[API레퍼런스] TICON_TICON

<JS파일 요약>

user.js	/signup : 페이스북 계정으로 어플리케이션 로그인
	/check : 회원 정보 갱신, 관리
package.js	/all : 현재 시간 기준 인기 이모티콘 순위 안내
	/main : 최신 이모티콘 패키지 5개 안내
	/new : 최신 이모티콘 패키지 목록
	/free : 무료 이모티콘 패키지 인기 순위
	/cost : 유료 이모티콘 패키지 인기 순위
	/myemoticon : 사용자가 만든 이모티콘 패키지 목록
	/mydownloaded : 사용자가 다운받은 이모티콘 패키지 목록
	/category : 카테고리별 이모티콘 패키지 목록
	/detail : 이모티콘 패키지 내 상세 정보 안내
	/download : 사용자의 코인과 이모티콘 가격을 비교하여 구매결정
image.js	/upload : AWS에 이모티콘 패키지 별 이미지 업로드
	/delete : 사용자가 만든 이모티콘 패키지 삭제
	/manager : 매니저 웹을 서버에 렌더링, 매니저웹에서 이모티콘 보러가기 버튼 클릭 시 매니저웹에서 공개여부 수정

<package.js>

① Get '/package/all '

: 메인 화면에서 현재 시간 기준 가장 다운로드 수가 높은 이모티콘 패키지를 안내합니다.

query

```
select user.user_name, package.pkg_name, package.pkg_id, image.image_url
from user, package, image
where user.user_id = package.user_id_fk and
        package.pkg_id = image.pkg_id_fk and pkg_enable=1
group by pkg_id
order by download_cnt desc
```

- query parameter 없이 메인 화면에 다운로드 수(download_cnt)가 제일 높은 pkg_name, 해당 패키지 작성 user_name, 패키지 대표 img_url 출력

result

User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다.
Pkg_name : <string> - 이모티콘 패키지의 이름입니다.
Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다.
Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

② **Get** **'/package/main'** : 메인 화면에서 5개의 최신 이모티콘 패키지를 안내합니다

query	1) select p.pkg_id, u.user_name, p.pkg_name from user u, package p where u.user_id=p.user_id_fk and pkg_enable=1 group by pkg_id order by pkg_id desc; - query parameter 없이 메인 화면에 이모티콘 pkg_name, user_name을 추출합니다. 2) select i.image_url, p.pkg_id from image i, package p where p.pkg_id=i.pkg_id_fk and p.pkg_enable =1 order by pkg_id desc ; - 중첩 connection query문을 통해 앞에서 추출한 이모티콘 패키지의 img_url을 추출합니다..
result	User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다. Pkg_name : <string> - 이모티콘 패키지의 이름입니다. Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다. Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

③ **Get** **'/package/new'** : 새롭게 등록된 이모티콘 패키지를 최신 순서대로 모두 안내

query	select image_url, package.pkg_id, user.user_name, package.pkg_name from user, package, image where user.user_id=package.user_id_fk and image.pkg_id_fk=package.pkg_id and pkg_enable=1

	group by pkg_id order by package.pkg_time desc; Query parameter 없이 새롭게 등록된 이모티콘 패키지를 공개여부(pkg_enable)를 판단하여 이모티콘 업로드 시간(pkg_time)에 따라 정렬 - Pkg_name, user_name, 패키지의 첫번째 img_url을 추출합니다.
result	User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다. Pkg_name : <string> - 이모티콘 패키지의 이름입니다. Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다. Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

④ Get **'/package/free'** : 무료 이모티콘을 다운로드 수가 높은 순서대로 목록을 나열

query	select image.image_url, package.pkg_id, user.user_name, package.pkg_name from user, package, image where user.user_id=package.user_id_fk and image.pkg_id_fk = package.pkg_id and pkg_cost=0 and pkg_enable = 1 group by pkg_id order by download_cnt desc - query parameter 없이 이모티콘 pkg_name, user_name, 패키지의 대표 img_url을 추출합니다.

- Pkg_cost = 0이고 Download_cnt 내림차순으로 정렬

result

User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다.
Pkg_name : <string> - 이모티콘 패키지의 이름입니다.
Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다.
Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

⑤ **Get** **'/package/cost'** : 유료 이모티콘을 다운로드가 높은 순서대로 목록을 나열

query

```
select image.image_url, package.pkg_id, user.user_name,
       package.pkg_name
from user, package, image
where user.user_id=package.user_id_fk and
      image.pkg_id_fk = package.pkg_id and pkg_cost<>0
      and pkg_enable = 1
group by package.pkg_id
order by download_cnt desc
```

- query parameter 없이 이모티콘 pkg_name, user_name, 패키지의 대표 img_url을 추출합니다.
- Pkg_cost <>0이고 Download_cnt 내림차순으로 정렬

result

User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다.
Pkg_name : <string> - 이모티콘 패키지의 이름입니다.
Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다.

Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

⑥ **Get** **'/package/{user_id}/myemoticon'** : 사용자가 만든 이모티콘을 확인.

query

```
select i.image_url, p.pkg_id, p.pkg_name, u.user_name
from user u, package p, image i
where u.user_id=p.user_id_fk and p.pkg_id=i.pkg_id_fk and user_id=?
group by pkg_id;
```

- Query parameter는 user_id로써 user_id로 사용자를 식별함으로써 해당 사용자의 pkg_name, user_name, 패키지 대표 img_url을 추출합니다.

result

User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다.
Pkg_name : <string> - 이모티콘 패키지의 이름입니다.
Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다.
Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

⑦ Get '/package/{user_id}/mydownloaded'

:사용자가 다운받은 이모티콘을 확인할 수 있습니다.

query

```
select i.image_url, p.pkg_id, p.pkg_name, u.user_name
from user u, package p, download d, image i
where d.pkg_id_fk = p.pkg_id and p.pkg_id = i.pkg_id_fk
        and p.user_id_fk = u.user_id and d.user_id_fk=?
group by pkg_id
order by d.download_id desc;
```

- Query parameter는 user_id로써 user_id로 사용자를 식별함으로써 다운로드 테이블에 적재된 해당 사용자의 pkg_name, 패키지 작성자 user_name, 패키지 대표 img_url을 추출합니다.

result

User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다.
Pkg_name : <string> - 이모티콘 패키지의 이름입니다.
Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다.
Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

⑧ Get '/package/{category_id}/category'

: 카테고리 종류 별로 분류된 이모티콘을 확인할 수 있습니다.

query

```
select p.pkg_name, u.user_name, i.image_url, p.pkg_id
from package p, user u, image i
where p.user_id_fk = u.user_id and p.pkg_id = i.pkg_id_fk
        and p.category_id_fk =? and p.pkg_enable =1
group by pkg_id;
```

- Query parameter는 category_id로써, 각기 다른 종류의 이모티콘 패키지를 분류하여 이모티콘 pkg_name, 패키지 작성자 user_name, 패키지의 대표 img_url을 추출합니다.

result

User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다.
Pkg_name : <string> - 이모티콘 패키지의 이름입니다.
Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다.
Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.

⑨ Get '/package/detail/{user_id}/{pkg_id}'
: 이모티콘 패키지의 상세 정보를 확인할 수 있습니다.

query

1) select * from download where user_id_fk =? and pkg_id_fk = ?
   - query parameter로 user_id와 pkg_id를 받아 기존 다운로드 여부를 확인합니다.
- 변수 already = 0 를 선언하여 기존에 동일 이모티콘 패키지를 다운 받았을 경우 1로 전환합니다.

2) select pkg_name, user_id, user_name, category_id, pkg_cost
   from package, user, category
   where package.user_id_fk = user.user_id
   and package.category_id_fk = category.category_id
   and package.pkg_id=?
- Query parameter로 pkg_id를 받아 패키지의 상세 정보인 pkg_name, user_id, user_name, category_id, pkg_cost를 추출합니다.

3) select image_url from image, package
where image.pkg_id_fk = package.pkg_id and pkg_id=?

- Query parameter로 pkg_id를 받아 해당 패키지의 모든 이미지 url을 추출합니다.

4) select image_url, package.pkg_id
from image, package, user
where package.user_id_fk = user.user_id and package.pkg_id = image.pkg_id_fk and user.user_id=? and package.pkg_enable = 1
group by pkg_id

- 위 쿼리문에서 추출한 이모티콘 패키지의 모든 상세정보를 출력합니다. 상세정보는 user_id, pkg_name, user_name, category_id, pkg_cost, img_url입니다.

result

User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름입니다.
Pkg_name : <string> - 이모티콘 패키지의 이름입니다.
Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다.
Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다.
Pkg_cost : <int> - 이모티콘 패키지 가격
Category_type : <string> - 이모티콘 패키지 카테고리

⑩ Post '/package/download'

: 1. 사용자의 코인과 이모티콘 코인 가격을 비교하여 구매 가능여부를 판단하여 이모티콘을 구매합니다. 구매 후에는 사용자의 코인이 차감됩니다.

