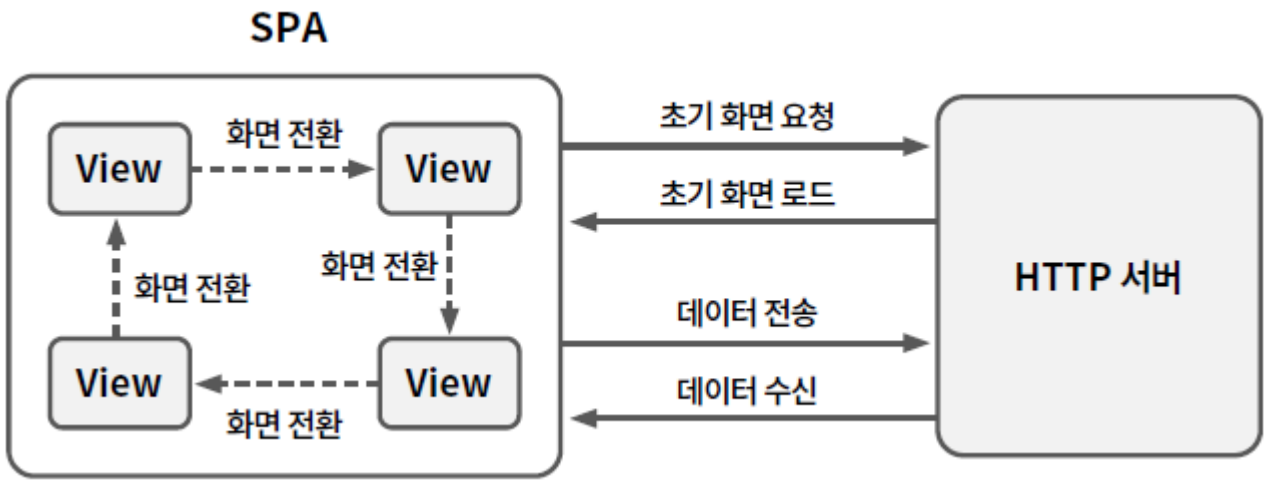
- react-router 따라하기

1. react-router란?

❖React Router란?

- 리액트 기반의 강력한 라우팅 라이브러리
  - URI 경로와 동기화하여 화면의 전환, 흐름을 제어할 수 있도록 하는 기능을 제공
- SPA(Single Page Application) 앱의 핵심 기능 중 하나
  - SPA는 하나의 HTML 페이지 안에서 화면을 전환한다.
  - 따라서 페이지 이동은 일어나지 않음.
  - URI 정보를 근거로 화면을 전환할 수 있는 기능을 React Router가 제공함.

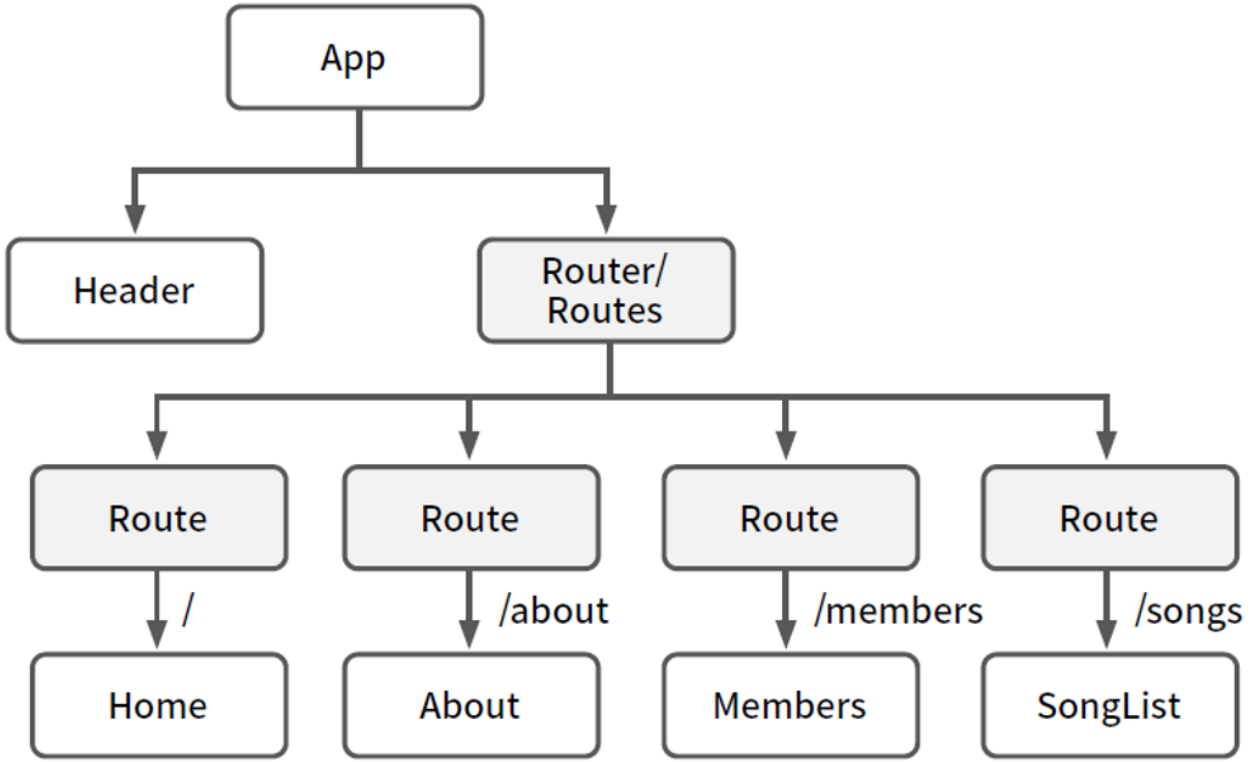



```
npm install  react-router  react-router-dom

//명시적으로 6 버전을 설치하려면 다음과 같이 입력합니다.
npm install  react-router@6.x.x  react-router-dom@6.x.x
```

2. react-router 기본 예제

❖기능 확인을 위한 간단한 프로젝트

- 프로젝트 초기화
  - npm init vite router-test-app -- --template react-ts
  - cd router-test-app
  - npm install bootstrap react-router react-router-dom
- assets 폴더, App.tsx 삭제
- 지원할 라우트 경로



2.1 URI 경로에 대응하여 렌더링을 결정하는 컴포넌트

❖Router
- 자식 컴포넌트로 URI 경로 정보를 처리하는 Routes, Route 컴포넌트를 배치
- 라우팅하는 방법을 결정

❖Routes
- Route 컴포넌트들을 묶어서 배치하는 역할

❖Route
- URI 경로와 렌더링할 컴포넌트나 요소를 지정하는 기능 제공

```
<Router>
  ......
    <Routes>
      <Route path="/" element={ <Home /> }>
      ......
    </Routes>
  ......
</Router>
```

2.2 컴포넌트 작성

❖각 경로별로 보여줄 컴포넌트 작성

- src/pages/Home.tsx작성
- About.tsx, SongList.tsx, Members.tsx 도 src/pages 디렉터리에 작성함.
  - 클래스명과 h2 태그 내부의 문자열만 변경함

```
const Home = () => {
  return (
    <div className="card card-body">
      <h2>Home</h2>
    </div>
  );
};

export default Home;
```

2.2 컴포넌트 작성

- src/components/Header.tsx 작성

```
<Link to={이동시킬 경로}>링크로 보여줄 요소</Link>
```

```
import { Link } from "react-router-dom";

const Header = () => {
  return (
    <div className="card bg-light">
      <div className="card-heading">
        <h2 className="text-center m-3">Foxes And Fossils</h2>
        <p>
          <a href="http://foxesandfossils.com">http://foxesandfossils.com</a>
        </p>
        <div className="row">
          <div className="col-12">
            <Link className="btn btn-success menu" to="/">Home</Link>
            <Link className="btn btn-success menu" to="/about">About</Link>
            <Link className="btn btn-success menu" to="/members">Members</Link>
            <Link className="btn btn-success menu" to="/songs">Songs</Link>
          </div>
        </div>
      </div>
    </div>
  );
};
export default Header;
```

2.2 컴포넌트 작성

- src/App.tsx 작성

```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Header from "./components/Header";
import Home from "./pages/Home";
import About from "./pages/About";
import SongList from "./pages/SongList";
import Members from "./pages/Members";

const App = () => {
  return (
    <Router>
      <div className="container">
        <Header />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/members" element={<Members />} />
          <Route path="/songs" element={<SongList />} />
        </Routes>
      </div>
    </Router>
  );
};

export default App;
```

