8

2024 GETIT 5기 SW 교육(리액트) 8차시

백과 함께 작업하기

GETIT 5기 운영진 이은새

참고사항

해당 강의자료는 MacOS를 기준으로 제작되었습니다.

도움이 필요하거나 질의사항이 있으시다면
이은새 운영진에게 개인적으로 연락하거나
카카오톡 오픈채팅을 이용해주세요!
GETIT 5기 질문방 : https://open.kakao.com/o/gc7jPrhg

해당 강의자료는 [이인제-소플의 처음 만난 리액트]를 참고하여 제작되었습니다.
5기 부원 외의 강의자료 및 영상 배포는 원칙적으로 금지합니다.

목차

01

axios

1

axios

- 브라우저와 Node.js 환경에서 사용되는 HTTP 클라이언트 라이브러리
- 특징
  - Promise를 사용하여 비동기 작업을 처리
  - HTTP 요청을 보내는 것이 쉬움. 간단하고 직관적임.
  - 요청 또는 응답을 가로채고 수정할 수 있는 인터셉터를 제공
  - 에러 처리 기능, 취소 요청 기능

2

axios 설치 및 사용

- 설치
  - npm install axios --save
- 가져오기
  - import axios from "axios";
- 사용
  - axios(config)

3

axios의 주요 요청 메서드

- axios에는 다양한 요청 메서드가 있습니다.
- 주요 요청 메서드
  - GET : axios.get()
  - POST : axios.post()
  - PUT : axios.put()
  - DELETE : axios.delete()

4

GET 요청

- axios.get(url)
- 서버에서 데이터를 가져올 때 사용

```
async getAllPosts() {
  try {
    return await axios.get(`${BASE_URL}/getAll`);
  } catch (err) {
    console.error(err);
    throw err;
  }
}
```

5

POST 요청

- axios.post(url, data)
- 서버에 새 데이터를 보낼 때 사용

```
async createPost(data) {
  try {
    return await axios.post(`${BASE_URL}/postCreate`, data);
  } catch (err) {
    console.error(err);
    throw err;
  }
}
```

02

서버에 요청 보내고 응답 받기

service.js

- 요청과 관련된 모든 작업을 service.js에 있는 하나의 클래스 (API)에서 관리하도록 하겠습니다.

```
import axios from "axios";

const BASE_URL = process.env.REACT_APP_API_URL;

class API {…
}

export default new API();
```

2

.env

- .env는 환경 변수를 설정하기 위해 사용되는 파일입니다.
- API 키나 데이터베이스 연결 정보 같은 민감한 정보를 저장함.
- process.env를 통해서 접근함

```
REACT_APP_API_URL=http://192.168.0.6:8080
```

```
import axios from "axios";

const BASE_URL = process.env.REACT_APP_API_URL;

class API {
  async getAllPosts() {
    try {
      return await axios.get(`${BASE_URL}/getAll`);
    } catch (err) {
```

3

API 명세서를 바탕으로 코드 작성

- 이해하기 쉬운 형태의 간단한 명세서입니다.

내용	요청 메서드	엔드포인트	요청 데이터	응답 데이터
모든 게시물 조회	GET	/getAll		[posts]
특정 게시물 조회	GET	/getOne	postId	{title, description}
게시물 추가	POST	/postCreate	{title, description}	
게시물 수정	POST	/postUpdate	{title, description}	
게시물 삭제	POST	/postDelete	{postId}	

모든 게시물 조회 – getAllPosts

service.js

```
class API {
  async getAllPosts() {
    try {
      return await axios.get(`${BASE_URL}/getAll`);
    } catch (err) {
      console.error(err);
      throw err;
    }
  }
```

Home

```
useEffect(() => {
  const fetchPosts = async () => {
    try {
      const response = await API.getAllPosts();
      setPosts(response.data);
    } catch (err) {
      console.error("Failed to fetch posts:", err);
    }
  };

  fetchPosts();
}, []);
```

특정 게시물 조회 – getPost

service.js

```
async getPost(postId) {
  try {
    return await axios.get(`${BASE_URL}/getOne/${postId}`);
  } catch (err) {
    console.error(err);
    throw err;
  }
}
```

PostPage

```
const { postId } = useParams();
const [post, setPost] = useState({});
const postIdInt = parseInt(postId);
useEffect(() => {
  const fetchPost = async () => {
    try {
      const response = await API.getPost(postIdInt);
      setPost(response.data);
    } catch (err) {
      console.error("Failed to fetch post:", err);
    }
  };
  fetchPost();
}, [postIdInt]);
```

게시물 추가 – createPost

service.js

```
async createPost(data) {
  try {
    return await axios.post(`${BASE_URL}/postCreate`, data);
  } catch (err) {
    console.error(err);
    throw err;
  }
}
```

CreatePost

```
const createPost = async () => {
  try {
    await API.createPost({ title: title, description: description });
    navigate("/");
  } catch (err) {
    console.error("Failed to create post:", err);
  }
};
```

게시물 수정 – UpdatePost

service.js

```
async updatePost(data) {
  try {
    return await axios.post(`${BASE_URL}/postUpdate`, data);
  } catch (err) {
    console.error(err);
    throw err;
  }
}
```

UpdatePost

```
useEffect(() => {
  const fetchPost = async () => {
    try {
      const response = await API.getPost(postId);
      setTitle(response.data.title);
      setDescription(response.data.description);
    } catch (err) {
      console.error("Failed to fetch post:", err);
    }
  };
  fetchPost();
}, [postId]);
```

```
const updatePost = async () => {
  try {
    await API.updatePost({
      postId,
      title,
      description,
    });
    navigate("/");
  } catch (err) {
    console.error("Failed to update post:", err);
  }
};
```

게시물 삭제 – deletePost

service.js

```
async deletePost(data) {
  try {
    return await axios.post(`${BASE_URL}/postDelete`, data);
  } catch (err) {
    console.error(err);
    throw err;
  }
}
```

DeletePost

```
const deletePost = async () => {
  try {
    await API.deletePost({ postId: postIdInt });
    navigate(-1);
  } catch (err) {
    console.error("Failed to delete post:", err);
  }
};
```