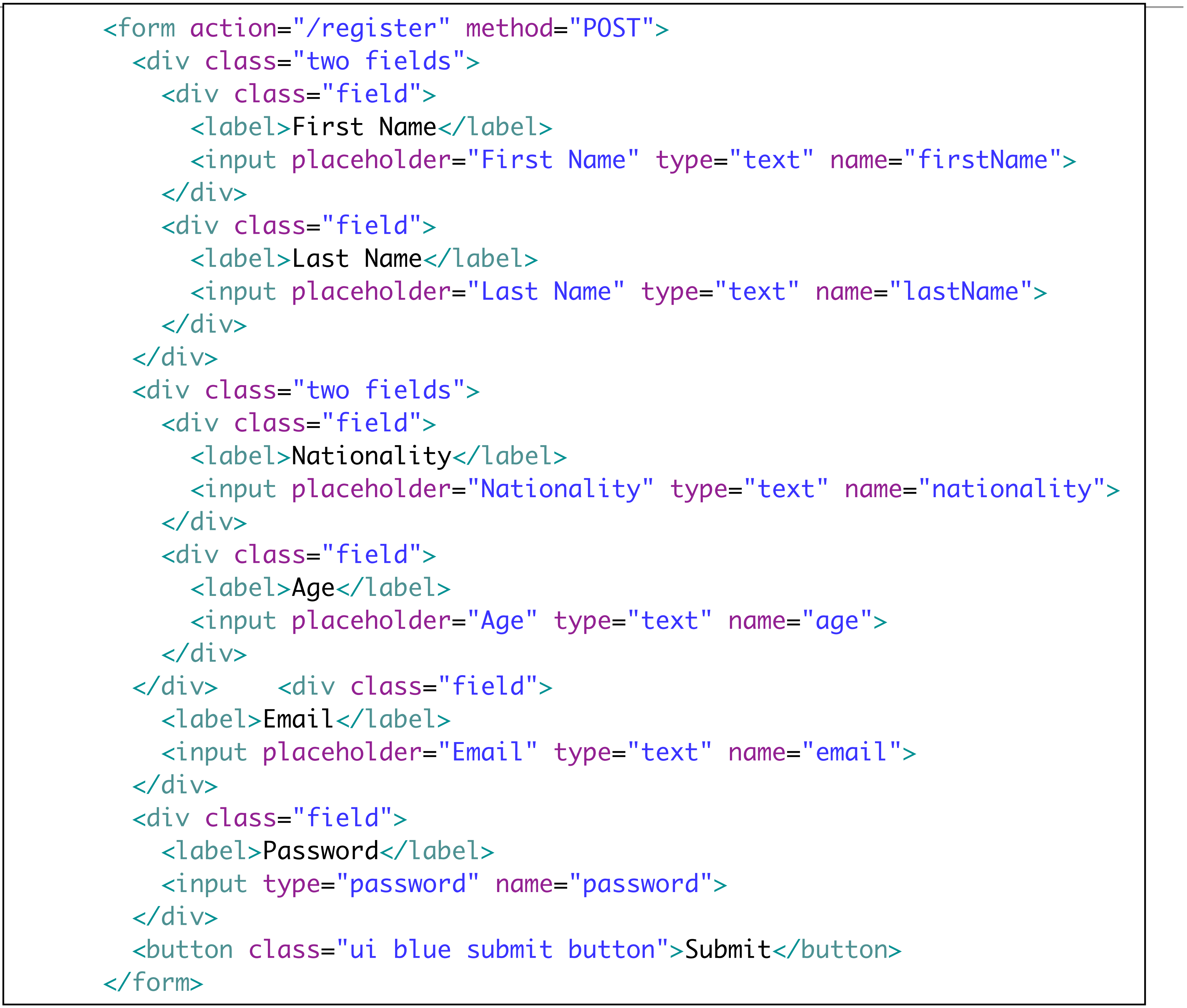
Assignments 2 Tips & Tricks

Simplified Parameter Passing

- ‘name’ field specified in ‘input’ paramater is matched to parameter to action

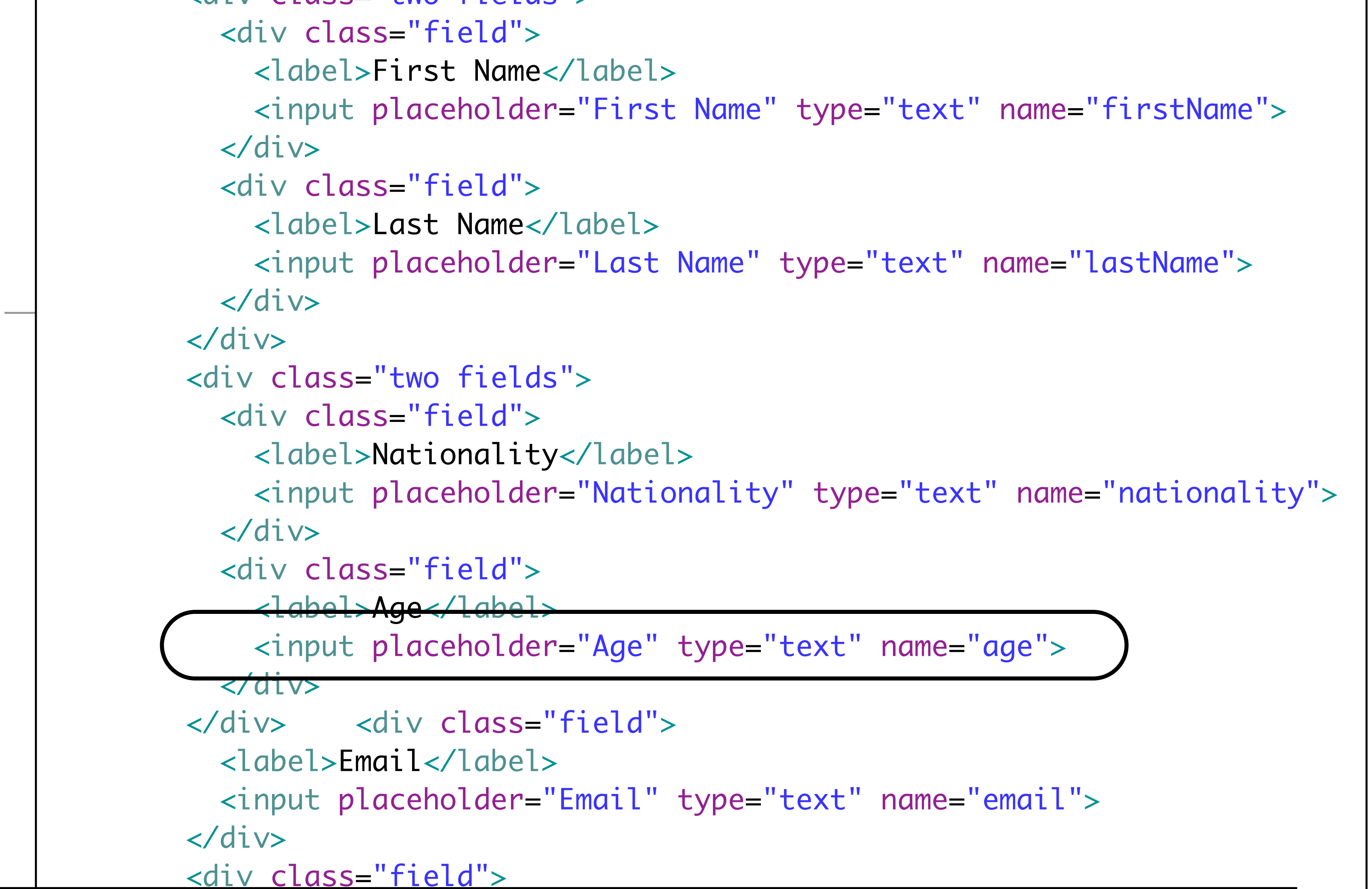


```
<form action="/register" method="POST">
  <div class="two fields">
    <div class="field">
      <label>First Name</label>
      <input placeholder="First Name" type="text" name="firstName">
    </div>
    <div class="field">
      <label>Last Name</label>
      <input placeholder="Last Name" type="text" name="lastName">
    </div>
  </div>
  <div class="two fields">
    <div class="field">
      <label>Nationality</label>
      <input placeholder="Nationality" type="text" name="nationality">
    </div>
    <div class="field">
      <label>Age</label>
      <input placeholder="Age" type="text" name="age">
    </div>
  </div>    <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name="email">
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name="password">
  </div>
  <button class="ui blue submit button">Submit</button>
</form>
```