2.3 index.css, main.tsx 변경

❖src/index.css 변경

```
body { margin: 0; padding: 0; font-family: sans-serif; }
.container {
    text-align: center; margin-top:20px;
}
.menu { width:25%; border-radius:0 !important; }
```

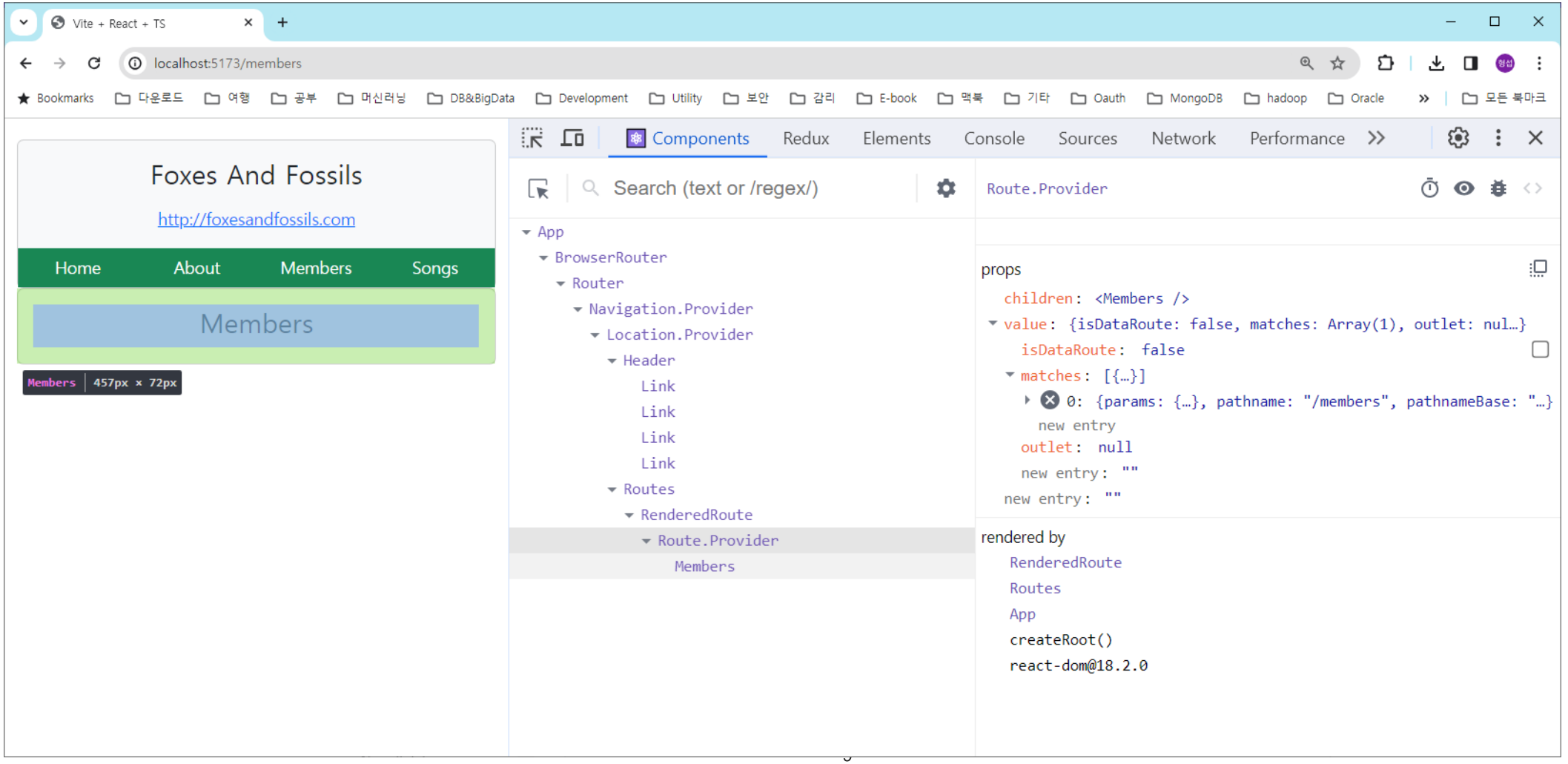
❖src/main.tsx 변경

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import "bootstrap/dist/css/bootstrap.css";
import './index.css'

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

2.4 실행 결과

❖실행 결과

- react devtools의 components 탭으로 Router.Provider 컴포넌트의 value 속성을 반드시 확인!!



3.1 라우팅된 컴포넌트로 속성 전달

❖라우팅되는 컴포넌트에 속성 전달

- element 에 의해 렌더링 되는 컴포넌트로 속성을 전달하면 됨
  - App.tsx 코드 변경

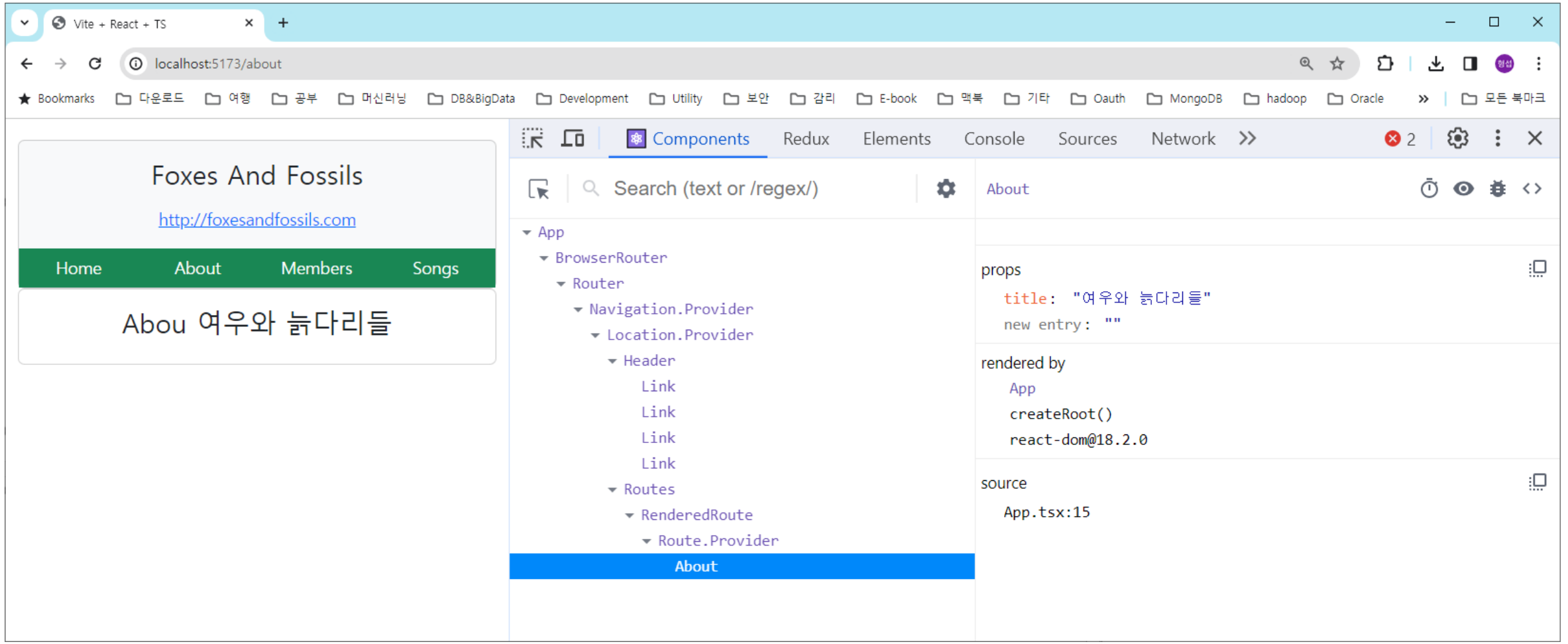
```
……
<Route path="/about" element={<About title={'여우와 늙다리들'} />} />
……
```

- App.js에서 About 컴포넌트로 속성을 전달하도록 코드 변경
  - src/pages/About.tsx

```
type PropsType = {    title: string;    };

const About = (props: PropsType) => {
  return (
    <div className="card card-body">
      <h2>Abou {props.title}</h2>
    </div>
  );
};

export default About;
```

3.1 라우팅된 컴포넌트로 속성 전달

❖ 속성 전달 결과 확인



3.2 복잡한 객체를 속성으로 전달

❖ 복잡한 객체 데이터를 속성으로 전달해보자.

