id,m1_0.email,m1_0.membership,m1_0
: login success
: sessionId=F38E54DC0AF00C79CFE1FE9F72727109
```

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebListener {

    Returns: description of the listener, if present
    String value() default "";
}
```

# Q. Servlet Container is not working simultaneously?

- I thought Servlet Container is always working with Spring container simultaneously.



# Implements HttpSessionListener

- Collect session id in static memory using hashmap data structure.

```
private static final Map<String, HttpSession> sessions = new HashMap<>();
```

```
@Override
public void sessionCreated(HttpSessionEvent se) {
    HttpSession session = se.getSession();
    String sessionId = session.getId();
    sessionCheck(session);
    sessions.put(sessionId, session);
    log.info("sessionId={} session={}", sessionId, session);
}
```

```
public void sessionCheck(HttpSession session){
    for(String sessionId: sessions.keySet()){
        HttpSession sessionPrevious = sessions.get(sessionId);
        if(session.equals(sessionPrevious)){
            log.info("invalidate session of duplicate access");
            sessionPrevious.invalidate();
        }
    }
}
```

# Implements HttpSessionListener

- Session.invalidate doesn't work.
- this isn't right method. Because value of sessions(HttpSession) is always different.

```
: email=sionwer5@gmail.com, password=123123
: select m1_0.id,m1_0.email,m1_0.membership,m1_0.name,m1_0.password from member m1_0 where m1_0.email=?
: login success
: sessionId=F7B97CF5525A5FD76FA6BD0D3A102B1A session=org.apache.catalina.session.StandardSessionFacade@56655abc
: email=sionwer5@gmail.com, password=123123
: select m1_0.id,m1_0.email,m1_0.membership,m1_0.name,m1_0.password from member m1_0 where m1_0.email=?
: login success|
: sessionId=81EB8635DD6A7C24F8B18D4E79A5BD23 session=org.apache.catalina.session.StandardSessionFacade@230e8b8d
```

# Use sessionAttributeListener

- we need an attribute listener because `getAttribute()` plays a role of identifying logged in user.

```
@Slf4j
@Component
public class WebSessionAttributeListener implements HttpSessionAttributeListener {
    private static final Map<String, Member> sessions = new HashMap<>();

    @Override
    public void attributeAdded(HttpSessionBindingEvent se) {

    }

    @Override
    public void attributeRemoved(HttpSessionBindingEvent se) {

    }

    @Override
    public void attributeReplaced(HttpSessionBindingEvent se) {
```



# Use both Listener

- sessionAttributeListener

```
@Component
public class WebSessionAttributeListener implements HttpSessionAttributeListener {
    private static final Map<Member, HttpSession> sessions = new HashMap<>();

    @Override
    public void attributeAdded(HttpSessionBindingEvent se) {
        HttpSession session = se.getSession();
        Member loginMember = (Member)session.getAttribute(Const.LOGIN_MEMBER);
        log.info("stored member={} with session={} ", loginMember, session);
        deletePreviousSession(loginMember);
        sessions.put(loginMember, session);
    }

    public void deletePreviousSession(Member loginMember){
        if (sessions.containsKey(loginMember)){
            HttpSession sessionPrevious = sessions.get(loginMember);
            sessionPrevious.invalidate();
            log.info("invalidate previous session of multiple access");
        }
    }
}
```



