

Spring Boot 및 React 기반 템플릿 프로젝트 소개 및 시연



Contents

1. React 소개
2. Spring Boot 기반 템플릿 프로젝트





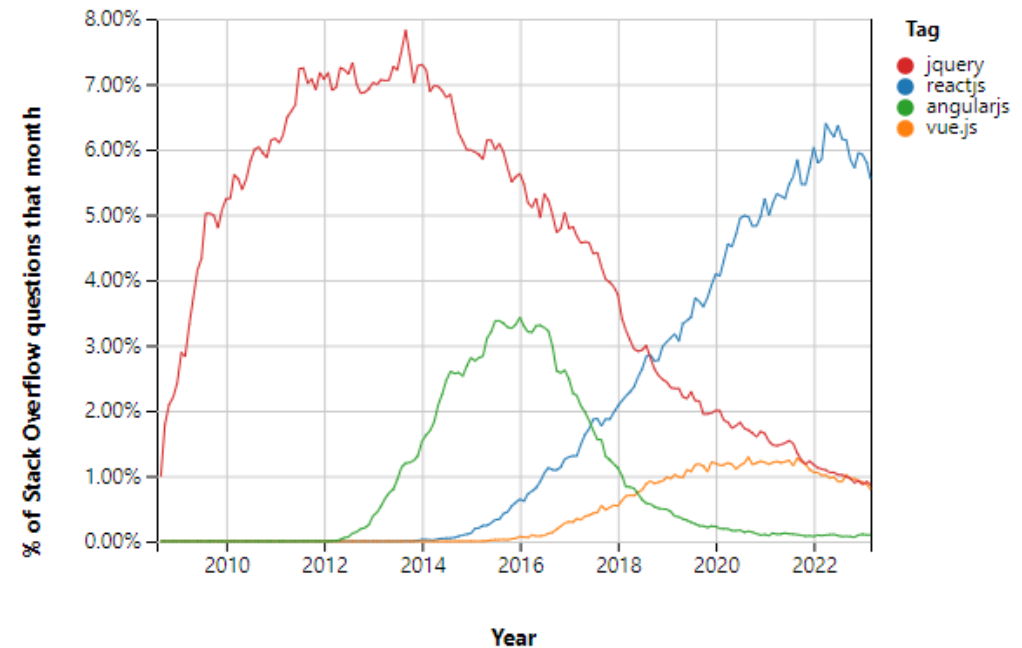
1. React 소개

1. SPA (Single Page Application)
2. React.js
3. 컴포넌트
4. CRA (create-react-app)

- ❑ 단일 페이지로 구성된 웹 애플리케이션
- ❑ 기존 SSR (Server Side Rendering)의 경우, 화면에 보여질 리소스를 서버로 요청
- ❑ 새로고침 및 로딩이 들어가기 때문에 화면 깜박임 및 사용자에게 좋지 않은 경험을 제공
- ❑ SPA의 경우, 모든 정적 리소스를 최초 한번 다운로드하고 새로운 페이지 요청 시에는 갱신에 필요한 데이터만을 전달받아 페이지를 갱신
- ❑ 트래픽 감소 및 렌더링 효율 증가, 빠른 화면 이동, 자연스러운 사용자 경험
- ❑ 모듈화 또는 컴포넌트별 개발이 용이

- ❑ 리소스를 최초 다운로드 -> 초기 구동 속도가 느리다
- ❑ Lazy loading, code splitting
- ❑ 검색엔진에 최적화되어 있지 않음 : 데이터가 없는 사이트로 인식
- ❑ Google의 경우는 SPA를 지원하도록 개선되어 있음
- ❑ SEO (Search Engine Optimization)에 노출되어야 하는 전략적인 페이지는 서버 렌더링 기술 적용 -> Next.js (React), Nuxt.js (Vue)

- ❑ 오픈 소스 자바스크립트 라이브러리
- ❑ SPA를 전제로 함 : 리렌더링이 잦은 웹에서 빠른 퍼포먼스
- ❑ 모듈형 개발 : 생산성이 높다.
- ❑ 프레임워크가 아닌 라이브러리 : 다른 프레임워크에 간단하게 붙여서 사용이 가능
- ❑ 타입스크립트나 Sass 등도 지원
- ❑ Next.js 등의 프레임워크를 이용하면 SSR도 가능
- ❑ Stackoverflow trend 비교 (Global)