2. 사용자가 만든 이모티콘이 다른 사용자에게 다운로드 될 시 해당 이모티콘 패키지 가격만큼의 코인을 추가로 부여합니다.

query

1. 이모티콘 구매

(1) select user_coin from user where user_id=?

: query parameter로 user_id를 이용하여 사용자의 현재 코인을 추출.

(2) select pkg_cost from package where pkg_id=?

: query parameter로 pkg_id를 이용하여 패키지의 가격(pkg_cost)을 불러옵니다. 사용자의 현재 코인과 비교하여 구매할 수 있을 경우 사용자의 user_coin을 삭감합니다.

(3) update user set user_coin=? where user_id=?

: query parameter로 user_id를 불러와 해당 사용자의 user_coin을 update합니다.

2. 패키지 다운로드 시 패키지 업로드 사용자 user_coin 증가

(1) select p.user_id_fk, p.pkg_cost
 from user u, package p where u.user_id = p.user_id_fk and pkg_id=?;

: query parameter로 pkg_id를 불러와 pkg_cost와 해당 이모티콘 패키지를 업로드한 user_id가 추출됩니다.

(2) update user set user_coin = user_coin + ? where user_id=?

: query parameter로 user_id를 불러와 다운로드가 된 이모티콘 패키지 업로드 user의 user_coin을 해당 패키지 가격만큼 더 부여합니다.

(3) INSERT INTO `download` (`user_id_fk`, `pkg_id_fk`, `download_time`) VALUES (?, ?,?);

: download 테이블에 새롭게 insert합니다.

result

Result : <Boolean> - true or false로 이모티콘 패키지의 업로드 성공 여부 반환

	Reason : <string> - 업로드 오류 구간에 오류 실패 이유를 반환. User_name : <string> - 이모티콘 패키지를 등록한 사용자의 이름. Pkg_name : <string> - 이모티콘 패키지의 이름.. Pkg_id : <int> - 이모티콘 패키지 별 고유 식별 번호입니다. Image_url : <string> - 이모티콘 패키지의 개별 이미지 주소입니다. User_coin : <int> - 사용자 보유 코인.

<user.js>

⑪ **get** **'/user/{user_id}/check** ': 회원 정보를 관리하고 갱신합니다**.**

query	select * from user where user_id=?; - Query parameter로 user_id 값을 불러와 user테이블 속 해당 user의 모든 정보를 추출.
result	user_id : <int> - 사용자에게 부여하는 고유 user_id user_email : <string> - 페이스북에 로그인 된 사용자 이메일 주소 user_name : <string> - 사용자 이름 user_facebook_id : <int> - 페이스북에서 사용자에게 부여하는 고유 번호 user_photo : <string> - 사용자의 페이스북 프로필 사진 이미지 url user_coin : <int> - 사용자 보유 코인

⑫ **post** **'/user/signup'** : 회원 정보를 관리합니다. 신규 회원에게 고유 번호를 부여하고 페이스북에서 얻어온 사용자 이름, 이메일, 페이스북 고유 id, 사용자 프로필 사진, 사용자 보유 코인을 저장합니다. 기존 회원일 경우 저장된 정보를 불러옵니다.

query	(1) select * from user where user_facebook_id = ?; : query parameter로 user_facebook_id를 얻어와 user테이블에 해당 user_facebook_id 존재 유무를 확인. (1)-1 insert into user (user_name, user_photo, user_facebook_id, user_email,user_coin) values (?, ?, ?,?,5000) : 기존 사용자가 아닐 경우 사용자 정보를 새롭게 user테이블에 추가. (1)-2 select * from user where user_id = : 기존 사용자일 경우 query parameter로 user_id를 불러와 기존 user 테이블에 저장된 사용자 정보를 출력한다.
result	User_id : <int> 어플 사용자에게 부여되는 고유 id User_name : <string> - 페이스북에 로그인 된 사용자의 이름입니다. User_photo : <string> - 페이스북 프로필 사진 이미지 url User_facebook_id : <int> - 페이스북에서 사용자에게 부여하는 고유 번호 User_email : <string> - 페이스북에 로그인 된 사용자 이메일 주소 User_coin : <int> - 사용자 보유 코인

<image.js>

⑬ **Post** **'/image/upload'** : 패키지를 업로드 하는 포스트 통신 (s3 이미지 업로드 포함)