- src/App.tsx 변경

```
import { useState } from "react";
……(중략)
export type MemberType = { name: string; photo: string };

const App = () => {
 const [members] = useState<MemberType[]>([
    { name: "Maggie Adams", photo: "photos/Mag.png" },
    { name: "Sammie Purcell", photo: "photos/Sam.png" },
    { name: "Tim Purcell", photo: "photos/Tim.png" },
    { name: "Scott King", photo: "photos/King.png" },
    { name: "Johnny Pike", photo: "photos/JPike.jpg" },
    { name: "Toby Ruckert", photo: "photos/Toby.jpg" },
  ]);

  return (
       <Router>
            ……(중략)
                 <Route path="/members" element={<Members members={members} />} />
            ……(중략)
       </Router>
   );
};
export default App;
```

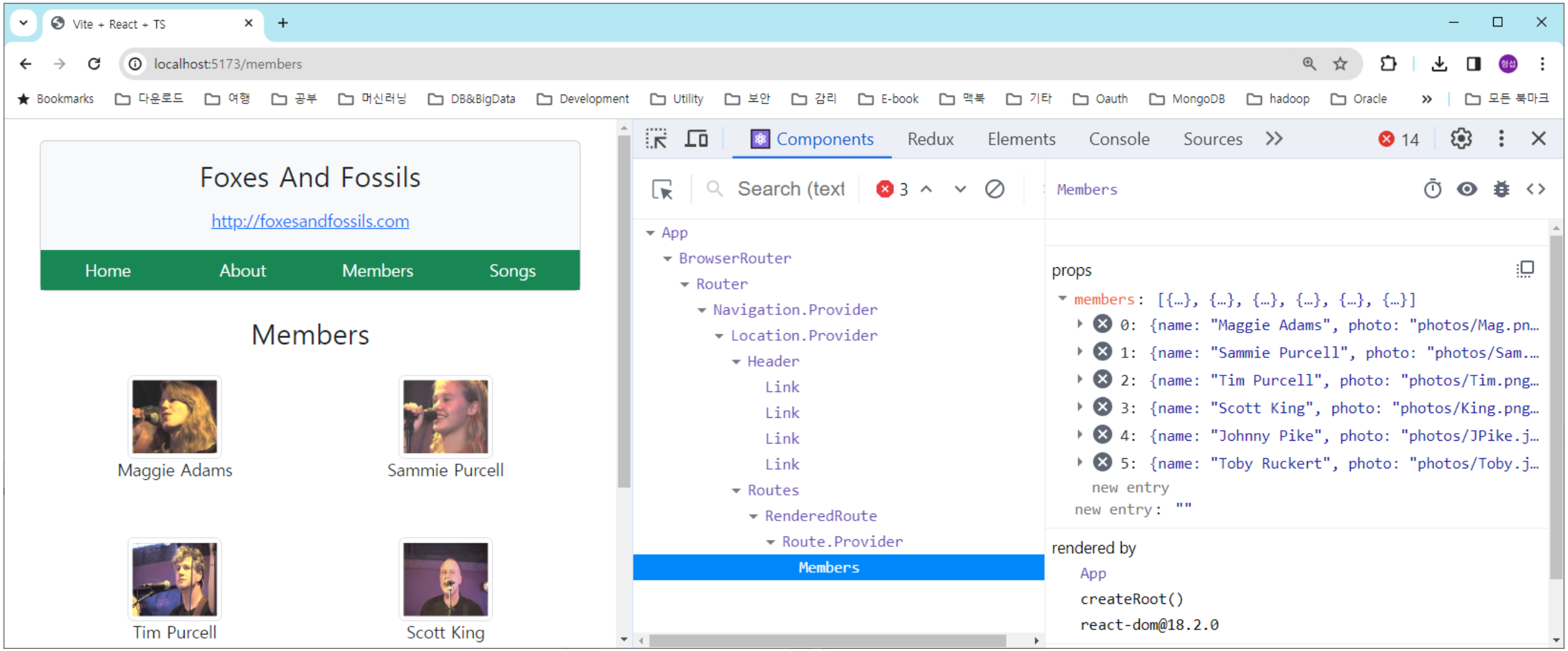
3.2 복잡한 객체를 속성으로 전달

- src/pages/Members.tsx 변경

```
import { MemberType } from "../App";

type PropsType = { members: MemberType[] };

const Members = (props: PropsType) => {
  let imgstyle = { width: 90, height: 80 };
  let list = props.members.map((member) => {
    return (
      <div key={member.name} className="col-6 col-md-4 col-lg-3">
        <img src={member.photo} alt={member.name} className="img-thumbnail" style={imgstyle} /><br />
        <h6>{member.name}</h6><br /><br />
      </div>
    );
  });
  return (
    <div>
      <h2 className="m-4">Members</h2>
      <div className="container">
        <div className="row">{list}</div>
      </div>
    </div>
  );
};
export default Members;
```

3.2 복잡한 객체를 속성으로 전달

❖실행 결과 확인



4.1 URI 파라미터 이용

❖URI 파라미터란?

- URI 경로의 동적인 값을 받아서 이용할 수 있도록 하는 리액터 라우터의 파라미터 사용법

```
<Route path="/songs/:id" element={<SongDetail songs={songs} />} />
```

```
type SongParam = { id: string }

const SongDetail = (……) => {

    const { id } = useParams<SongParam>();

}
```

- 참조
  - /orders/:id/:date
  - /groups/*

4.2 URI 파라미터 적용

❖해야할 일

- App 컴포넌트에 곡(노래 영상) 상태 정보를 추가
- SongList, SongDetail 컴포넌트에 전달하도록 작성
- 곡 정보는 강사로부터 songs.json 전달받아 사용


```
[
   { "id": 1, "title": "Fallin' for you", "musician": "Colbie callet", "youtube_link": "PABUl_EX_hw" },
   { "id": 2, "title": "Can't hurry love", "musician": "The supremes", "youtube_link": "EJDPhjQft04" },
   { "id": 3, "title": "Landslide", "musician": "Dixie chicks", "youtube_link": "V2N7gYom9-A" },
   { "id": 4, "title": "Can't let go", "musician": "Linda ronstadt", "youtube_link": "P-EpGKXmoe4" },
   { "id": 5, "title": "Doctor my eyes", "musician": "Jackson Browne", "youtube_link": "7JlFKS_1oZk" },
   { "id": 6, "title": "We gotta get you a woman", "musician": "Todd Rundgren", "youtube_link": "EyUjbBViAGE" },
   { "id": 7, "title": "Hip to my heart", "musician": "Band Perry", "youtube_link": "vpLCFnD9LFo" },
   { "id": 8, "title": "Rolling in the deep", "musician": "Adele", "youtube_link": "EvK8pDK6IQU" },
]
```

4.2 URI 파라미터 적용

❖src/App.tsx 변경

```
import { useState } from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
......
import SongDetail from './pages/SongDetail';

export type SongType = { id: number; title: string; musician: string; youtube_link: string };

const App = () => {
  const [members] = useState([ ..... ]);

  //songs 상태 추가
  const [songs] = useState<SongType[]>([
    { id: 1, title: "Fallin' for you", musician: "Colbie callet", youtube_link: "PABUl_EX_hw" },
    { id: 2, title: "Can't hurry love", musician: "The supremes", youtube_link: "EJDPhjQft04" },
    { id: 3, title: "Landslide", musician: "Dixie chicks", youtube_link: "V2N7gYom9-A" },
    { id: 4, title: "Can't let go", musician: "Linda ronstadt", youtube_link: "P-EpGKXmoe4" },
    { id: 5, title: "Doctor my eyes", musician: "Jackson Browne", youtube_link: "7JlFKS_1oZk" },
    { id: 6, title: "We gotta get you a woman", musician: "Todd Rundgren", youtube_link: "EyUjbBViAGE" },
    { id: 7, title: "Hip to my heart", musician: "Band Perry", youtube_link: "vpLCFnD9LFo" },
    { id: 8, title: "Rolling in the deep", musician: "Adele", youtube_link: "EvK8pDK6IQU" },
  ]);

(다음 페이지에 이어짐)
```

