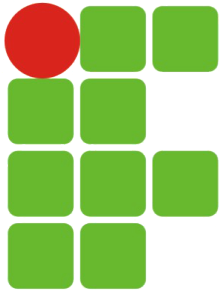


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE
Campus Natal - Central

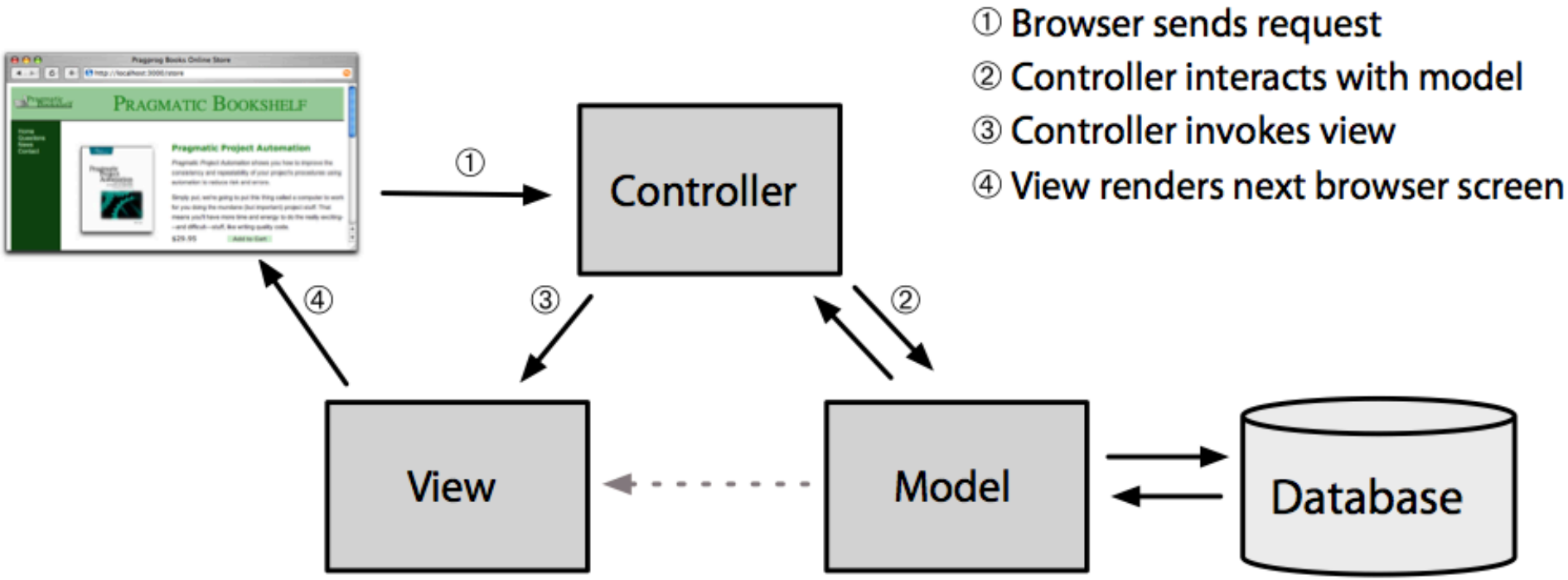
Arquitetura de Aplicações Rails

Prof. Fellipe Aleixo (fellipe.aleixo@ifrn.edu.br)

Models, *Views* e *Controllers*

- Modelo arquitetural – aplicações interativas
- Responsabilidades bem definidas
  - *Model* – manter o estado da aplicação e implementar as regras de negócio associadas
  - *View* – gerar a interface com o usuário, baseada em informações dos elementos do modelo
  - *Controller* – "orquestram" a aplicação, recebem os eventos externos e repassam aos elementos de modelo apropriados, e o resultado para uma visão

