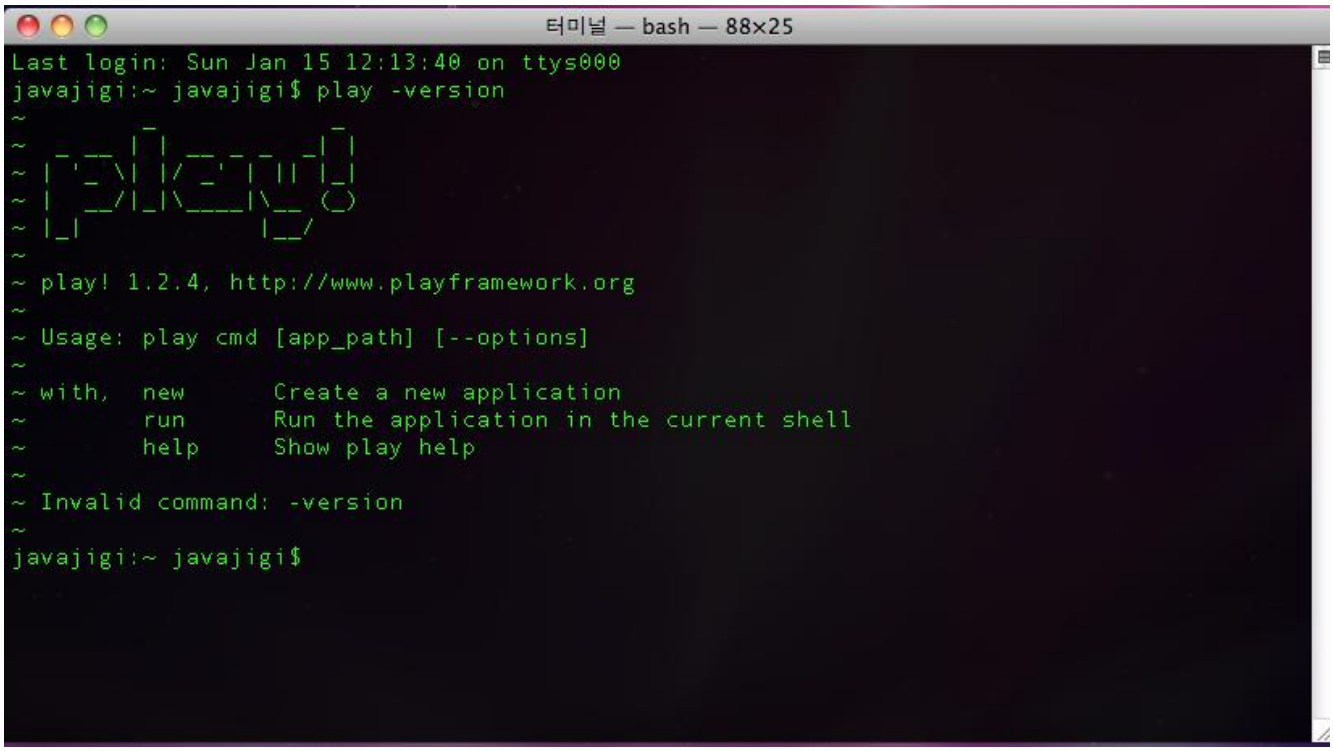
플레이 프레임워크 시작하기 1

- 왜 플레이 프레임워크인가?
- 플레이 프레임워크 설치 및 통합 개발 환경
- 플레이 프레임워크 시작하기 1
- 플레이 프레임워크 시작하기 2

새로운 기술과 개념을 익힐 때 가장 좋은 방법은 프로젝트 복잡도가 낮은 샘플 애플리케이션을 만들어보는 것이다. 플레이 기반으로 “Hello World”를 출력하는 웹 애플리케이션을 개발해 본다. 이 과정은 http://www.playframework.org/documentation/1.2.4/firstapp 문서를 기반으로 작성했다.