# Use both Listener

- sessionListener
- Remove value from sessions memory storage in case of logout

```
@Slf4j
@Component
public class WebSessionListener implements HttpSessionListener {
    private static final Map<String, Member> sessions = new HashMap<>();

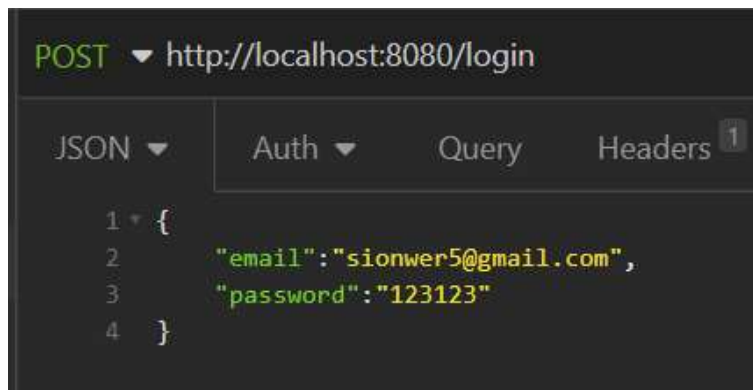
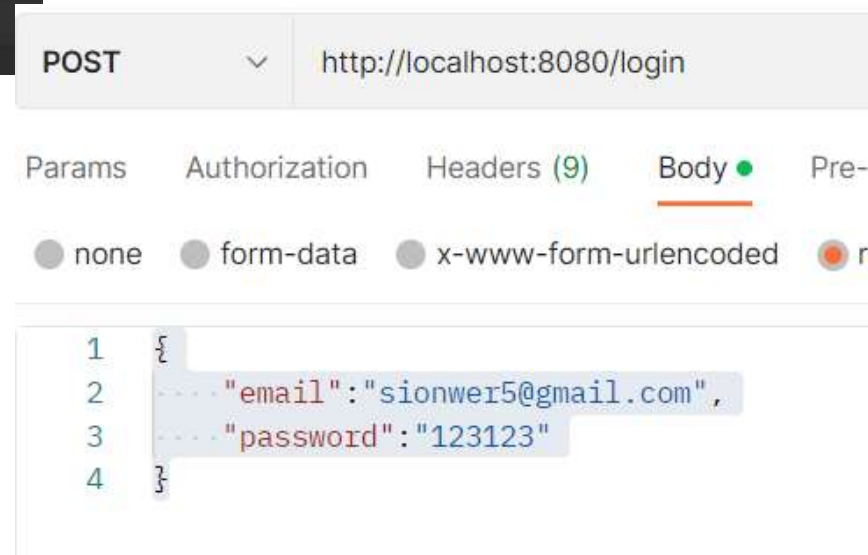
    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        String sessionId = session.getId();
        log.info("session invalidate, remove member from memory storage");
        sessions.remove(sessionId);
    }
}
```

# Check code working

- Check log by using iteration to show all keys and values of 'sessions' hashmap.

```
for(Member key: sessions.keySet()){  
    HttpSession value = sessions.get(key);  
    log.info("sessions storage key={} value={}", key, value);  
}
```

postman



insomnia

# Check code working

- Identical user, but different session

```
login success
stored member=Member(id=1, name=season, email=sionwer5@gmail.com, password=123123)
sessions storage key=Member(id=1, name=season, email=sionwer5@gmail.com, password=123123)
email=sionwer5@gmail.com, password=123123
select m1_0.id,m1_0.email,m1_0.membership,m1_0.name,m1_0.password from member m1_0 where m1_0.id=1
login success
stored member=Member(id=1, name=season, email=sionwer5@gmail.com, password=123123)
session invalidate, remove member from memory storage
invalidate previous session of multiple access
sessions storage key=Member(id=1, name=season, email=sionwer5@gmail.com, password=123123)
org.apache.catalina.session.StandardSessionFacade@7f370035
p=false) value=org.apache.catalina.session.StandardSessionFacade@7f370035
with session=org.apache.catalina.session.StandardSessionFacade@6c24f3dd
p=false) value=org.apache.catalina.session.StandardSessionFacade@6c24f3dd
```

# Check code working

- Logout is also working

```
: stored member=Member(id=1, name=season, email=sionwer5@  
: certification activated. access to URI=/logout  
: session invalidate, remove member from memory storage  
: logout success  
: no user info
```

```
@Override  
public void sessionDestroyed(HttpSessionEvent se) {  
    HttpSession session = se.getSession();  
    String sessionId = session.getId();  
    log.info("session invalidate, remove member from memory storage");  
    sessions.remove(sessionId);  
}
```