NaverCafeService.java

```java
package com.example.gingeraebi.navercafe.network;

import ...

public interface NaverCafeService {
    @GET("/contents")
    Call<List<Content>> getContents();

    @POST("/contents")
    Call<Content> saveContent(@Body Content content);

}
```

이번에는 POST를 보내는 방법에 대해서 알아보겠습니다.

NaverCafeService.java

```java
package com.example.gingeraebi.navercafe.network;

import ...

public interface NaverCafeService {
    @GET("/contents")
    Call<List<Content>> getContents();

    @POST("/contents")
    Call<Content> saveContent(@Body Content content);

}
```

지난번 GET하는 방법과 크게 다르지 않습니다.
우리가 만들어 놓은 서비스 인터페이스에 @POST라는 Annotation을 매핑하여 필요한 메서드를 만들어줍시다.

```
public Content saveContent(Content content) {

}
```

그 후 Proxy클래스로 이동하여 메서드를 하나 구현해 주도록 하겠습니다.
이 메서드는 Content를 서버에 저장하라는 네트워크 통신을 위해서 만들어진 메서드 입니다.

```
public Content saveContent(Content content) {
    Call<Content> call = service.saveContent(content);
    try {
        Content savedContent = call.execute().body();
        return savedContent;
    } catch (IOException e) {
        Log.e("Proxy", e.getMessage());
    }

    return null;
}
```

코드는 전과 크게 다르지 않습니다. 다만, 매개변수로 Content가 들어갔을 뿐입니다.

NaverCafeService.java

```java
package com.example.gingeraebi.navercafe.network;

import ...

public interface NaverCafeService {
    @GET("/contents")
    Call<List<Content>> getContents();

    @POST("/contents")
    Call<Content> saveContent(@Body Content content);

}
```

인터페이스에서 Body태그를 달고 Content를 넘겨주겠다고 정의를 해 놓았기 때문에
saveContent 메서드가 실행되는 순간 Content를 자동으로 JSON으로 바꿔서 던져주게 될 것입니다.

```java
public class POSTContentsTest extends AsyncTask<Content, Void, Content> {

    @Override
    protected Content doInBackground(Content... params) {
        Proxy proxy = new Proxy();
        Content savedContent = proxy.saveContent(params[0]);
        Log.i("WriteContentActivity", "SavedContent : " + savedContent.toString());
        return savedContent;
    }
}
```

테스트를 위해 AsyncTask를 상속받은 Task 클래스를 하나 만들고 Parameter로 Content를 넘기도록 하겠습니다.
서버쪽에서 제대로 Content를 저장한다면 다시 그 Content를 되돌려주도록 작성 할 것이기 때문에
Return Type또한Content로 만들었습니다.

```java
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.button_writeContent_send:
            Intent intent = new Intent(this, ContentListActivity.class);
            Log.i("TEST",content.toString());

            Content content = exportContnet();

            dao.insertContent(content);
            new POSTContentsTest().execute(content);
            startActivity(intent);
            finish();
    }
}
```

이 코드는 ContentWriteActivity에 작성되었으며 SEND버튼을 눌렀을 때 post를 보내는 것처럼 구현하였습니다.