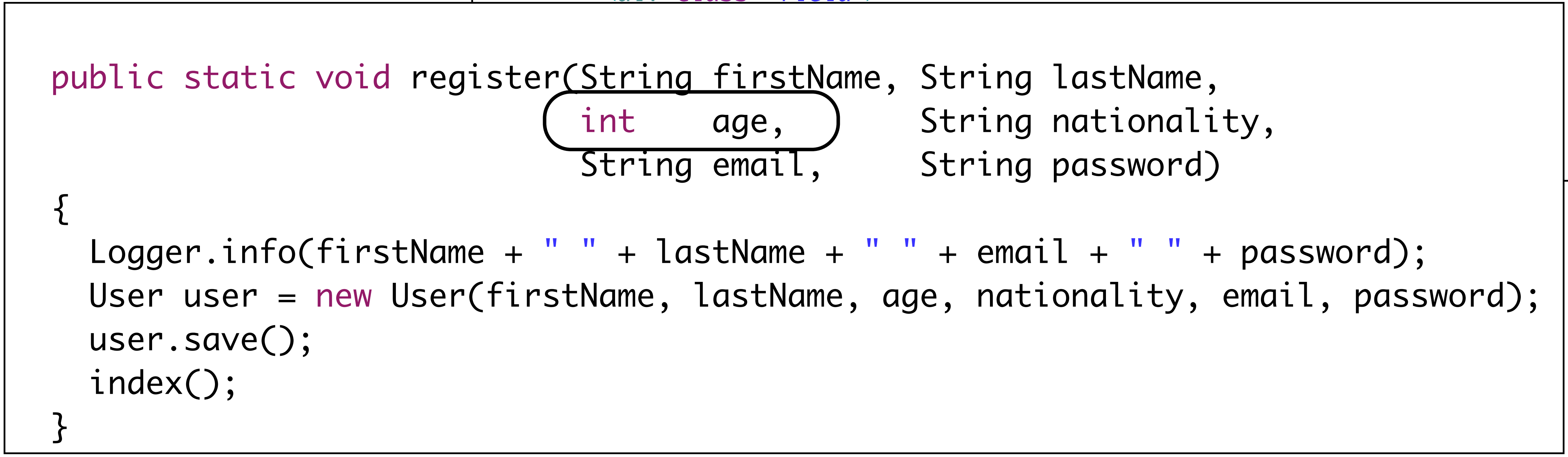
- Values are copied to the correctly named paramater

```
public static void register(String firstName, String lastName,
                            int    age,       String nationality,
                            String email,     String password)
{
  Logger.info(firstName + " " + lastName + " " + email + " " + password);
  User user = new User(firstName, lastName, age, nationality, email, password);
  user.save();
  index();
}
```

- Example: ‘age’
  - ‘name’ attribute in html element must be same as age parameter in action



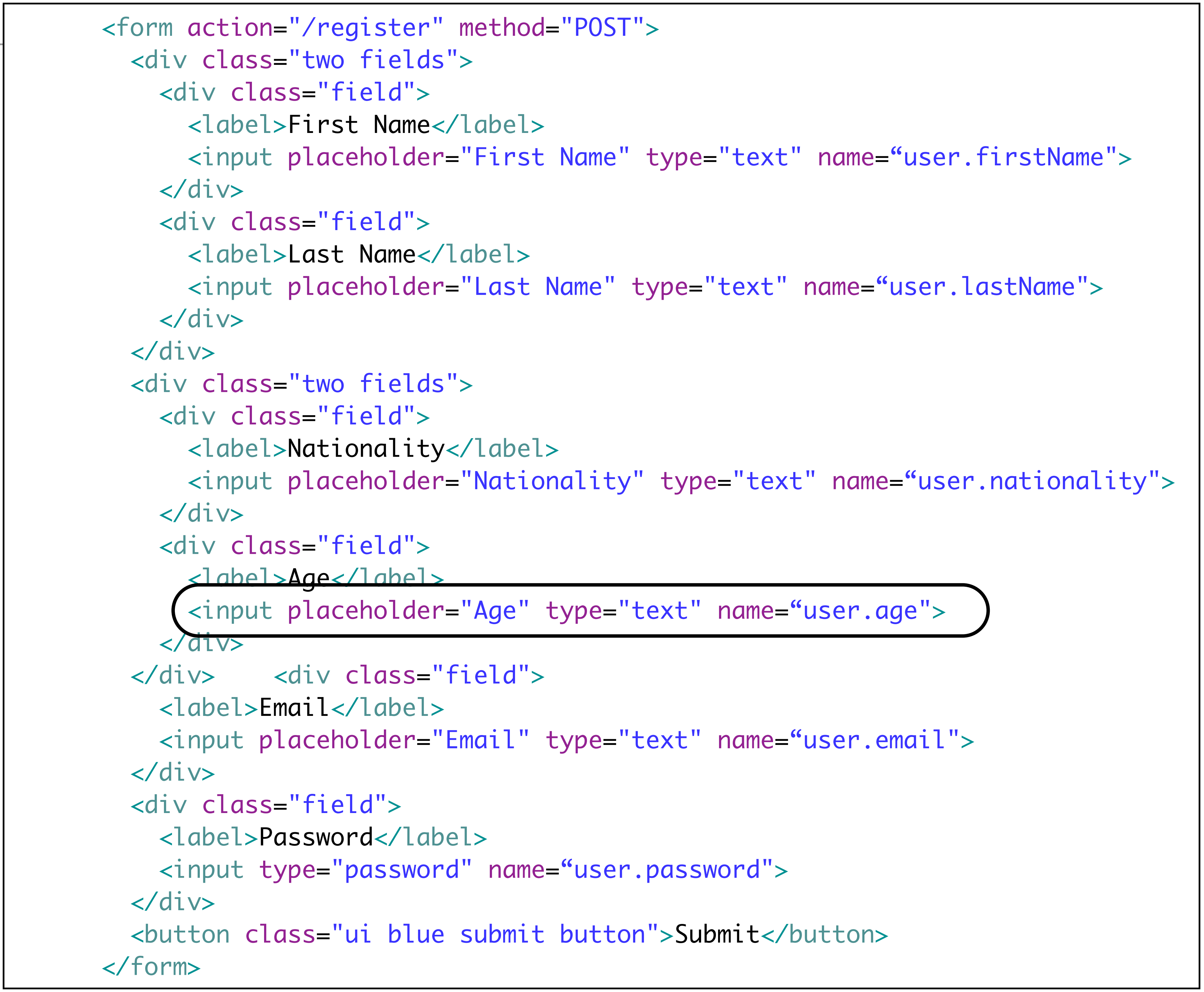
```
<div class="two fields">
  <div class="field">
    <label>First Name</label>
    <input placeholder="First Name" type="text" name="firstName">
  </div>
  <div class="field">
    <label>Last Name</label>
    <input placeholder="Last Name" type="text" name="lastName">
  </div>
</div>
<div class="two fields">
  <div class="field">
    <label>Nationality</label>
    <input placeholder="Nationality" type="text" name="nationality">
  </div>
  <div class="field">
    <label>Age</label>
    <input placeholder="Age" type="text" name="age">
  </div>
</div>    <div class="field">
  <label>Email</label>
  <input placeholder="Email" type="text" name="email">
</div>
<div class="field">
```



```
public static void register(String firstName, String lastName,
                            int    age,       String nationality,
                            String email,     String password)
{
  Logger.info(firstName + " " + lastName + " " + email + " " + password);
  User user = new User(firstName, lastName, age, nationality, email, password);
  user.save();
  index();
}
```

