Indo além: RESTFull

Laboratório de Programação

Até então:

- Usamos `GET` do HTTP
  - Serve para obter dados. Show

> “
>
> Quais as outras opções?
>
> ”

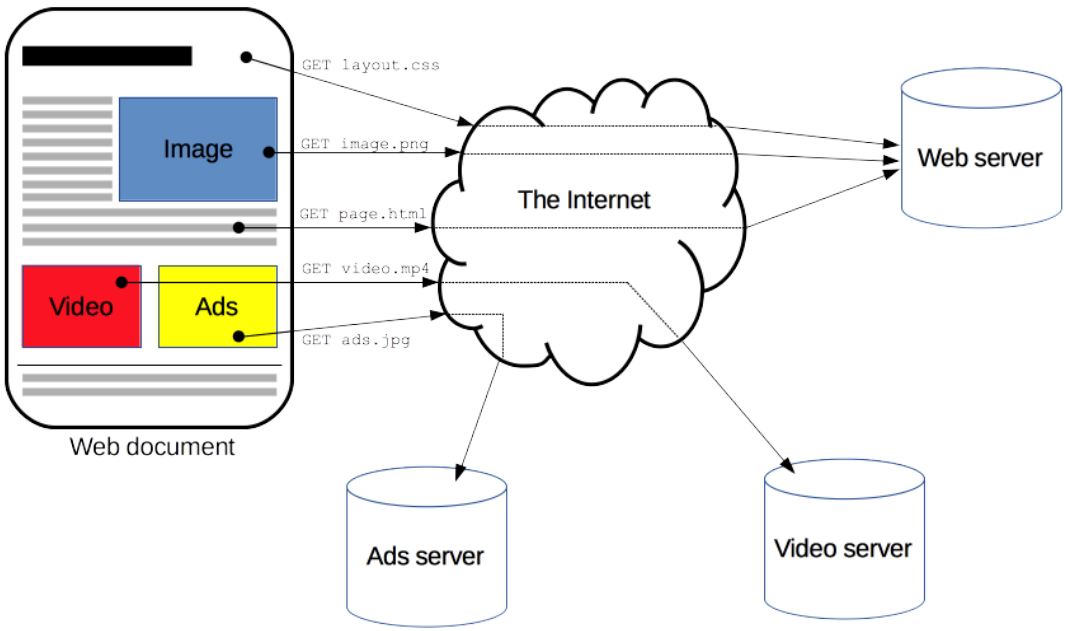
HTTP

> “
>
> Hypertext Transfer Protocol:
>
> ”

- Clientes e servidores se comunicam trocando mensagens individuais (em oposição a um fluxo de dados).
  - As mensagens enviadas pelo cliente, geralmente através de navegador da Web, são chamadas de `requests`
  - As mensagens enviadas pelo servidor como resposta são chamadas de `responses`.