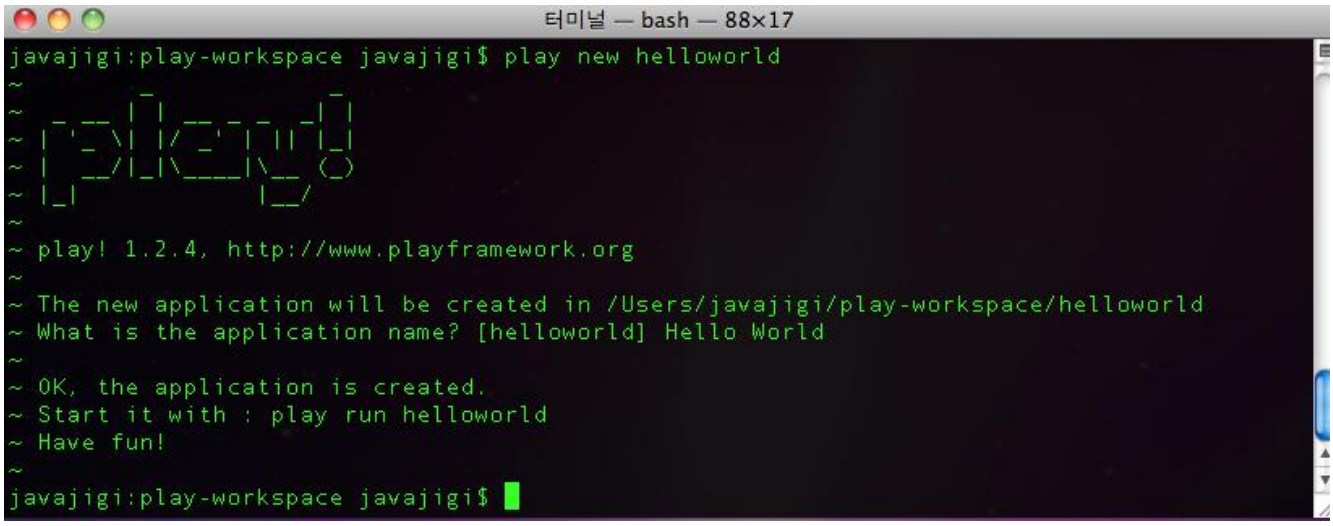
“play -version”을 실행해 플레이가 정상적으로 설치되었는지 확인한다.

~$ play -version



helloworld 프로젝트를 생성한다.