Simplification

- Prefix each name attribute with ‘user.’



```
<form action="/register" method="POST">
  <div class="two fields">
    <div class="field">
      <label>First Name</label>
      <input placeholder="First Name" type="text" name=“user.firstName">
    </div>
    <div class="field">
      <label>Last Name</label>
      <input placeholder="Last Name" type="text" name=“user.lastName">
    </div>
  </div>
  <div class="two fields">
    <div class="field">
      <label>Nationality</label>
      <input placeholder="Nationality" type="text" name=“user.nationality">
    </div>
    <div class="field">
      <label>Age</label>
      <input placeholder="Age" type="text" name=“user.age">
    </div>
  </div>    <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name=“user.email">
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name=“user.password">
  </div>
  <button class="ui blue submit button">Submit</button>
</form>
```

```
public static void register(User user)
{
  Logger.info(user.firstName + " " + user.lastName + " " + user.email + " " + user.password);
  user.save();
  index();
}
```

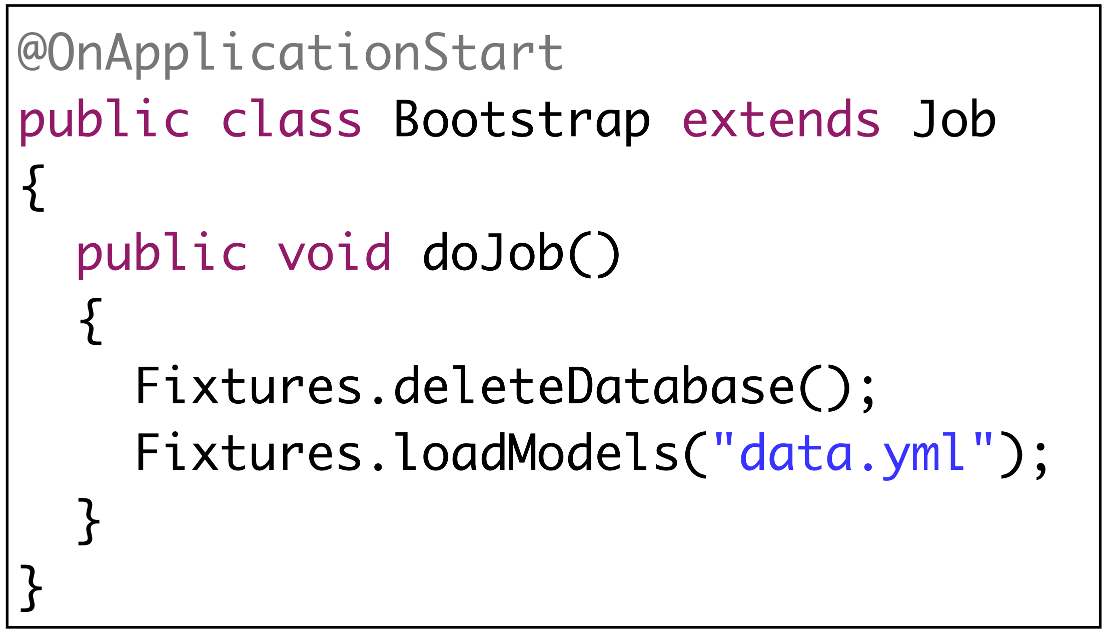
- Action now take an object!
- Fields already filled with data from form

```
      <div class="field">
        <label>First Name</label>
        <input placeholder="First Name" type="text" name=“user.firstName">
      </div>
      <div class="field">
        <label>Last Name</label>
        <input placeholder="Last Name" type="text" name=“user.lastName">
      </div>
    </div>
    <div class="two fields">
      <div class="field">
        <label>Nationality</label>
        <input placeholder="Nationality" type="text" name=“user.nationality">
      </div>
      <div class="field">
        <label>Age</label>
        <input placeholder="Age" type="text" name=“user.age">
      </div>
    </div>    <div class="field">
      <label>Email</label>
      <input placeholder="Email" type="text" name=“user.email">
    </div>
    <div class="field">
      <label>Password</label>
      <input type="password" name=“user.password">
    </div>
    <button class="ui blue submit button">Submit</button>
  </form>
```

```
public static void register(User user)
{
  Logger.info(user.firstName + " " + user.lastName + " " + user.email + " " + user.password);
  user.save();
  index();
}
```

Preloading Images

- Images cannot loaded from the data.yaml file
- Recall, the yaml file is loaded in bootstrap:
- We can intervene here, and load specific extra information - including images!



```java
@OnApplicationStart
public class Bootstrap extends Job
{
  public void doJob()
  {
    Fixtures.deleteDatabase();
    Fixtures.loadModels("data.yml");
  }
}
```

```java
@OnApplicationStart
public class Bootstrap extends Job
{
  public void doJob() throws FileNotFoundException
  {
    Fixtures.deleteDatabase();
    Fixtures.loadModels("data.yml");

    String photoName = "homer.gif";
    Blob blob = new Blob ();
    blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));
    User homer = User.findByEmail("homer@simpson.com");
    homer.profilePicture = blob;
    homer.save();
  }
}
```

Preloading Images Example

The image file must be in the folder of your app

```
    String photoName = "homer.gif";
```