4.2 URI 파라미터 적용

❖src/App.tsx 변경(이어서)

```
  return (
      <Router>
        <div className="container">
          <Header />
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/about" element={<About title={'여우와 늙다리들'} />} />
            <Route path="/members" element={<Members members={members} />} />
            <Route path="/songs" element={<SongList songs={songs} />} />
            <Route path="/songs/:id" element={<SongDetail songs={songs} />} />
          </Routes>
        </div>
      </Router>
   );
};

export default App;
```

4.2 URI 파라미터 적용

❖src/pages/SongList.tsx 변경

```
import { Link } from "react-router-dom";
import { SongType } from "../App";

type Props = { songs: SongType[] };

const SongList = (props: Props) => {
  let list = props.songs.map((song) => {
    return (
      <li className="list-group-item" key={song.id}>
        <Link to={`/songs/${song.id}`} style={{ textDecoration: "none" }}>
          {song.title} ( {song.musician} )
        </Link>
      </li>
    );
  });
  return (
    <div>
      <h2 className="m-5">Song List</h2>
      <ul className="list-group">{list}</ul>
    </div>
  );
};

export default SongList;
```

4.2 URI 파라미터 적용

❖src/pages/SongDetail.tsx 추가

```
import { Link, useParams, useNavigate } from "react-router-dom";
import { SongType } from "../App";
import { useEffect, useState } from "react";

type PropsType = { songs: SongType[] };
type SongParam = { id: string };

const SongDetail = (props: PropsType) => {
  const { id } = useParams<SongParam>();
  const navigate = useNavigate();
  const [title, setTitle] = useState<string>("");
  const [musician, setMusician] = useState<string>("");
  const [link, setLink] = useState<string>("");
  const YOUTUBE_LINK = "https://m.youtube.com/watch?v=";

  useEffect(() => {
    const song = props.songs.find((song) => song.id === parseInt(id ? id : "", 10));
    if (song) {
      setLink(song?.youtube_link ? YOUTUBE_LINK + song.youtube_link : "");
      setTitle(song?.title ? song.title : "");
      setMusician(song?.musician ? song?.musician : "");
    } else {
      navigate("/songs");
    }
  }, []);
```

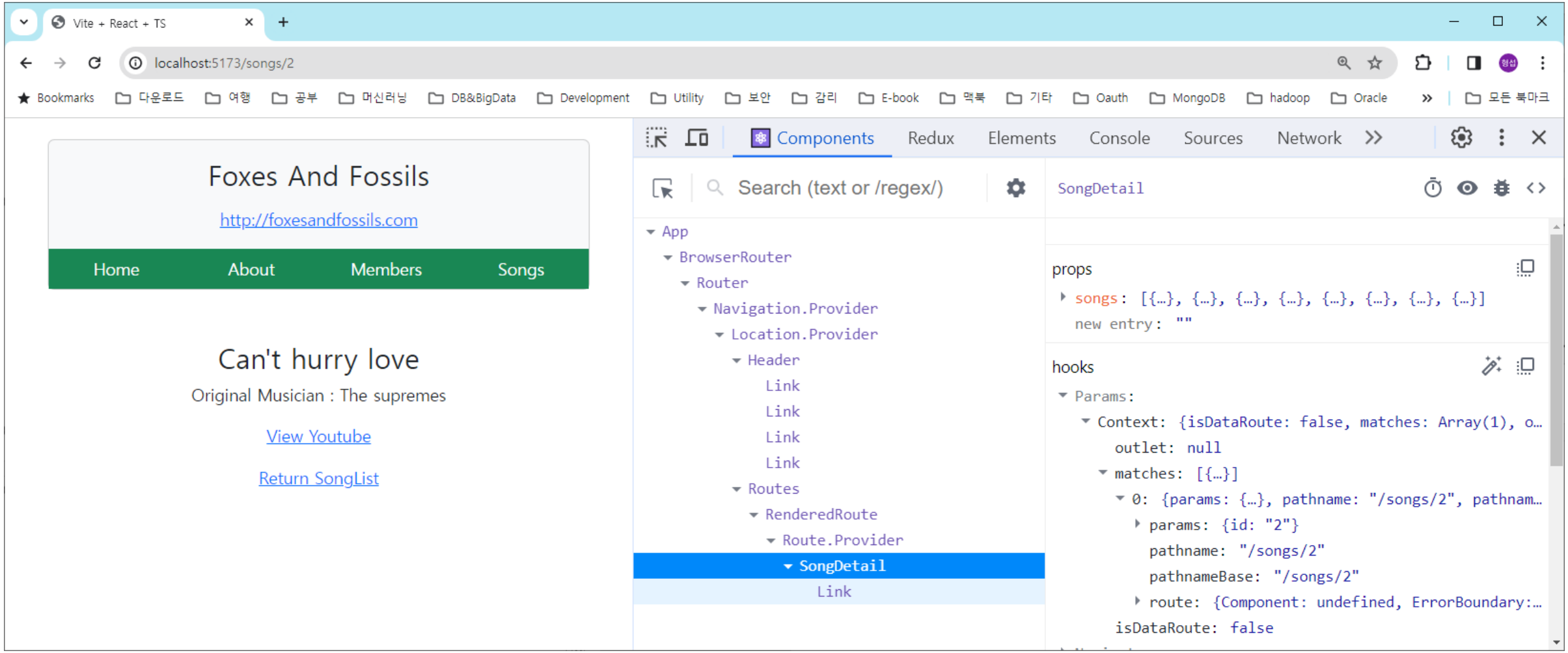
4.2 URI 파라미터 적용

❖src/pages/SongDetail.tsx 추가(이어서)

```
 return (
    <div className="mt-5">
      <h2>{title}</h2>
      <p>Original Musician : {musician}</p>
      <p>
        <a href={link} target="new">
          View Youtube
        </a>
      </p>
      <Link to="/songs">Return SongList</Link>
    </div>
  );
};

export default SongDetail;
```

4.2 URI 파라미터 적용

❖실행 결과 확인



5.1 react-router의 중첩 라우트

❖ 중첩 라우트

- 〈Route /〉 에 렌더링된 컴포넌트에 기존 라우트의 중첩된 〈Route /〉의 컴포넌트가 나타나도록 구성하는 〈Route /〉 컴포넌트의 적용방법
- 이미 1회차에서 설명한 바 있음

❖ 이전단계 예제 Route : 중첩라우트 아님

```
<Routes>
  ……
  <Route path="/songs" element={<SongList songs={songs} />} />
  <Route path="/songs/:id" element={<SongDetail songs={songs} />} />
</Routes>
```

- /songs 로 요청하는 경우 : SongList 컴포넌트 렌더링
- /songs/:id 로 요청하는 경우 : SongDetail 컴포넌트 렌더링

5.1 react-router의 중첩 라우트

❖ 중첩라우트

```
<Routes>
  ……
  <Route path="/songs" element={<SongList songs={songs} />}>
    <Route path=":id" element={<Player songs={songs} />} />
  </Route>
</Routes>
```

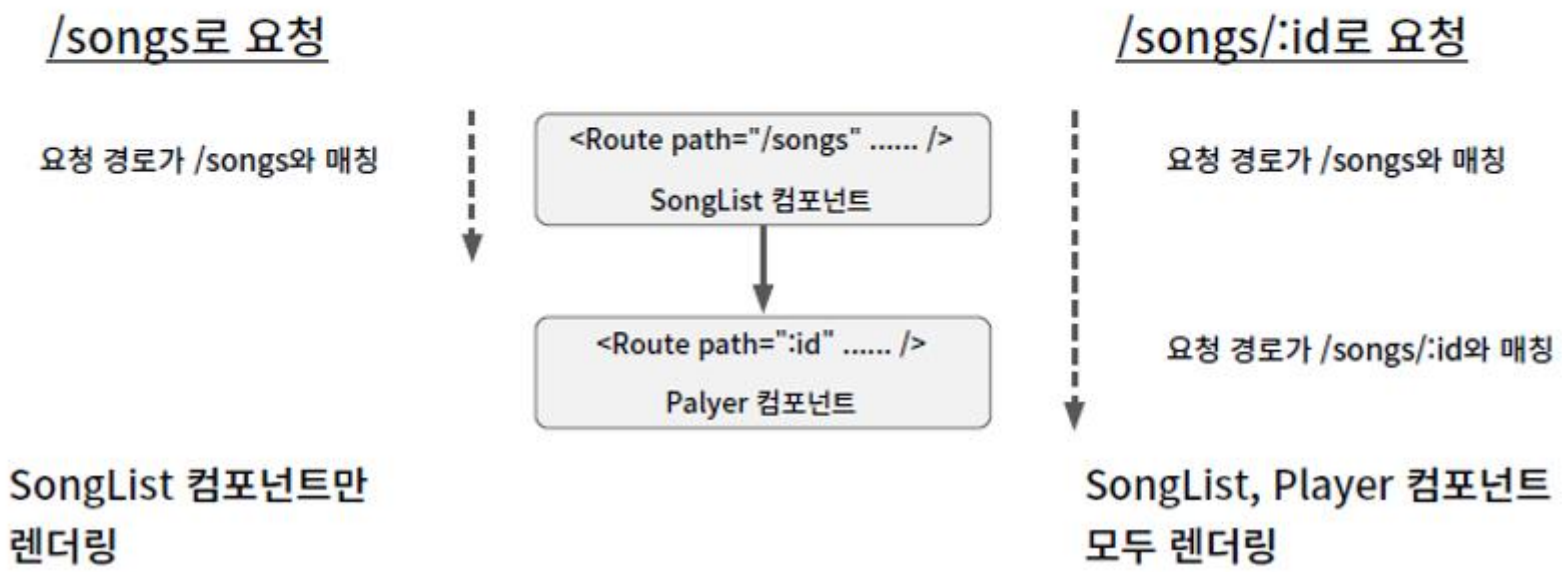
- /songs 로 요청하는 경우 : SongList 컴포넌트 렌더링
- /songs/:id 로 요청하는 경우 : SongList, Player 컴포넌트 렌더링
- 상위 라우트에 의해 렌더링되는 컴포넌트에 〈Outlet /〉 컴포넌트가 존재해야 함

```
//SongList 컴포넌트에…
<div>
   <h2 className="m-5">Song List</h2>
   <ul className="list-group">{list}</ul>
   <Outlet />
</div>
```