□ 리액트 이전

```
<header>
  <h1>Logo</h1>
</header>
<nav>
  <ul>
    <li>
      <a href="#">메뉴 1</a>
    </li>
    <li>
      <a href="#">메뉴 2</a>
    </li>
  </ul>
</nav>
<section>
  <p>Hello World!</p>
</section>
```

❑ 리액트 적용 (class 문법)

```
const Hlogo = () => {  
  return ( <header> <h1>Logo</h1> </header> );  
};  
  
const Anav = () => {  
  return ( <nav> <ul> <li> <a href="#">메뉴1</a> </li> <li> <a href="#">메뉴2</a> </li> </ul> </nav> );  
};  
  
const Bsection = ({ amumal }) => {  
  return ( <section> <p>{amumal}</p> </section> );  
};  
  
const App = () => {  
  return ( <div className="App"> <Hlogo /> <Anav /> <Bsection amumal="Hello World!" /> </div> );  
};  
  
export default App;
```

❑ 컴포넌트 : 화면에서 UI요소를 구분하는 단위. 컴포넌트 클래스는 대문자

❑ JSX : React에서 HTML을 표현. 빌드 시 Babel에 의해 자바스크립트로 변환된다

- ❑ React 개발환경 구축에는 Webpack (여러 자바스크립트 파일을 하나로 합쳐주는 모듈 번들러), Babel (JSX 변환) 등 여러 라이브러리가 추가로 필요하다.
- ❑ 부가적 모듈을 개발자가 일일이 설치하기에는 번거로움
- ❑ 초기 세팅 작업을 미리 하고 그 환경을 다시 패키징한 툴 : create-react-app
- ❑ <https://create-react-app.dev/>

```
npx create-react-app my-app  
cd my-app  
npm start
```