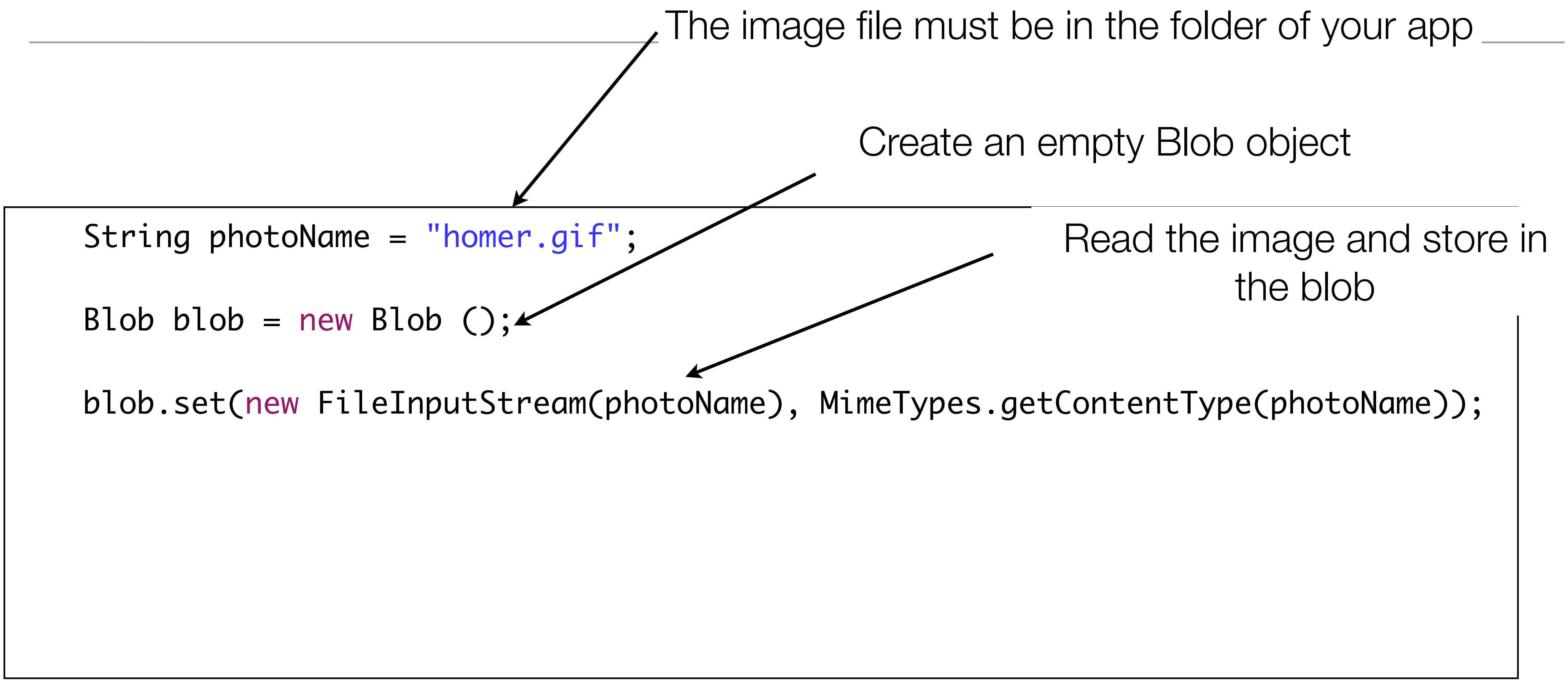
Preloading Images Example

The image file must be in the folder of your app

Create an empty Blob object

```
    String photoName = "homer.gif";

    Blob blob = new Blob ();
```

Preloading Images Example



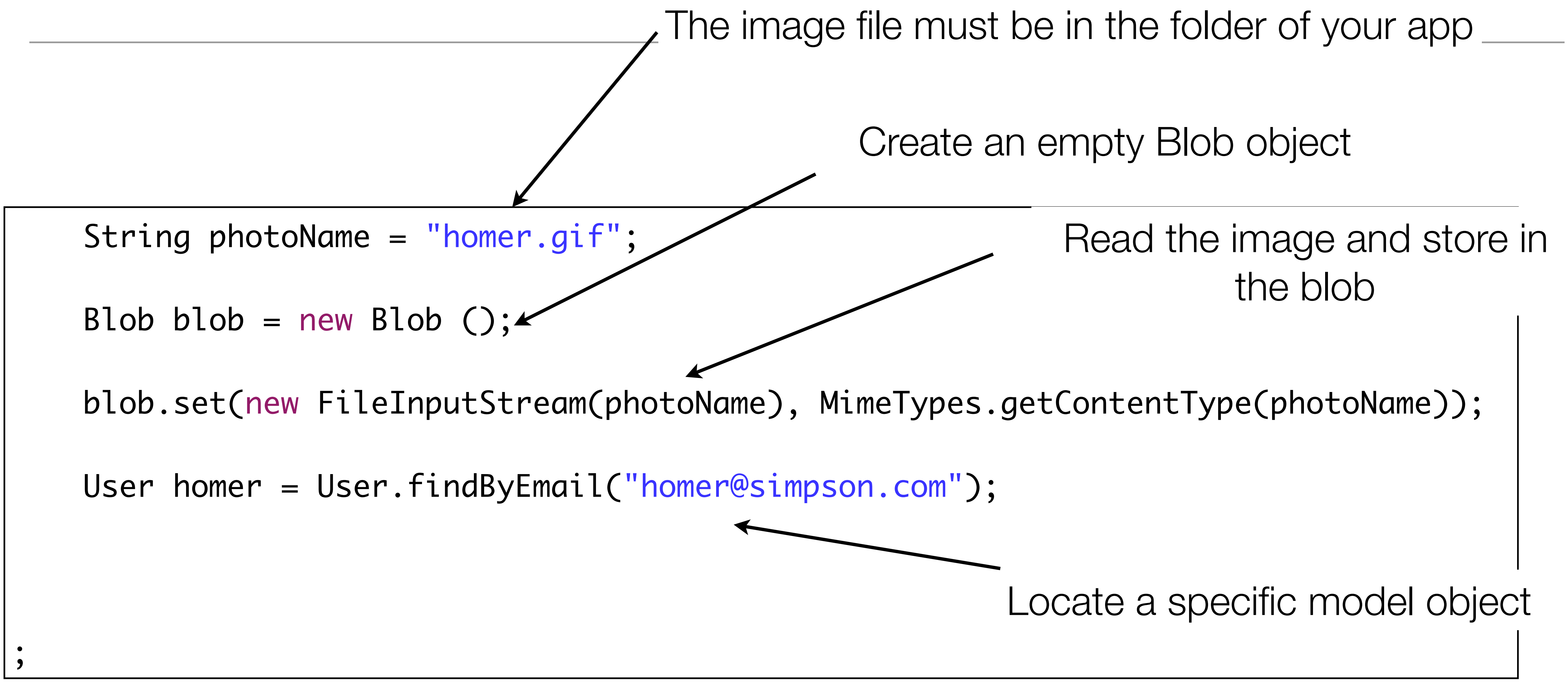
The image file must be in the folder of your app