5.1 react-router의 중첩 라우트

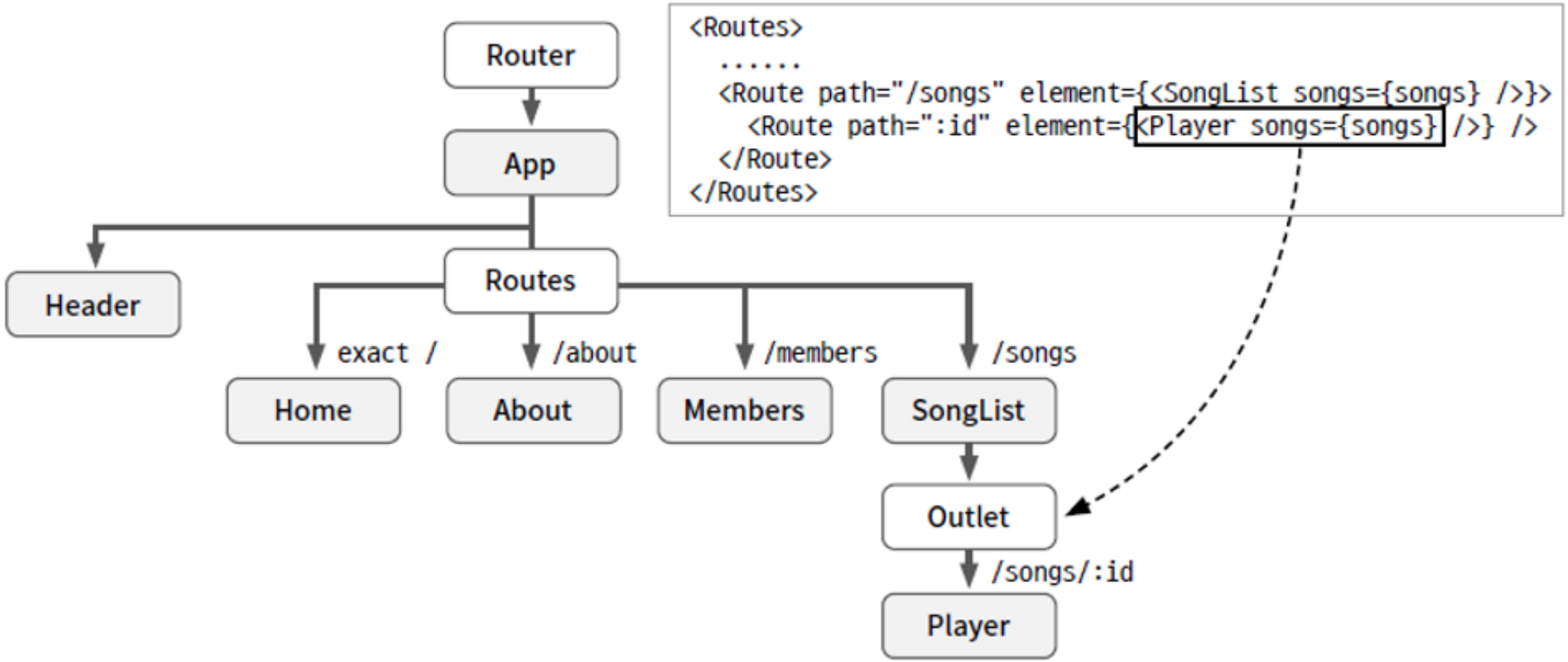
❖ 실행 방식

```
......
  <Route path="/songs" element={<SongList songs={songs} />}>
     <Route path=":id" element={<Player songs={songs} />} />
  </Route>
  ......
```