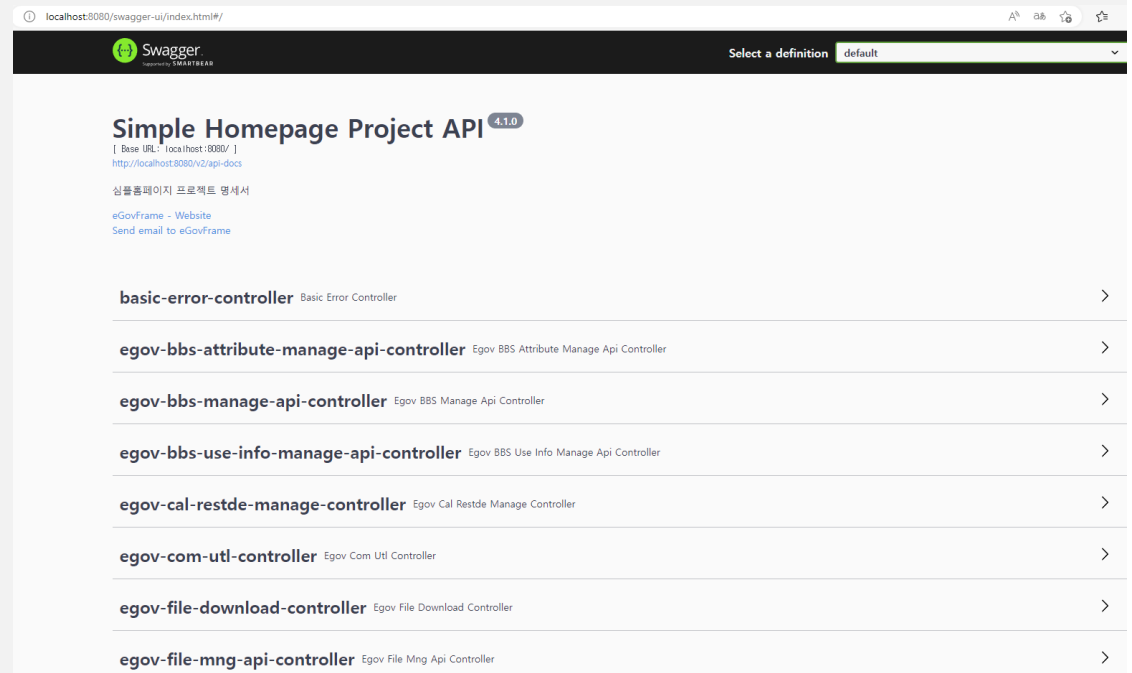
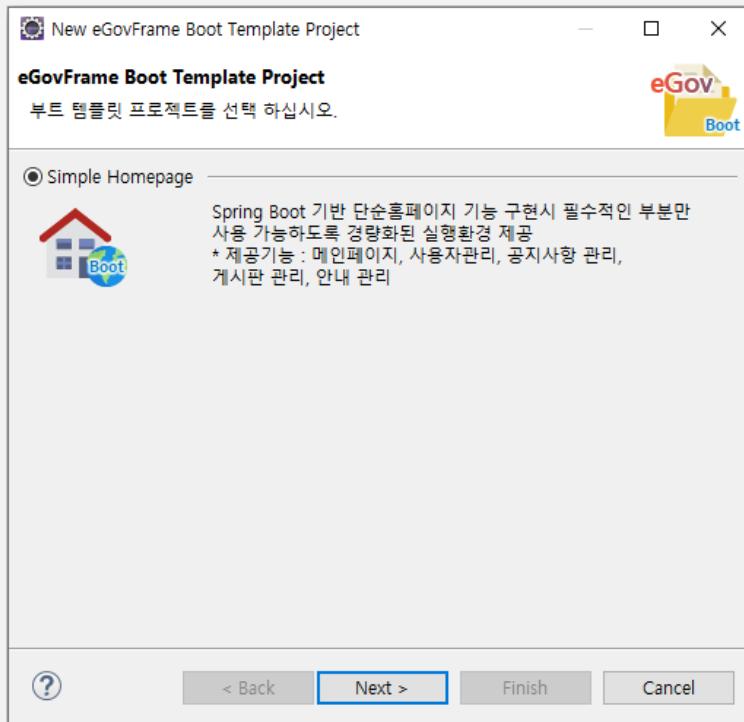


2.SpringBoot 기반 템플릿 프로젝트

1. 템플릿 프로젝트 소개 (심플 홈페이지)
2. Backend 구성 및 세부 내역
3. Frontend 구성 및 개선 사항
4. 종합 시연

❑ 개발환경 – Boot Template Project (Simple Homepage)

- 표준프레임워크에서 제공하는 공통 컴포넌트 기능 중 일부를 선정하여 기본 구성
- 기존 JSP 뷰 방식의 템플릿 프로젝트를 Spring Boot 기반의 backend와 React 기반의 Frontend 로 분리
- Backend API의 엔드포인트를 확인 가능하도록 4.1부터 Swagger 추가



❑ 개발환경 – Boot Template Project (Simple Homepage)

- Frontend는 CRA (create-react-app) 기반으로 구성
- 표준프레임워크 GitHub Repository에서 제공 : fork 및 Pull Request를 통한 Contribution 가능
- <https://github.com/eGovFramework/egovframe-template-simple-react>

eGovFramework / **egovframe-template-simple-react** Public

Edit Pins Unwatch 14 Fork 57 Star 134

Code Issues Pull requests 7 Actions Projects Wiki Security 2 Insights Settings

contribution 4 branches 2 tags Go to file Add file Code

File/Folder	Description	Last Commit
.github	Fix: test case 추가	2 weeks ago
Docs	Doc: 파일명 변경	last year
public	Doc: 파일명 변경	last year
src	Fix: 렌더링 오류 수정	last week
.env.development	Style: 코드 정리	2 years ago
.env.production	Fix: 배포시 소스맵 삭제 설정	2 years ago
.gitignore	Initialize project using Create React App	2 years ago
.project	egovframe-template-simple-react 4.1.0 FINAL	last month
README.md	Doc: CI 아이콘 위치 변경	5 days ago
jsconfig.json	Feature: 점검회의를 위한 임시 커밋	2 years ago
package-lock.json	package-lock.json 추가	2 weeks ago
package.json	egovframe-template-simple-react 4.1.0 FINAL	last month