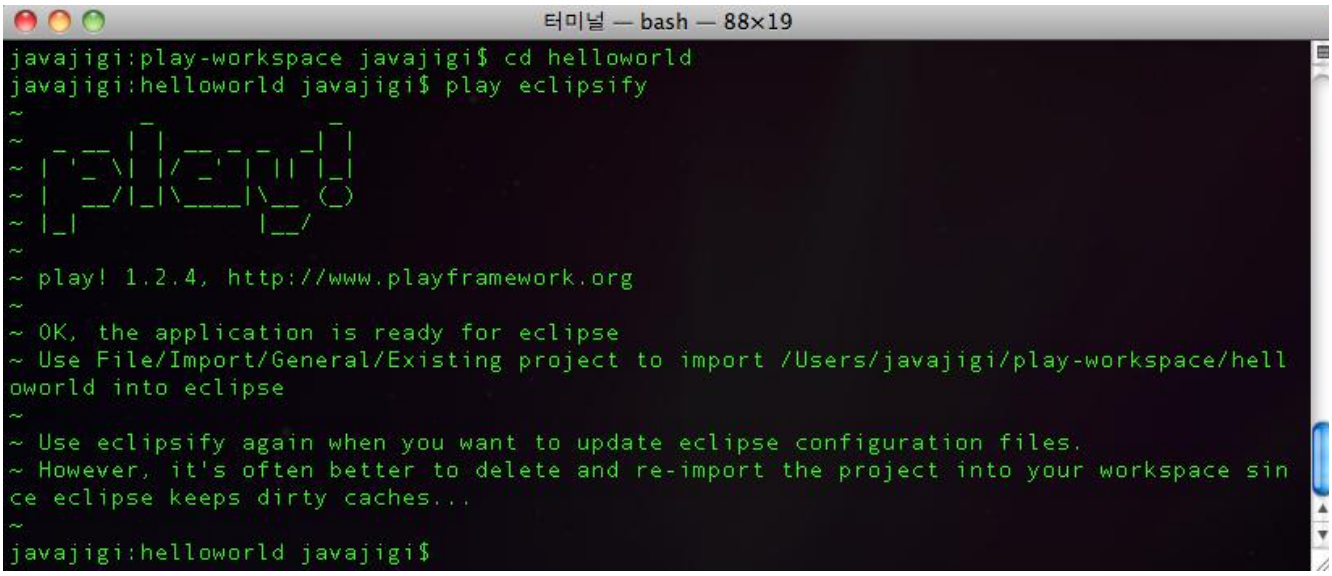
~$ play new helloworld



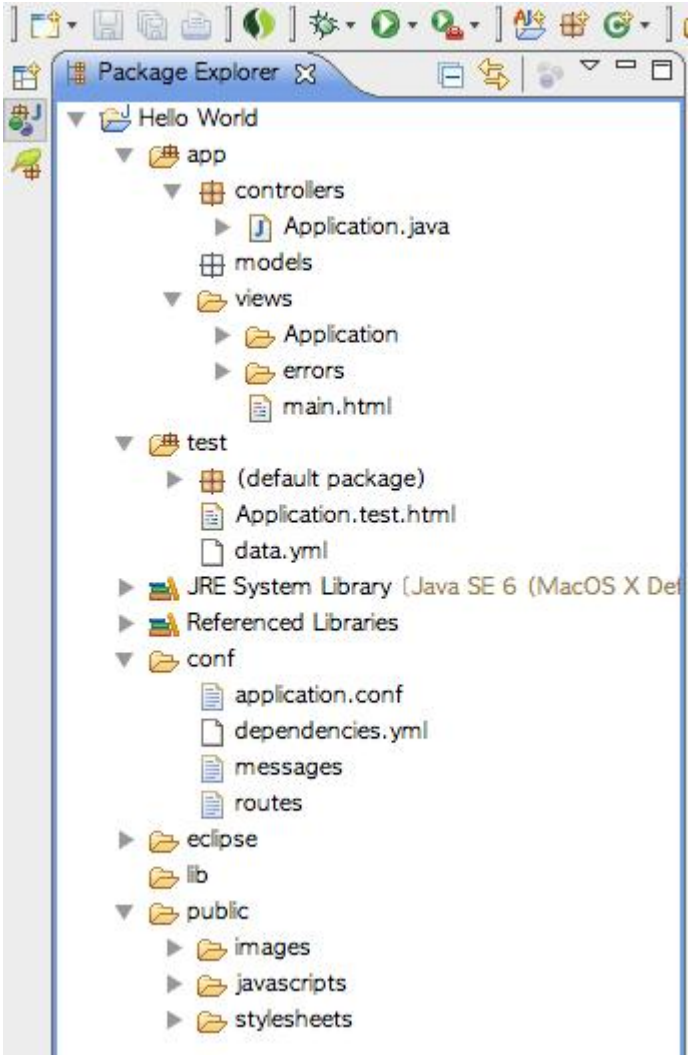
위 명령을 실행하면helloworld 디렉토리 아래에 플레이 기반 웹 애플리케이션이 생성된다. helloworld 디렉토리로 이동한 후 자신이 선호하는 통합 개발 환경으로 구축한다. 이 글에서는 이클립스 기반으로 개발 환경을 구축했다.