5.1 react-router의 중첩 라우트

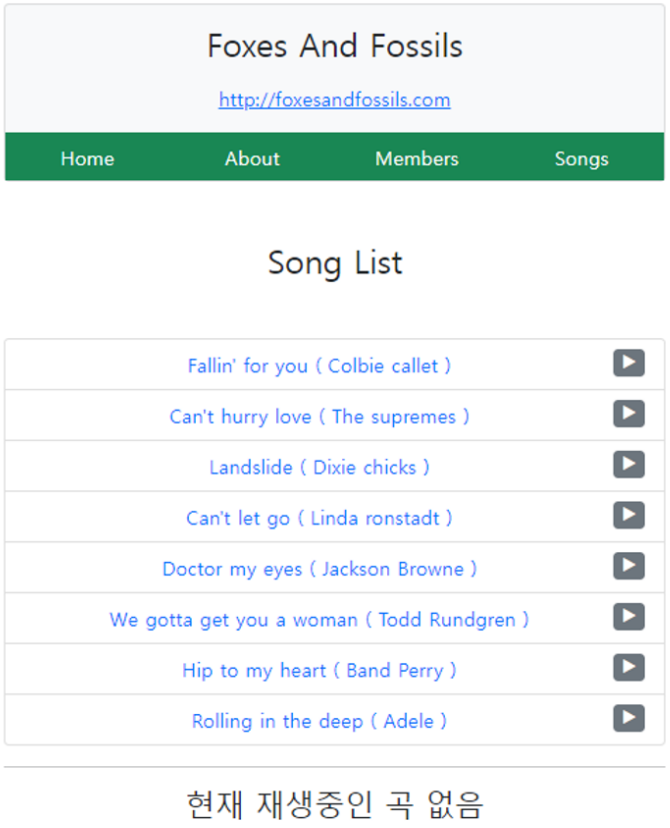
❖ 중첩 라우트 실행 구조



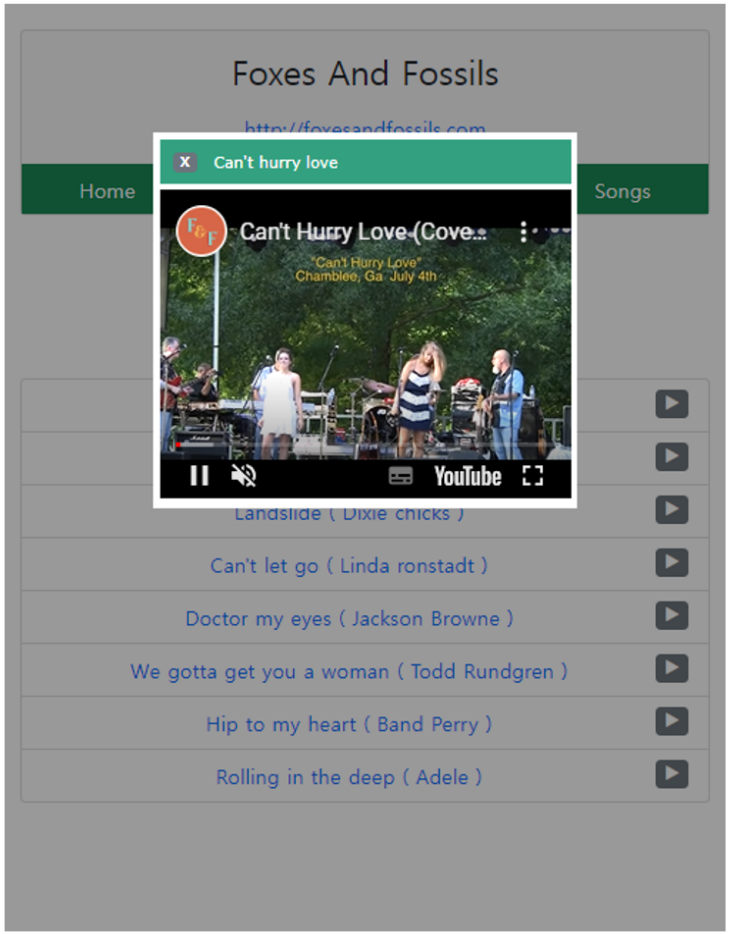
5.1 react-router의 중첩 라우트

❖ 중첩 인덱스 라우트

/songs 요청



/songs/:id 요청



5.2 중첩 라우트 적용

❖ 유튜브 관련 패키지 설치

- npm install react-youtube

❖src/pages/Player.tsx 추가

```
import { useEffect, useState } from "react";
import { useParams, useNavigate } from "react-router";
import { Link } from "react-router-dom";
import Youtube from "react-youtube";
import { SongType } from "../App";
type PropsType = { songs: SongType[] };
type SongParam = { id: string };

const Player = (props: PropsType) => {
  const params = useParams<SongParam>();
  const navigate = useNavigate();
  const [title, setTitle] = useState<string>("");
  const [youtubeLink, setYoutubeLink] = useState<string>("");

  useEffect(() => {
    const id = params.id ? parseInt(params.id, 10) : 0;
    const song = props.songs.find((song) => song.id === id);
    if (song) {
      setTitle(song?.title ? song.title : "");
      setYoutubeLink(song?.youtube_link ? song.youtube_link : "");
    } else {
      navigate("/songs");
    }
  });
```

5.2 중첩 라우트 적용

❖src/pages/Player.tsx(이어서)

```
  return (
    <div className="modal">
      <div className="box">
        <div className="heading">
          <Link className="menu" to="/songs">
            <span className="float-start badge bg-secondary pointer">X</span>
          </Link>
          <span className="title">   {title}</span>
        </div>
        <div className="player">
          <div>
            <Youtube videoId={youtubeLink} opts={{ width: "320", height: "240", playerVars: { autoplay: 1 } }} />
          </div>
        </div>
      </div>
    </div>
  );
};

export default Player;
```

5.2 중첩 라우트 적용

❖src/pages/SongList.tsx 변경

```
import { Link, Outlet } from "react-router-dom";
import { SongType } from "../App";
type Props = { songs: SongType[] };

const SongList = (props: Props) => {
  let list = props.songs.map((song) => {
    return (
      <li className="list-group-item" key={song.id}>
        <Link to={`/songs/${song.id}`} style={{ textDecoration: "none" }}>
          {song.title} ( {song.musician} )
          <span className="float-end badge bg-secondary">
            <i className="fa fa-play"></i>
          </span>
        </Link>
      </li>
    );
  });
  return (
    <div>
      <h2 className="m-5">Song List</h2>
      <ul className="list-group">{list}</ul>
      <Outlet />
    </div>
  );
};

export default SongList;
```

5.2 중첩 라우트 적용

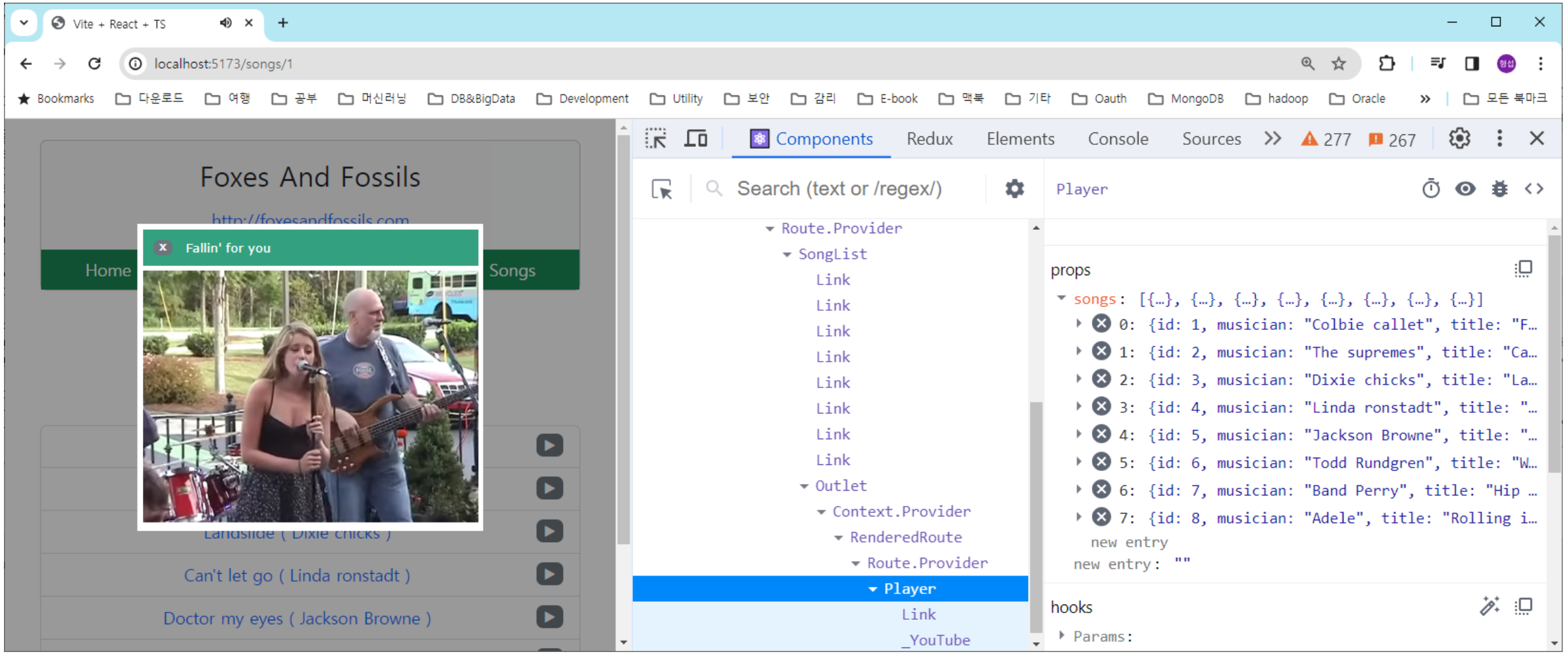
❖src/index.css 변경, App.tsx 변경

```
/* font-awesome이 제공하는 아이콘을 사용하기 위한 참조 */
@import "https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css";

.....(기존 스타일 생략)

/* 모달을 위한 스타일 추가 */
.modal { display: block; position: fixed; z-index: 1;
    left: 0; top: 0; width: 100%; height: 100%;
    overflow: auto; background-color: rgb(0,0,0);
    background-color: rgba(0,0,0,0.4); }
.box { background-color: white; margin:100px auto;
    max-width: 330px; min-width: 100px; min-height: 250px;
    font: 12px "verdana";
    padding: 5px 5px 5px 5px;  }
.box div { padding: 0;  display: block;  margin: 5px 0 0 0; }
.box .heading { background: #33A17F; font-weight: 300;
    text-align: left; color: #fff;
    margin:0px; padding: 10px 10px 10px 10px; min-width:200px;
    max-width:360px; }
.box .player { background:white; }
.pointer { cursor:pointer; }
.play-button { width:15px; height:15px; }
.play-button-disabled { opacity:0.3 }
```

```
<Route path="/songs" element={<SongList songs={songs} />}>
    <Route path=":id" element={<Player songs={songs} />} />
</Route>
```

5.2 중첩 라우트 적용

❖실행 결과



5.3 중첩 index 라우트

❖ 중첩 인덱스 라우트

- 기본 자식 라우트의 성격(default child route)
- 요청된 경로가 부모 경로와 일치할 때 기본적으로 보여줄 자식 라우트

```
<Routes>
  ……
  <Route path="/songs" element={<SongList songs={songs} />}>
    <Route index element={<SongIndex />} />
    <Route path=":id" element={<Player songs={songs} />} />
  </Route>
</Routes>
```

- /songs 로 요청할 경우 : SongList, SongIndex 컴포넌트 렌더링
- /songs/:id 로 요청한 경우 : SongList, Player 컴포넌트 렌더링

- 이미 1회차에서 다루었던 내용으로 예제 작성은 생략

6.1 Router 컴포넌트

❖다양한 Router 컴포넌트

- **BrowserRouter**

  BrowserRouter는 HTML5 History API를 사용하여 URI와 UI를 동기화한 상태를 유지할 수 있는 기능을 제공합니다. BrowserRouter는 URI 경로를 사용하여 브라우저의 주소를 저장하고, 브라우저 history 객체의 스택을 사용해 탐색합니다. BrowserRouter 사용은 웹 브라우저에서 리액트 라우터를 적용할 때 가장 권장하는 방법입니다.