About
No description, website, or topics provided.
Readme
134 stars
14 watching
57 forks
Report repository

Releases 2
V4.1.0 Latest
on Mar 2
+ 1 release

Packages
No packages published
Publish your first package

❑ EgovBootApplication.java

- 프로젝트의 최상단에 위치하는 클래스
- main 에서 run을 호출하면 내장 톰캣으로 웹 어플리케이션이 실행되고 IoC 컨테이너가 초기화됨

❑ @SpringBootApplication

- main에 위치하며 해당 위치부터 설정을 읽는다.
- 해당 어노테이션을 통해 Spring Boot의 자동 설정, Bean 읽기와 생성이 모두 자동으로 설정됨

❑ @EnableAutoConfiguration

- 미리 정의된 Bean들을 가져와서 등록
- spring-boot-autoconfigure > META-INF > spring.factories

❑ @ComponentScan

- 현재 패키지 이하에서 다음과 같은 어노테이션이 붙어 있는 클래스들을 찾아서 Bean으로 등록
- @Component, @Configuration, @Repository, @Service, @Controller, @RestController

❑ @SpringBootConfiguration

- Spring Boot 환경 설정 클래스임을 나타내는 어노테이션

❑ src/main/java

- 클래스, 인터페이스 등 자바 파일이 위치
- Jwt/config 패키지에서 jwt 관련 사항을 확인 가능

❑ src/main/resources

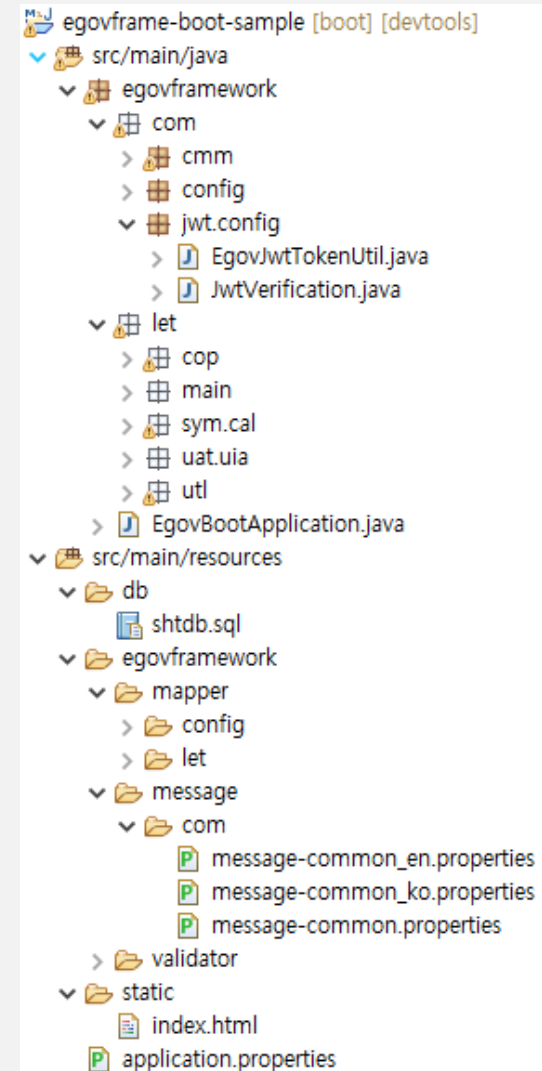
- db : Local hsql용 DDL, DML
- mapper : 6종 DB의 mapper, 설정 파일 (DDL, DML은 DATABASE 폴더)
- message : 사전 정의된 메시지 관련 properties 파일, 다국어 설정
(EgovConfigAppCommon.java)

❑ Pom.xml

- Maven 빌드 정보를 기재

❑ application.properties

- Spring Boot가 Application을 구동할 때 자동으로 로딩하는 파일
- key - value 형식으로 값을 정의하면 application에서 참조하여 사용 가능