~$ cd helloworld

~$ play eclipsify



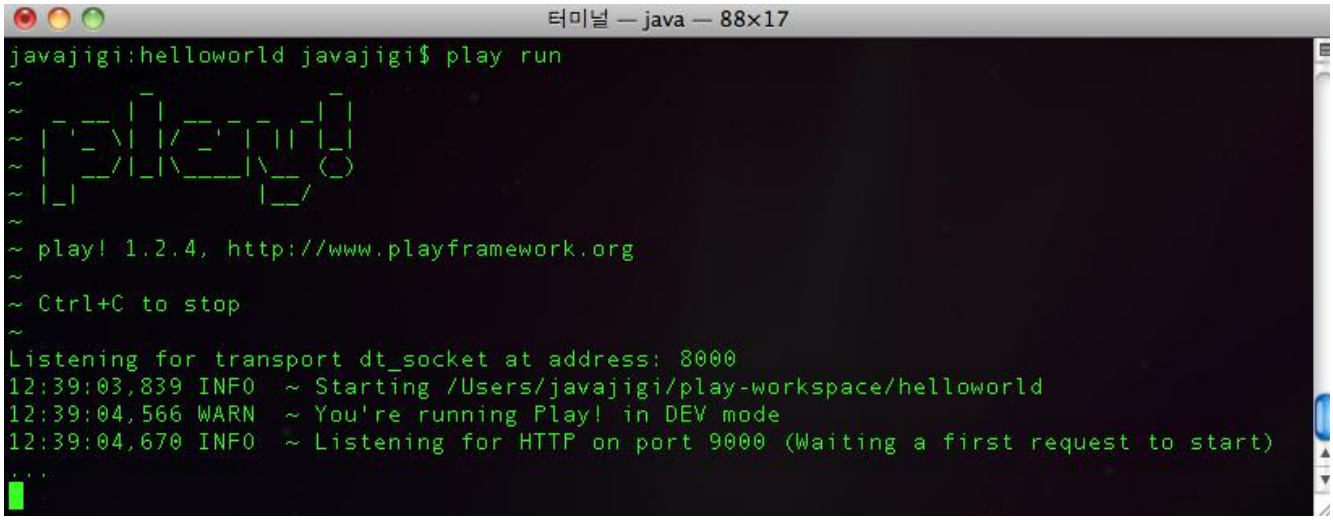
이클립스에서 helloworld 프로젝트를 가져오기(import) 한다.