Create an empty Blob object

Read the image and store in the blob

```
    String photoName = "homer.gif";

    Blob blob = new Blob ();

    blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));
```

Preloading Images Example



The image file must be in the folder of your app

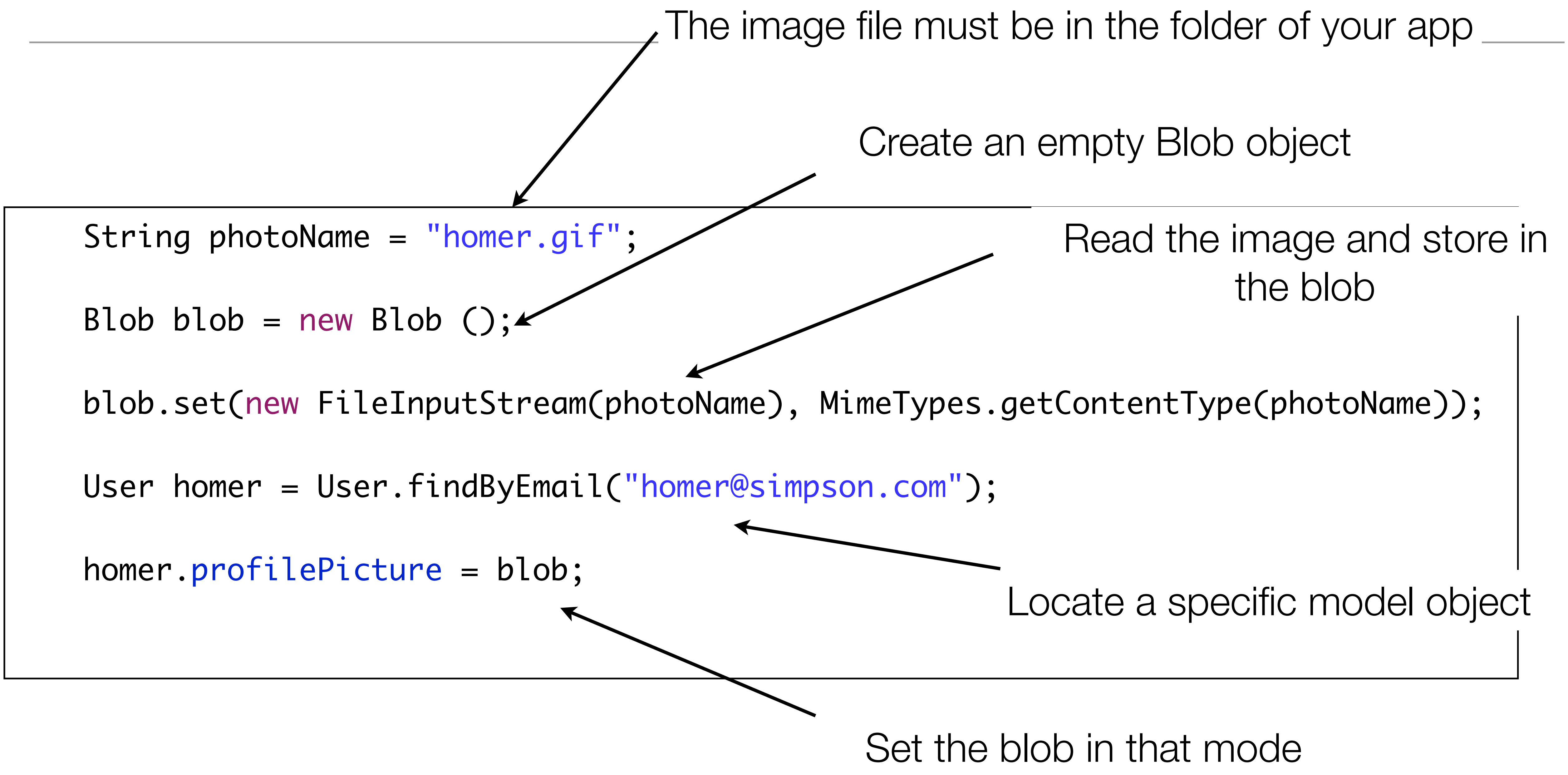
Create an empty Blob object

Read the image and store in the blob

Locate a specific model object

```
    String photoName = "homer.gif";

    Blob blob = new Blob ();

    blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));

    User homer = User.findByEmail("homer@simpson.com");


;
```