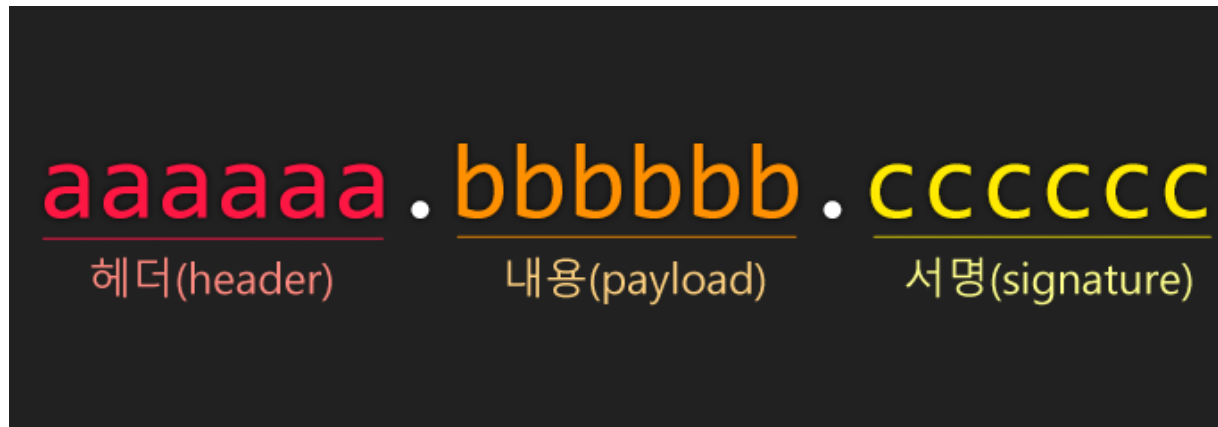


□ JWT (JSON Web Token)

- 웹 표준(RFC 7519)으로 JSON 포맷을 이용해 정보를 가볍고 안전하게 전송하기 위한 Claim 기반의 Web Token
- 서버만 알고 있는 Secret Key로 디지털 서명화되어 있기 때문에 신뢰 가능
- 토큰 자체에 정보를 담아서 HTTP 헤더에 전달하므로 유저 세션을 유지할 필요가 없음

□ JWT의 구조

- Header : 타입 및 해싱 알고리즘
- Payload : 토큰에 담을 정보, 토큰 생성 시간, 만료시간 등
- Signature : Header의 인코딩값과 정보의 인코딩 값을 합친 후 Secret Key 로 해쉬를 하여 생성



❑ JWT 생성 및 검증

- Jwt.io : 브라우저 상에서 JWT 토큰을 검증하고 생성할 수 있게 해주는 디버거 서비스
- secret key : 탈취 시 생성, 변조 등의 악의적 행위가 가능해짐 → secret key는 최소 512bits(약 64글자) 이상으로 설정 권장
- 토큰의 payload에는 민감하거나 중요한 데이터는 포함하지 않도록 해야 한다.

The screenshot displays the JWT.io interface with the 'Decoded' tab selected. The 'Algorithm' is set to 'HS512'. The 'Encoded' section shows a long base64-encoded string. The 'Decoded' section shows the token's structure: a header with 'alg': 'HS512', and a payload with 'sub': 'USRadmin', 'exp': 1681197663, and 'iat': 1681194863. A tooltip indicates the expiration time as 'Tue Apr 11 2023 15:21:03 GMT+0900 (한국 표준시)'. The 'VERIFY SIGNATURE' section shows the HMACSHA512 formula and a checkbox for 'secret base64 encoded' which is unchecked. A blue 'SHARE JWT' button is at the bottom right. A 'Signature Verified' message with a checkmark is at the bottom left.