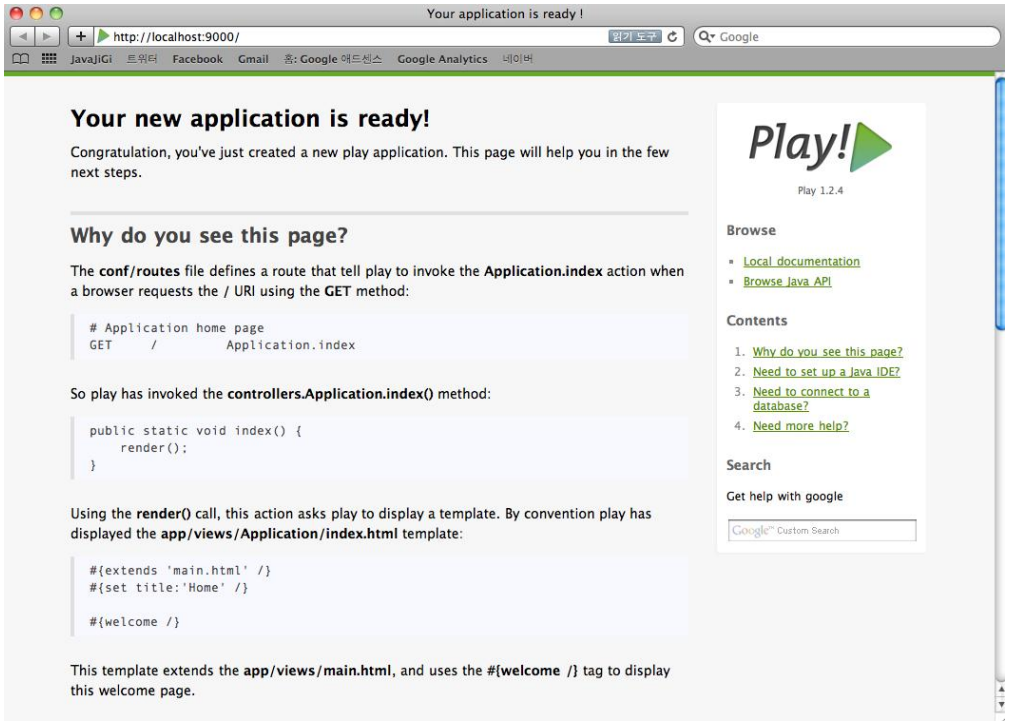


위 그림을 보면 플레이 기반 웹 애플리케이션의 기본 구조를 파악할 수 있다. 이미 레일즈(rails) 애플리케이션 기반으로 개발해본 경험이 있다면 위 디렉토리 구조에 쉽게 적응할 수 있다. 각 디렉토리별 용도를 살펴보면 다음과 같다. conf 디렉토리에는 설정 파일과 국제화(internationalization, i18n) 파일이 위치한다. app 디렉토리는 개발하는 애플리케이션의 핵심이 되는 소스를 포함하는 디렉토리로 models, controllers, views를 하위 디렉토리로 가진다. models 디렉토리는 모델 정의와 관련한 소스 코드를 포함하고, controllers 디렉토리는 비즈니스 로직 관련 소스가 위치한다. views 디렉토리는 HTML과 플레이 템플릿 언어 기반 소스 코드를 포함한다. lib 디렉토리는 애플리케이션을 실행할 때 필요한 jar 라이브러리가 위치하는 곳이며, public 디렉토리는 자바스트립트, CSS, 이미지와 같은 정적인 파일을 포함한다. 마지막으로 test 디렉토리는 모든 종류의 테스트 소스 코드를 포함한다.

helloworld 애플리케이션을 실행한다.

~$ play run





http://localhost:9000으로 접근하면 위 화면을 볼 수 있다. 위 화면이 어떤 과정을 통해 서비스되는지 확인해 보자.

http://localhost:9000으로 접근하면 플레이는 "/"에 해당하는 컨트롤러(Controller) 매핑을 찾는다. 이 설정은 conf/routes 파일에 다음과 같이 구현되어 있다.

GET / Application.index

위 설정은 "/" URL에 HTTP GET으로 호출하는 경우 Application 컨트롤러 클래스의 index() 메소드를 통해 서비스를 한다는 의미이다. Application 컨트롤러 클래스는 app/controllers 디렉토리에 위치한다.

```
public class Application extends Controller {
    public static void index() {
        render();
    }
}
```

위와 같이 컨트롤러를 구현하면 뷰(View) 경로는 app/views/${controller}/{method}.html 규칙으로 결정된다. 따라서 위 "/"URL에 대한 뷰 경로는 app/views/Application/index.html 이다.

index.html 파일 소스 코드는 다음과 같다.

```
#{extends 'main.html' /}
#{set title:'Home' /}

#{welcome /}
```