Flow + Query	Formidable 라이브러리를 이용 Form.parse 함수를 통해 Fields와 files를 전송 Fields에는 각종 객체 데이터를 담을 수 있어서 작성자 정보, 패키지 이름등의 상세 정보를 입력 mysql Connection과 연결해 DB갱신 Files에는 이미지파일들의 정보를 담아 전송 S3의 파라미터 값으로 이미지 fs.createReadStream을 body에 담아 파일패스를 전송 bucket이름과 파일이름을 동적으로 형성하여 전달

S3에 정상적으로 파일이 업로드되면 임시이미지파일을 삭제시켜서 ec2 서버의 저장공간을 낭비하지 않음.(fs.unlink 사용)
업로드됨과 동시에 동적으로 형성된 s3파일주소를 DB 이미지 테이블에 INSERT.
>삽입문
insert into package(user_id_fk, pkg_name, pkg_enable, pkg_cost, category_id_fk, pkg_time)

- User_id_fk : 작성자 이름, pkg_name : 패키지명, pkg_enable : 공개여부, pkg_cost : 패키지가격 , category_id_fk: 카테고리id값, pkg_time :업로드시간

>S3 파라미터값
Bucket : bucketname,// 버킷이름
Key : 'emoticon/' + newFileName, // 동적으로 형성된 주소값

>최종 이미지주소 디비업데이트
'http://' + s3.endpoint.host + '/' + bucketname + '/' + params.Key;
위의 주소값을 image_url값에 넣고 원래 받아왔던 pkg_id값을 포함해
INSERT INTO `image` (`image_url`, `pkg_id_fk`) 실행

result

Result : <Boolean> - true or false를 반환해 이미지업로드에 성공여부 리턴
Reason : <Strng> - 업로드에 실패한 오류구간을 리턴

⑭ **Post** **'/image/delete'** : 패키지를 Delete 하는 포스트 통신 (s3 데이터 삭제 포함)

Flow + Query	DB상에 package를 delete할때 cascade로 설정해서
	참조된 다른 테이블도 삭제가 되는 형태
	- download, package , image table, 3테이블 동시 삭제진행
	Delete from package where pkg_id =?
	- pkg_id값을 받아오면 해당 쿼리를 실행 // 실패시 원인 리턴
	- 삭제문이 성공적으로 실행된다면 S3저장소의 이미지 삭제 실행
	
	>S3 파라미터값
	var newFileName = req.body.pkg_id+ "_" + i +".png"
	{
	Bucket : bucketname,// 버킷이름
	Key : 'emoticon/' + newFileName, // 동적으로 형성된 주소값
	}
	s3.deleteObject 함수를 이용 저장소에 저장된 데이터를 삭제.

result	Result : <Boolean> - true or false를 반환해 삭제문의 성공여부 리턴 Reason : <Strng> - 삭제문 실패한 오류구간, 원인을 리턴

⑮ **GET** **'/manager'** : 메니저 웹을 서버에 렌더링

Flow + Query	select image_url, package.pkg_id, user.user_name, package.pkg_name, package.pkg_enable, package.pkg_time, package.download_cnt from user, package, image where user.user_id=package.user_id_fk and

	image.pkg_id_fk=package.pkg_id group by pkg_id order by package.pkg_id desc - 관리자 페이지이기때문에 패키지 공개여부에 상관없이 모든 페키지 데이터와 이미지 데이터를 반환해줌. - ejs템플릿을 사용해서 해당 데이터들을 이용해 동적으로 테이블 형성. Manager.ejs에 렌더링 {
result	성공시 manager.ejs에 렌더링 // 실패시 Status 500을 리턴 // Result: <Boolean> true or false를 리턴 Reason: <string> 실패구간의 실패 이유를 리턴!

(16) GET **'/manager/:pkg_id'** 매니저웹에서 이모티콘 보러가기 버튼 클릭시

Flow + Query	select image_url from image where pkg_id_fk = ? - 해당 pkg 아이디값을 제이쿼리를 통해 가져옵니다. - Pkg와 관련된 img URL데이터를 ejs에 뿌려줍니다
result	성공시 image.ejs에 이미지를 렌더링 // 실패시 Status 503을 리턴 // Result: <Boolean> true or false를 리턴

(17) POST /manager 매니저웹에서 공개여부 수정

Flow **+** **Query**	update package set pkg_enable= ? where pkg_id=? - pkg_enable =공개여부(0 or 1) , pkg_id = 페키지 아이디
result	성공시 공개여부를 수정! 실패시 Status 503을 리턴 // Result: <Boolean> true or false를 리턴