Algorithm: HS512

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJVU1JhZG1pbiIsImV4cCI6MTY4MTE5NzY2MywiaWF0IjoxNjg1MTk0MDYzfQ.1dN0qoqMaM3l8uSm-qwv34CK-e_cc1RHxIVKyC5G_7TneUqBWUGg6HsM25oj_cknHuj6RSjfqmA0B-dVpCAM3Q
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS512"}
```

PAYLOAD: DATA

```
{  "sub": "USRadmin",  "exp": 1681197663,  "iat": 1681194863}
```

Tue Apr 11 2023 15:21:03 GMT+0900 (한국 표준시)

VERIFY SIGNATURE

```
HMACSHA512(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  egovframe)
```

☐ secret base64 encoded

Signature Verified

SHARE JWT

❑ node_modules/

- 외부 모듈을 설치하면 다운로드되는 폴더
- 각 모듈 저장소에 이미 코드가 있으므로 github 등에 올릴 때에는 .gitignore 처리함

❑ src/api/egovFetch.js

- Promise 기반 비동기 Http 통신 라이브러리인 Fetch를 사용. 공통함수 처리
- requestFetch(url, requestOptions, handler, errorHandler)
- url : 호출 주소
- requestOptions : 호출 옵션 (header 설정 및 body에 담을 데이터 설정)
- handler : 성공 시 처리할 function
- errorHandler : 실패 시 처리할 function => 미지정 시 default 처리 수행

❑ index.html

- 서버에서 읽고, 브라우저가 표출하는 파일

❑ index.js

- App 컴포넌트를 document 내 id 값이 root인 태그 안에 렌더링

❑ package.json

- 프로젝트 종속성 및 설명, 버전, 라이선스 정보 등의 메타 데이터

❑ Fetch vs Axios

	Fetch	Axios
장점	<ul style="list-style-type: none">• Import를 하지 않고 사용 가능• 내장 라이브러리로 업데이트에 따른 에러 방지가 가능	<ul style="list-style-type: none">• Fetch보다 다양한 기능을 지원• JSON 데이터의 자동 변환• Response timeout 처리 방법이 존재
단점	<ul style="list-style-type: none">• Response timeout 제공이 되지 않는다• 요청 시 <code>JSON.stringify()</code> 로 객체를 문자열로 변환 후 본문에 할당 필요• 응답 데이터를 JSON으로 얻기 위해 일반적으로 두 개의 <code>.then()</code> 호출 필요• 일반적으로 Axios 보다 속도가 빠르다	<ul style="list-style-type: none">• 사용을 위해 모듈 설치 필요 (<code>npm i axios</code>)• Fetch보다 무겁다
종합	<ul style="list-style-type: none">• Axios의 React 대응이 늦을 수 있어 React Native처럼 변화가 빠른 환경에서는 Fetch를 사용	<ul style="list-style-type: none">• Axios의 JSON 자동 변환은 경우에 따라 단점이 될 수 있음

□ 공통함수 사용

- Import * as EgovNet from 'api/egovFetch'
- EgovHeader.jsx : 로그아웃 핸들러 사용 예