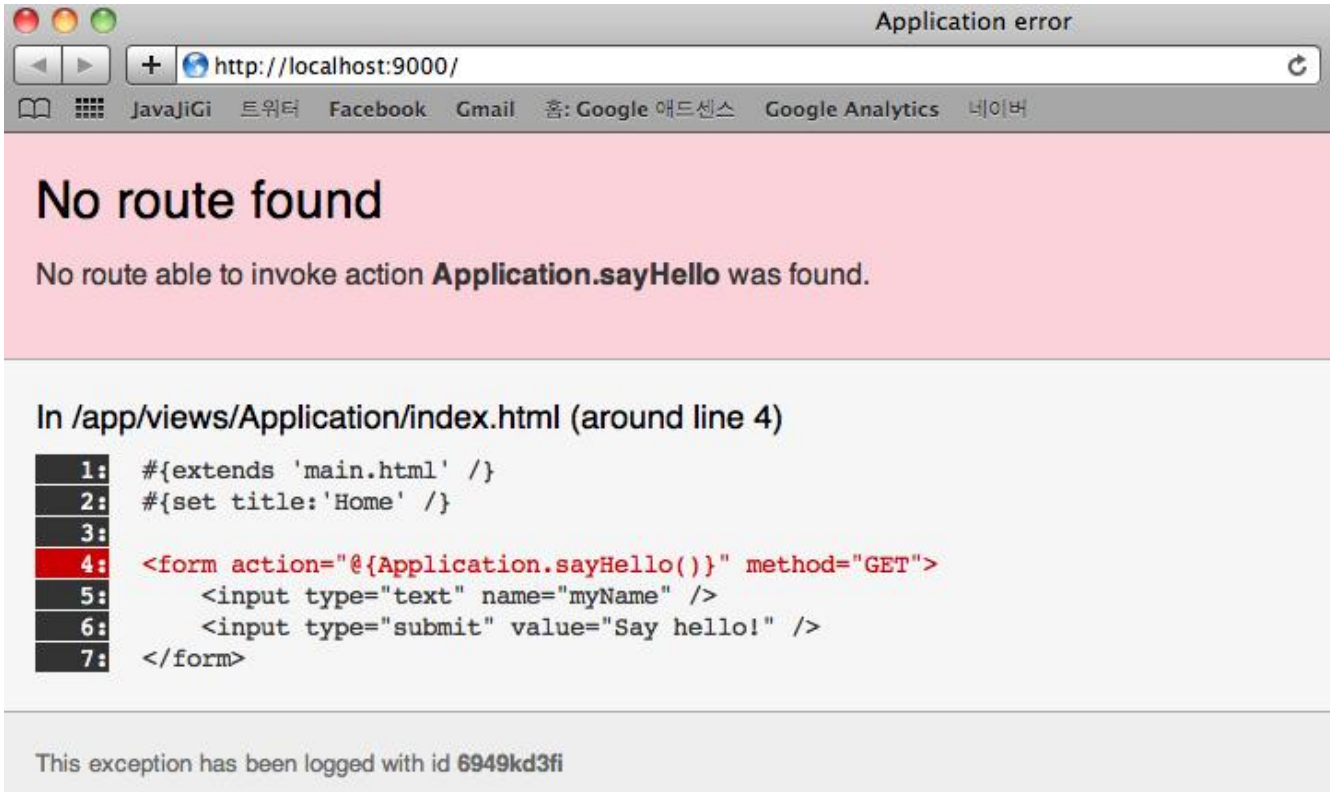
플레이는 이와 같은 구조로 웹 요청에 대한 서비스를 하고 있다. 지금까지 살펴본 과정을 통해“Hello World”를 출력하는 페이지를 개발해 보자.
단순히 “Hello World”를 출력하는 것이 아니라 인덱스 페이지에서 이름을 입력받아 “박재성~ Hello World”라는 메시지를 출력하도록 구현해보겠다.

먼저 index.html 파일에서 이름을 입력할 수 있도록 다음과 같이 구현한다.

```
#{extends 'main.html' /}
#{set title:'Home' /}

<form action="@{Application.sayHello()}" method="GET">
    <input type="text" name="myName" />
    <input type="submit" value="Say hello!" />
</form>
```

위와 같이 소스 코드를 수정하고 http://localhost:9000에 다시 접근하면 다음과 같이 에러 화면을 볼 수 있다.