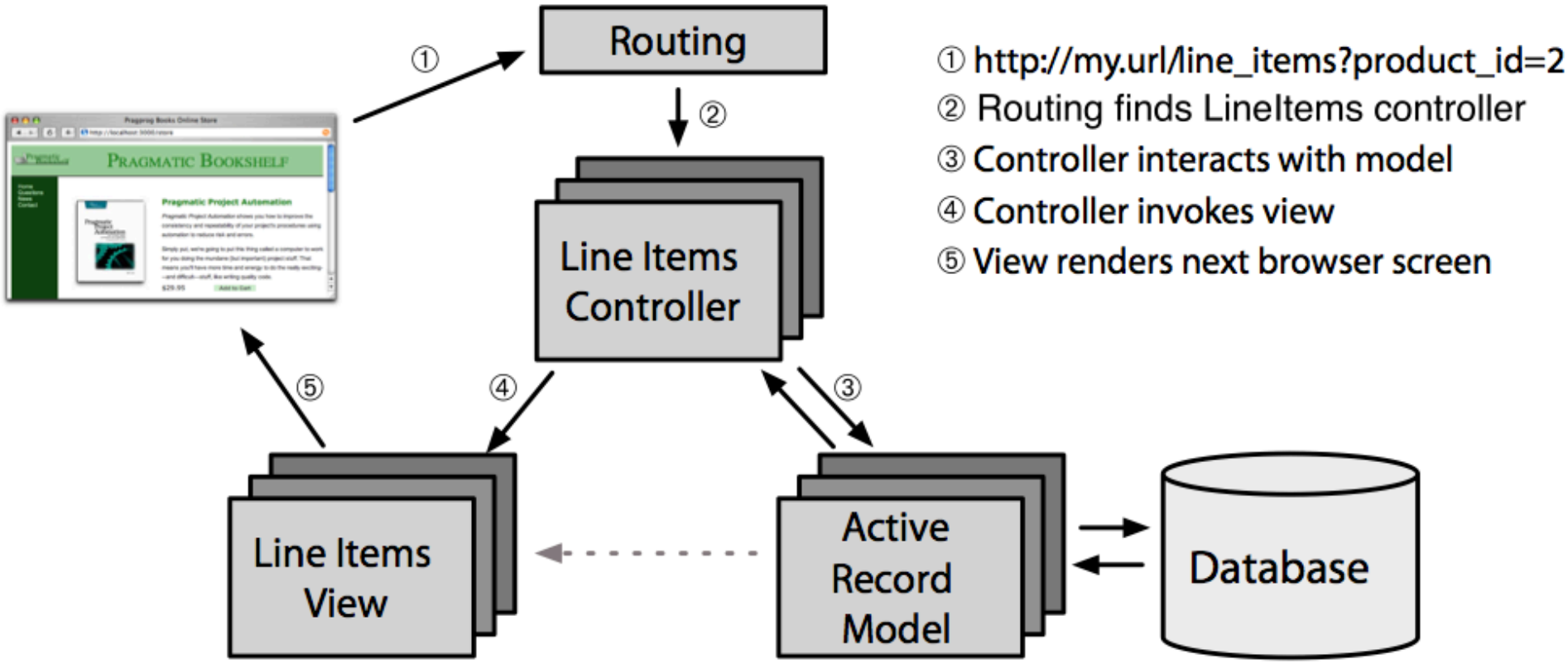
Models, Views e *Controllers*



Contexto Ruby on Rails

- Trata-se de um framework MVC
- São desenvolvidos os elementos de modelo, controle e visão de forma separada
- Filosofia Rails: convenção sobre configuração

Tratando Requisições



Requisições

- Cada requisição possui
  - Um caminho (definido pela URL)
    - O qual pode referenciar um controlador
  - Um método
    - Os mais comuns são: GET, POST, PUT, PATCH e DELETE
    - Por convenção o método POST são associados às ações “create()” – o qual, trata a requisição do usuário
    - No exemplo anterior, identificar o produto de id “2” e adicionar no mesmo no “carrinho de compras”

Elementos de suporte ao

MODELO RAILS

Mapeamento Objeto-Relacional

- Mapeamento de tabelas de banco de dados para classes de objetos (ORM)
  - Colunas mapeadas em atributos
- As classes que encapsulam tabelas de banco oferecem métodos que abstraem as ações relacionadas à referida tabela
  - Objetos representando registro de banco oferecem métodos para operar com o mesmo

Active Record

- Trata-se de uma camada ORM disponibilizada com o Rails
  - Opção pré-configurada
  - Na definição de uma classe de modelo, basta
    - Importar a biblioteca “active_record”
    - Estender uma classe base do *Active Record*
    - Definir dados necessários à conexão com o banco

Action Pack

- Provê as funções de *view* e *controller*
  - Dado a intima relação entre tais componentes
  - Implementação compartimentalizada

- A visão, geralmente, fica responsável por criar a resposta que será exibida no navegador
  - Formato HTML – conteúdo estático e dinâmico
  - Conteúdo dinâmico gerado por *templates*

Templates

- O mais comum esquema de *Templates* é o
  - ERB (*Embedded Ruby*)
    - Possibilita incluir código Ruby dentro do código HTML
    - Pode ser perigoso – inclusão de código que deveria estar no *controller* ou mesmo no *model*
    - Permite a construção de fragmentos Javascript (AJAX)
- Rails também disponibiliza um construtor XML

Controle

- Um *controller* Rails representa o “centro lógico” da aplicação
  - Coordena a interação – usuário, *view* e *model*
  - Roteia as requisições para as ações internas
  - Gerencia as sessões
  - Gerencia a “cache”, para aumentar performance
  - Gerencia os módulos auxiliares (*helpers*)