Preloading Images Example



The image file must be in the folder of your app

Create an empty Blob object

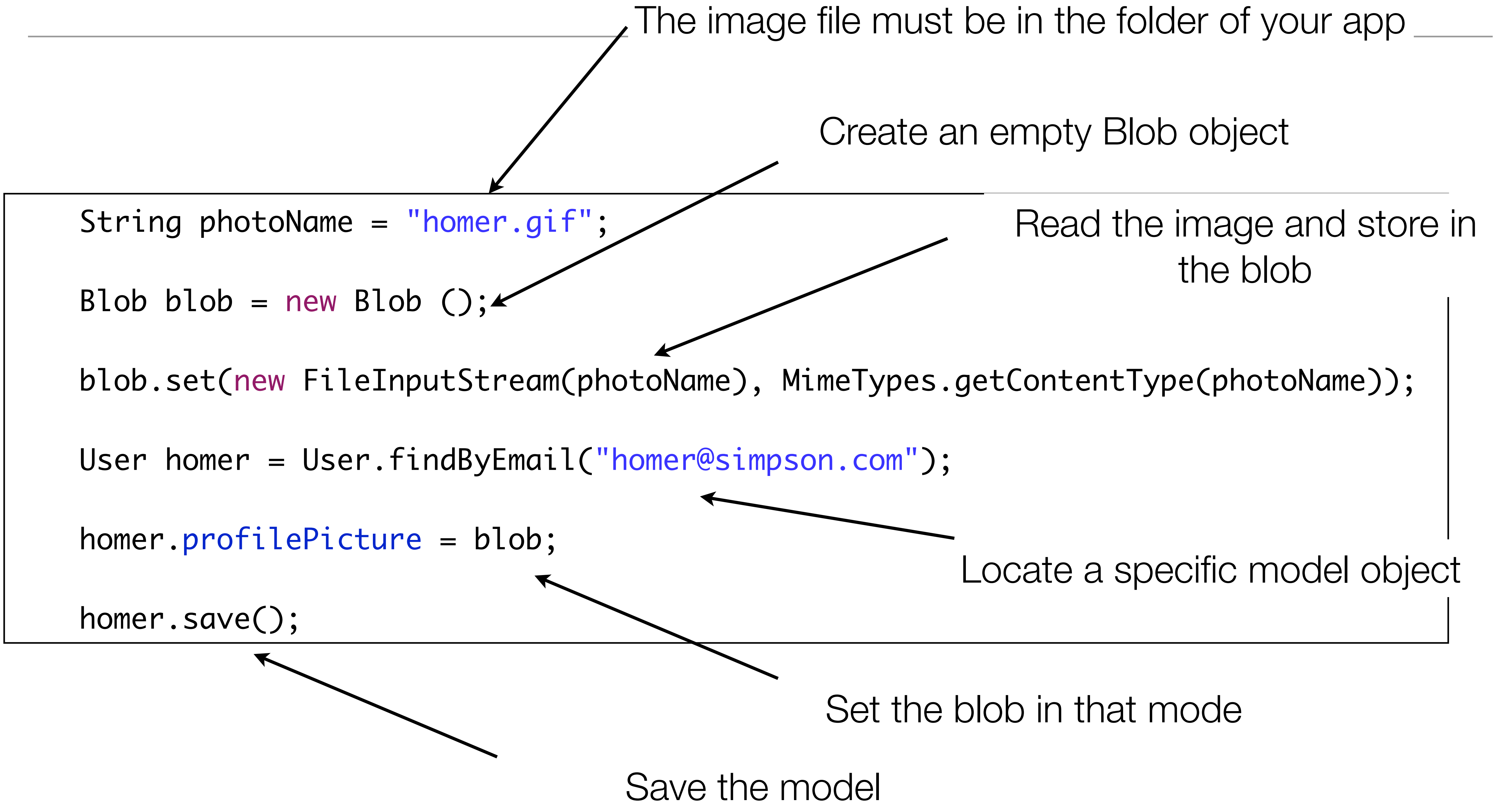
```
String photoName = "homer.gif";

Blob blob = new Blob ();

blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));

User homer = User.findByEmail("homer@simpson.com");

homer.profilePicture = blob;
```

Read the image and store in the blob

Locate a specific model object

Set the blob in that mode

Preloading Images Example



```
String photoName = "homer.gif";

Blob blob = new Blob ();

blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));

User homer = User.findByEmail("homer@simpson.com");

homer.profilePicture = blob;

homer.save();
```

The image file must be in the folder of your app

Create an empty Blob object

Read the image and store in the blob

Locate a specific model object

Set the blob in that mode

Save the model