- **HashRouter**

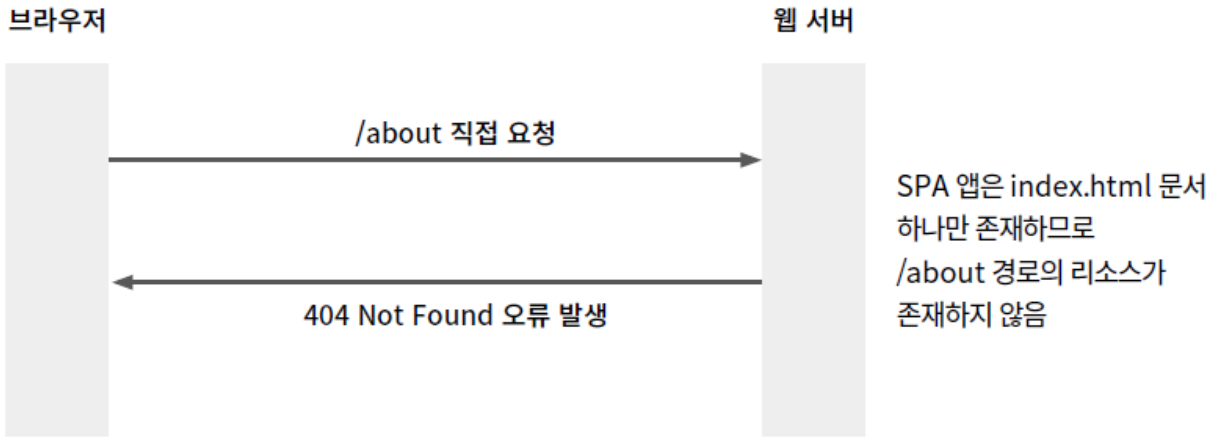
  URL의 해시 정보를 이용해서 URI 경로와 UI를 동기화한 상태로 유지시킵니다. 해시는 # 기호로 표시됩니다. 이 라우터는 주로 BrowserRouter가 지원되지 않는 환경일 때 사용할 것을 권장합니다. HashRouter는 http://localhost:3000/#/about과 같이 # 다음에 /about처럼 라우팅에 사용하는 경로가 브라우저의 주소 입력란에 찍힙니다.

- **MemoryRouter**

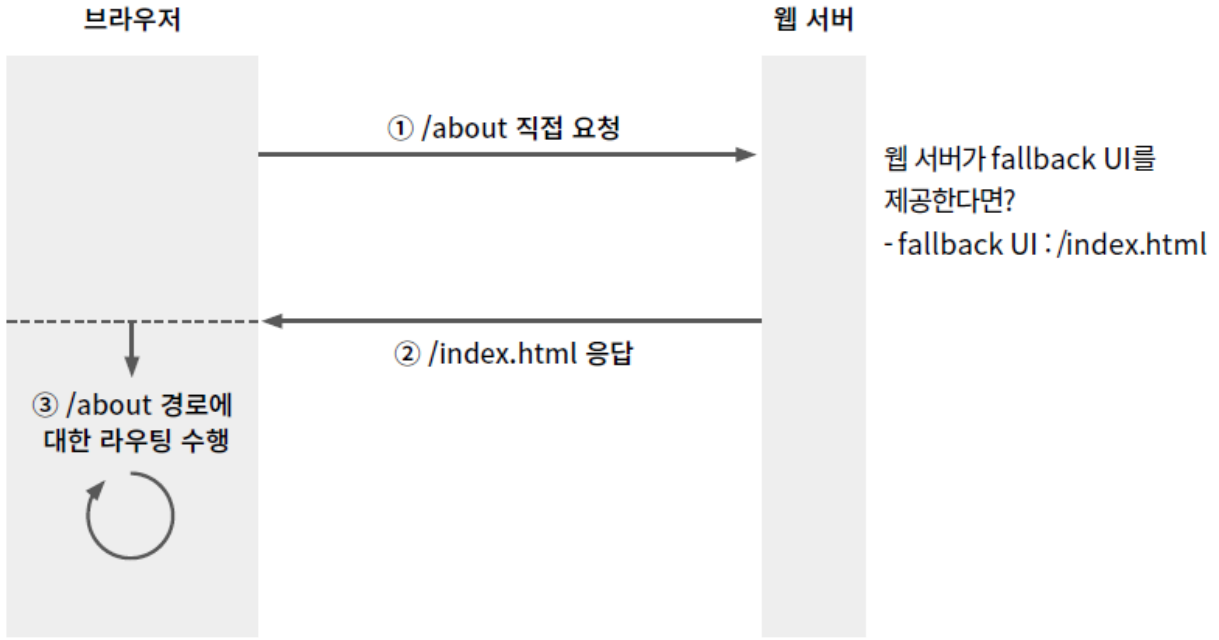
  MemoryRouter는 애플리케이션의 메모리 영역에 배열을 만들어 라우팅 정보를 저장하고 UI와 동기화시킵니다. 따라서 URI 경로가 브라우저의 주소창에 표시되지 않고 메모리에만 유지됩니다. 브라우저 주소 UI를 보여주지 않아도 되는 하이브리드 앱 같은 경우에 사용할 수 있습니다.

6.1 Router 컴포넌트

❖fallback UI 지원여부에 따른 실행 결과



[그림 9-20] fallback UI를 지원하지 않는 웹 서버일 때



[그림 9-21] fallback UI를 지원하는 웹 서버일 때

Q&A 웹 서버에 fallback UI를 어떻게 설정하나요?

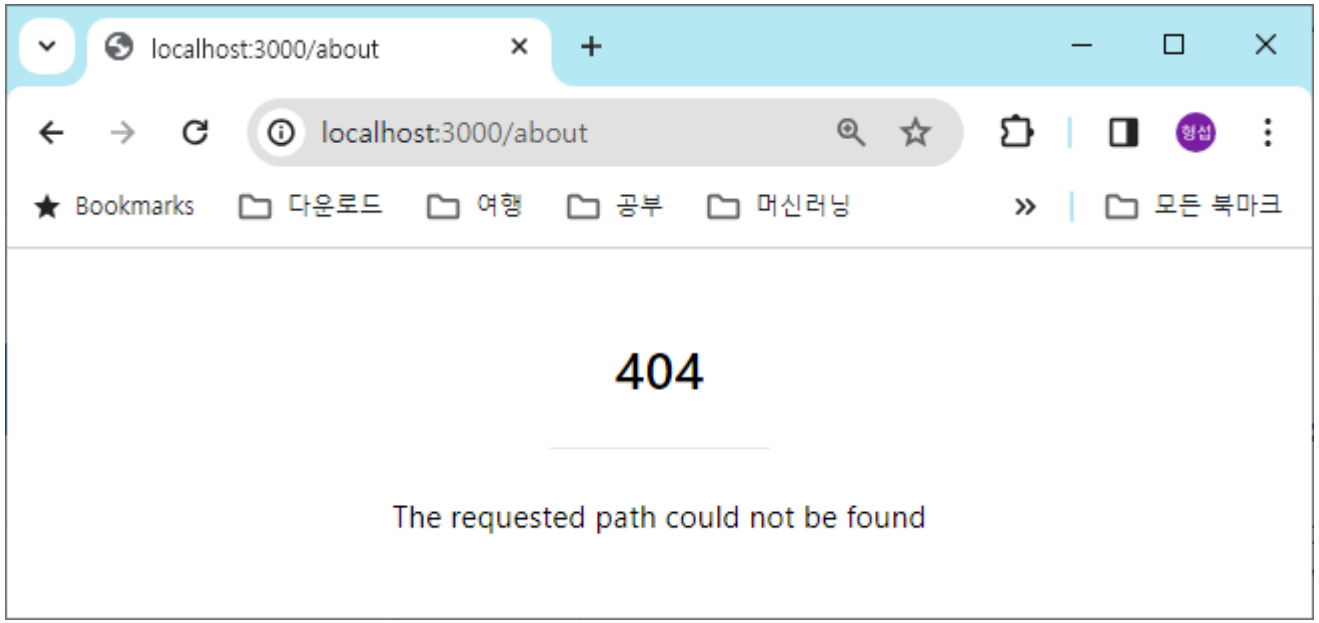
웹 서버마다 fallback UI를 지정하는 방법이 다릅니다. 따라서 인터넷을 검색해서 fallback UI를 지원하는지를 찾아보는 것이 좋습니다. 다음은 리액트가 아니라 Vue의 공식 문서 내용이지만 Vue Router의 사용 가이드에 서버 설정 방법이 소개되어 있으므로 참고해보세요. Apache, Nginx, Node.js, Express, IIS 등 주요 웹 서버의 설정 방법이 안내되어 있습니다.