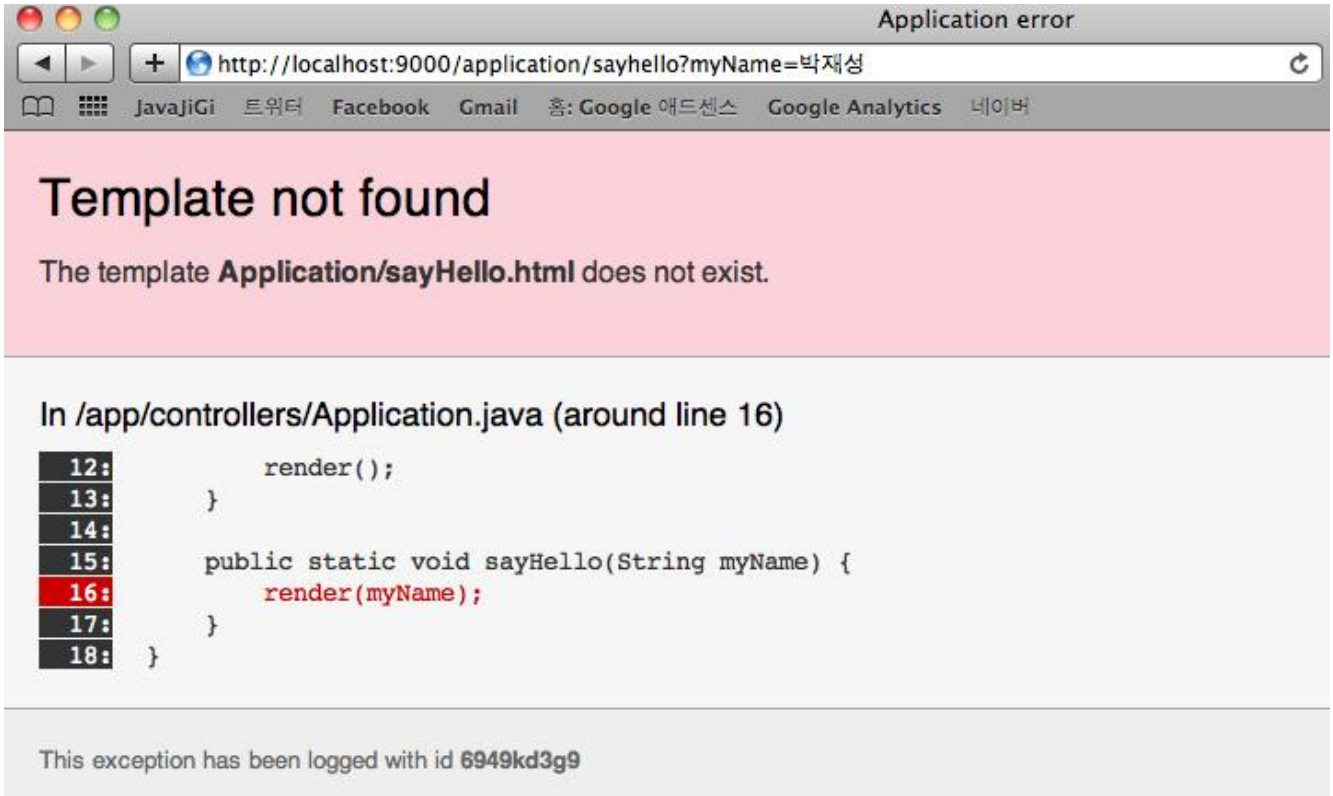
에러가 발생하는 원인은 Application 컨트롤러에 sayHello() 메소드가 존재하지 않기 때문이다. 플레이 장점 중의 하나는 애플리케이션을 개발하는 중에 에러가 발생할 경우 위 화면과 같이 에러의 원인을 쉽게 파악할 수 있다는 것이다.

Application 컨트롤러에 sayHello() 메소드를 추가한다.

```
public class Application extends Controller {
    public static void index() {
        render();
    }

    public static void sayHello(String myName) {
        render(myName);
    }
}
```

Application에 sayHello() 메소드를 추가한 후 http://localhost:9000에 접속하면 위와 같이 이름을 입력할 수 있는 화면이 나타난다. 이름을 입력한 후 다음 단계로 넘어가 보자. 그런데 다시 한번 에러가 발생한다.

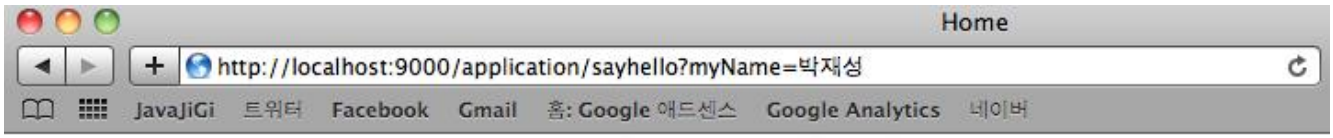


에러 메시지를 확인해보면 app/views/Application/sayHello.html 파일이 존재하지 않기 때문이다. app/views/Application/sayHello.html 파일을 추가한다.

```
#{extends 'main.html' /}
#{set title:'Home' /}

<h1>${myName ?: 'guest'}~ Hello World</h1>

<a href="@{Application.index()}">Back to form</a>
```



박재성~Hello World

Back to form