```
const logOutHandler = () => {
  const logOutUrl = '/uat/uia/actionLogoutAPI.do';
  const requestOptions = {
    credentials: 'include',
  }
  EgovNet.requestFetch(logOutUrl, requestOptions,
    function (resp) {
      console.log("===>>> logout resp= ", resp);
      if (parseInt(resp.resultCode) === parseInt(CODE.RCV_SUCCESS)) {
        onChangeLogin({ loginVO: {} });
        sessionStorage.setItem('loginUser', JSON.stringify({"id":""}));
        window.alert("로그아웃되었습니다!");
        navigate(URL.MAIN);
      }
    }
  );
}
```

❑ putMapping 및 deleteMapping 추가

- postMapping : 반복 수행될 경우 매번 데이터가 등록됨 (역등성 x)
- putMapping : 반복 수행하더라도 같은 데이터 (역등성 o)

❑ react-router-dom 버전을 v5에서 v6으로 변경

- 브라우저 라우팅을 관리하기 위한 라이브러리
- 라우팅 방식 및 기존에 지원하던 기능들의 삭제, 수정이 이루어짐
- <https://reactrouter.com/en/6.10.0/upgrading/v5>

❑ React 버전을 v17 에서 v18로 변경

- 유니크 아이디를 생성 가능한 useId hooks 추가
- 컴포넌트가 undefined를 리턴하더라도 에러 처리되지 않음
- 메모리 사용량 최적화 ... etc
- <https://yceffort.kr/2022/04/react-18-changelog#new-feature>

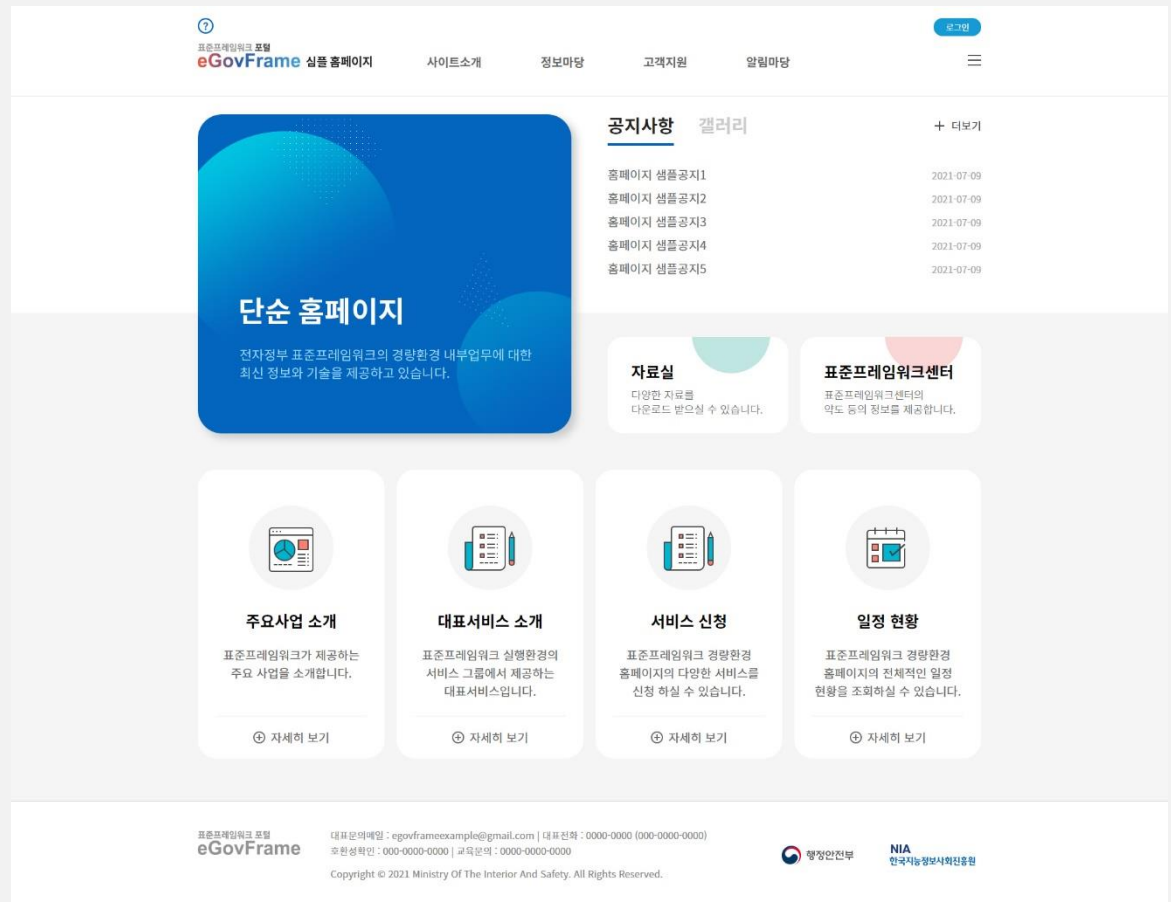
❑ 컴포넌트 키 누락 이슈 수정 및 데이터 등록 시 validation 수정

❑ 등록 기능 개선

- 일정 등록 시 선택한 날짜 디폴트 처리, 파일 첨부 가능 여부 및 게시판 사용 여부 적용 구현

❑ 개발환경 – Boot Template Project (Simple Homepage)

- Backend 및 Frontend 구동 및 동작 확인
- 화면 및 메뉴 설명
- 사이트 소개
- 정보마당
- 고객지원
- 알림마당
- 사이트 관리 (로그인 필요)



감사합니다.