https://v3.router.vuejs.org/kr/guide/essentials/history-mode.html#서버-설정-예제

6.2 fallback UI가 없는 웹서버에서의 에러 확인

❖지금까지 실행은 npm run dev로 개발서버 실행한 것
- 개발서버는 fallback UI 지원(/index.html)

❖에러를 확인하기 위해 다른 서버를 구동해야 함
- npm run build
- npx serve dist --listen 3000
- 직접 브라우저를 열어서 http://localhost:3000/about 요청!!



6.2 fallback UI가 없는 웹서버에서의 에러 확인

❖만일 fallback UI를 사용할 수 없는 서버라면? 또는 서버 설정을 변경할 권한이 없다면?

- HashRouter

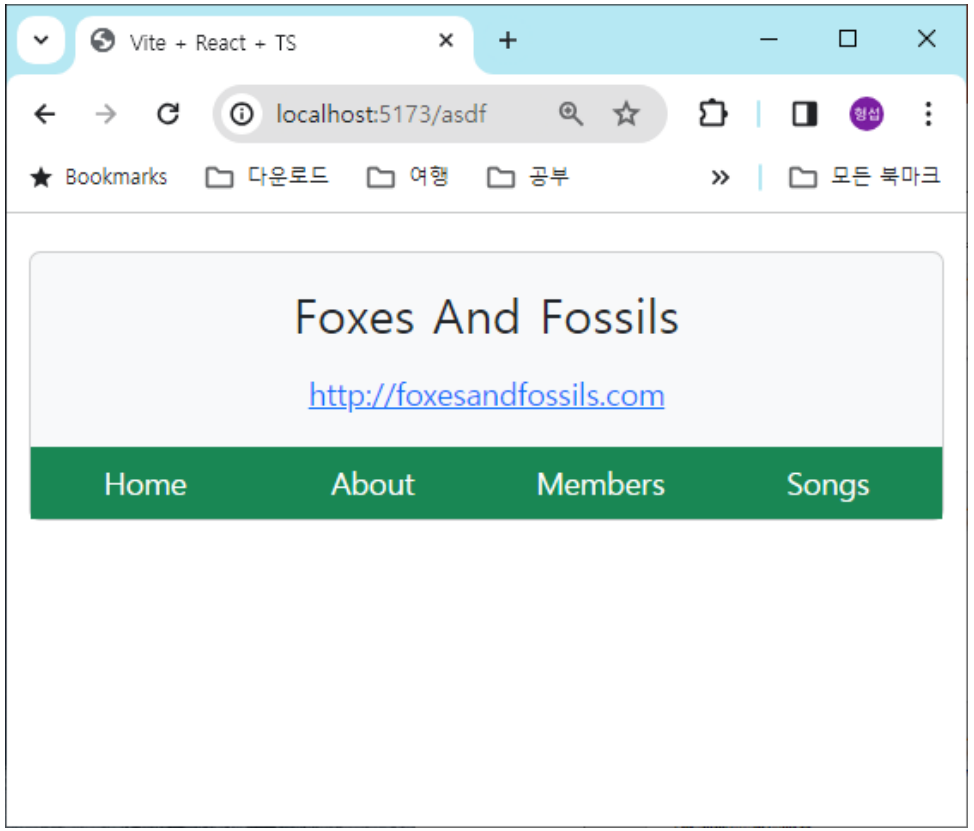
```
import { HashRouter as Router, Routes, Route } from "react-router-dom";
```

- http://localhost:3000/#/about 의 형태로 요청
- # 기호를 기준으로 앞쪽이 요청 경로이고 뒷부분은 문서 내부의 컨텐츠를 의미함.

6.3 404 라우트와 리디렉션 구성

❖존재하지 않는 경로로 요청하면?

- 예) http://localhost:3000/asdf
- 에러가 없지만 빈 화면!!



매칭

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About title={"여우와 늙다리들"} />} />
  <Route path="/members" element={<Members members={members} />} />
  <Route path="/songs" element={<SongList songs={songs} />}>
    <Route index element={<SongIndex />} />
    <Route path=":id" element={<Player />} />
  </Route>
</Routes>
```

```
//마지막 Route로 404 라우트를 배치합니다.
<Routes>
  ......
  <Route path="*" element={ <NotFound /> } />
</Routes>
```

```
// /a로 요청했을 때 /b로 리디렉션
<Route path="/a" element={<Navigate to="/b" />} />
<Route path="/b" element={<BComponent />} />
```

6.3 404 라우트와 리디렉션 구성

❖잘못된 경로일 때 보여줄 컴포넌트 추가

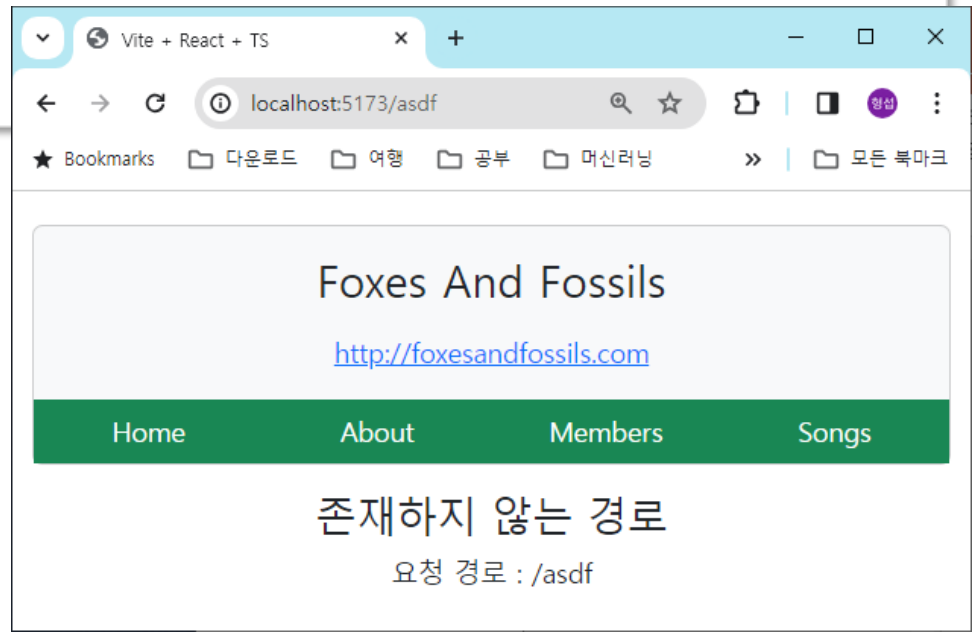
- src/components/NotFound.tsx

```
import { useLocation } from "react-router"

const NotFound = () => {
  const location = useLocation();

  return (
    <div className="m-3">
      <h3>존재하지 않는 경로</h3>
      <p>요청 경로 : {location.pathname}</p>
    </div>
  )
}

export default NotFound
```

6.3 404 라우트와 리디렉션 구성

❖App 컴포넌트의 Route 변경

```
<Routes>
  <Route path="/" element={<Navigate to="/home" />} />
  <Route path="/home" element={<Home />} />
  <Route path="/about" element={<About title={'여우와 늙다리들'}/>} />
  <Route path="/members" element={<Members members={members} />} />
  <Route path="/songs" element={<SongList songs={songs} />}>
          <Route path=":id" element={<Player songs={songs} />} />
  </Route>
  <Route path="*" element={<NotFound />} />
</Routes>
```



Q&A