Laravel for Beginners 09

Error Handling and Validation

Error Handling – avoid exceptions

If you want to show a song that doesn't exist, your application will crash with an exception! Always try to avoid a situation where an exception can occur!

→ check first: if ($song != null) …
→ send the error to the view
   via the method
withErrors('Your message goes here');

```php
public function show($id)
{
    // if $id is not valid, find returns null
    $song = Song::find($id);

    // check if $song is null
    if ($song != null)
    {
        return view('songs.show')->with('song', $song);
    }
    else
    {
        // send an error via ->withErrors('your message')
        return redirect()->route('songs.index')
            ->withErrors('No song with id=' . $id . ' found!');
    }
}
```

Error Handling – handling exceptions

If you you cannot avoid an exception, surround the critical part with try/catch!

→ send the error to the view
 via the method
 withErrors('Your message goes here');

Add the following code to your view
to display all errors:

```
@if($errors)
    @foreach ($errors->all() as $error)
        <div>{{ $error }}</div>
    @endforeach
@endif
```

```php
public function store(Request $request)
{
    try
    {
        $song = new Song();
        $song->title = $request->title;
        $song->artist = $request->artist;
        $song->album = $request->album;
        $song->save();

        return redirect()->route('songs.index');
    }
    catch (Exception $ex)
    {
        return redirect()->route('songs.index')
            ->withErrors('Cannot create song!');
    }
}
```

Validate the request parameters

- Before you store a new song check the request parameters!
- Insert this code at the beginning of the store() and update() method:

```
$this->validate($request,
    [
    'title' => 'required|max:150',
    'artist' => 'required|max:160',
    'album' => 'required|max:130'
    ]
);
```

- If the validation fails the controller automatically returns the last view with the validation errors!