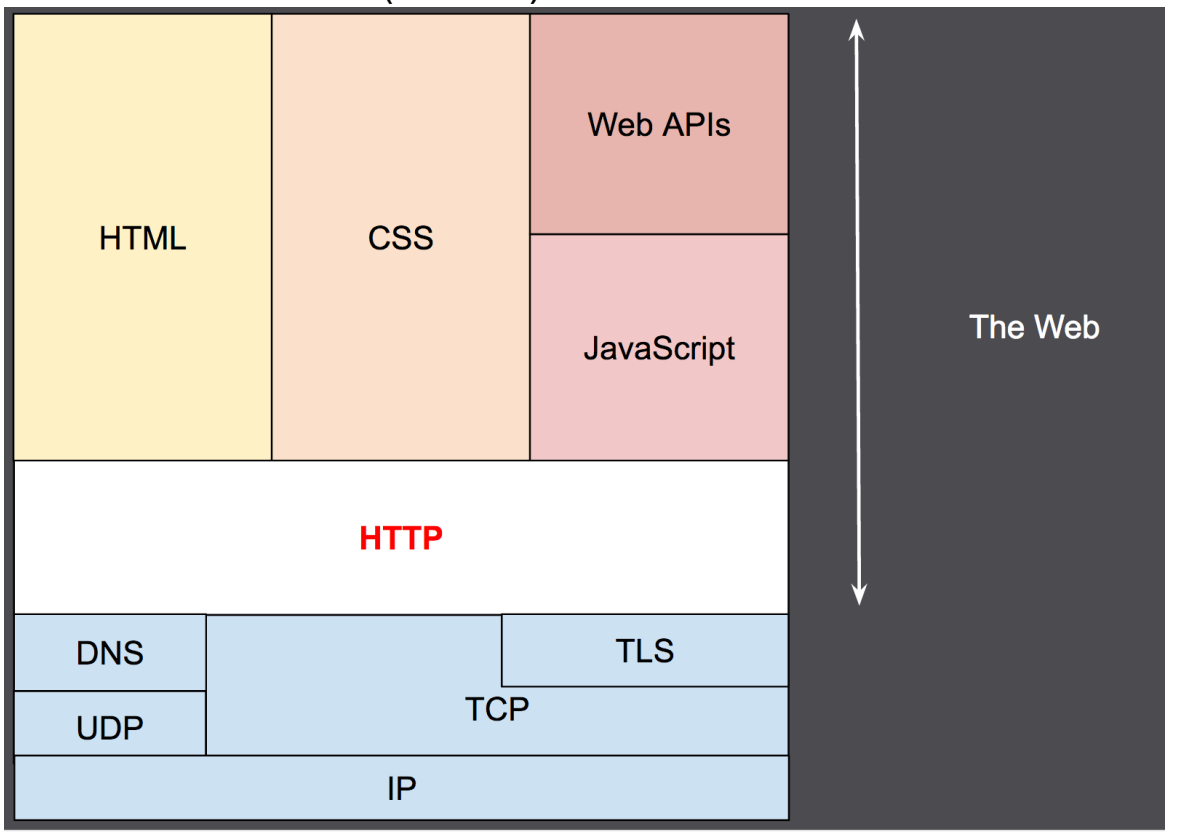
HTTP

> “
>
> Hypertext Transfer Protocol:
>
> ”



HTTP - Camada

- Enviado via TCP ou TLS (HTTPS)



HTTP - Componentes

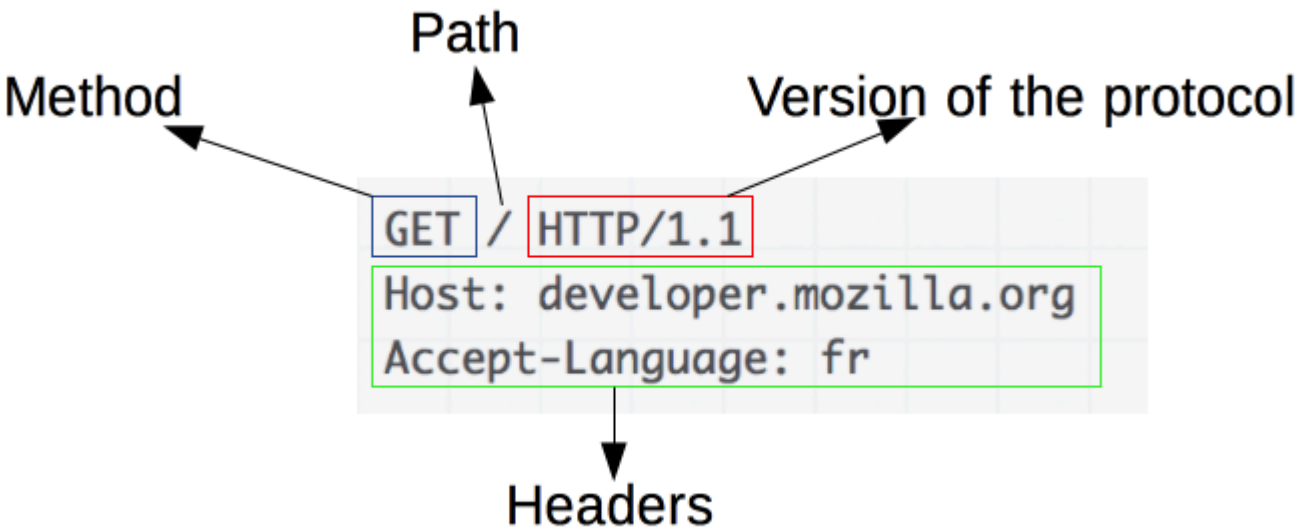
- Cliente: `user-agent`
  - quem inicializa a `request`
  - Ex: algum navegador
- Proxies: controla o fluxo
  - cache, filtro, balance, autenticação, logging
- Web Server
  - Quem `reponse` a `request`
  - Apache, Jetty, Nginx ...

HTTP - Características

- Simples
- Extensível
- Sem estado, mas com sessão

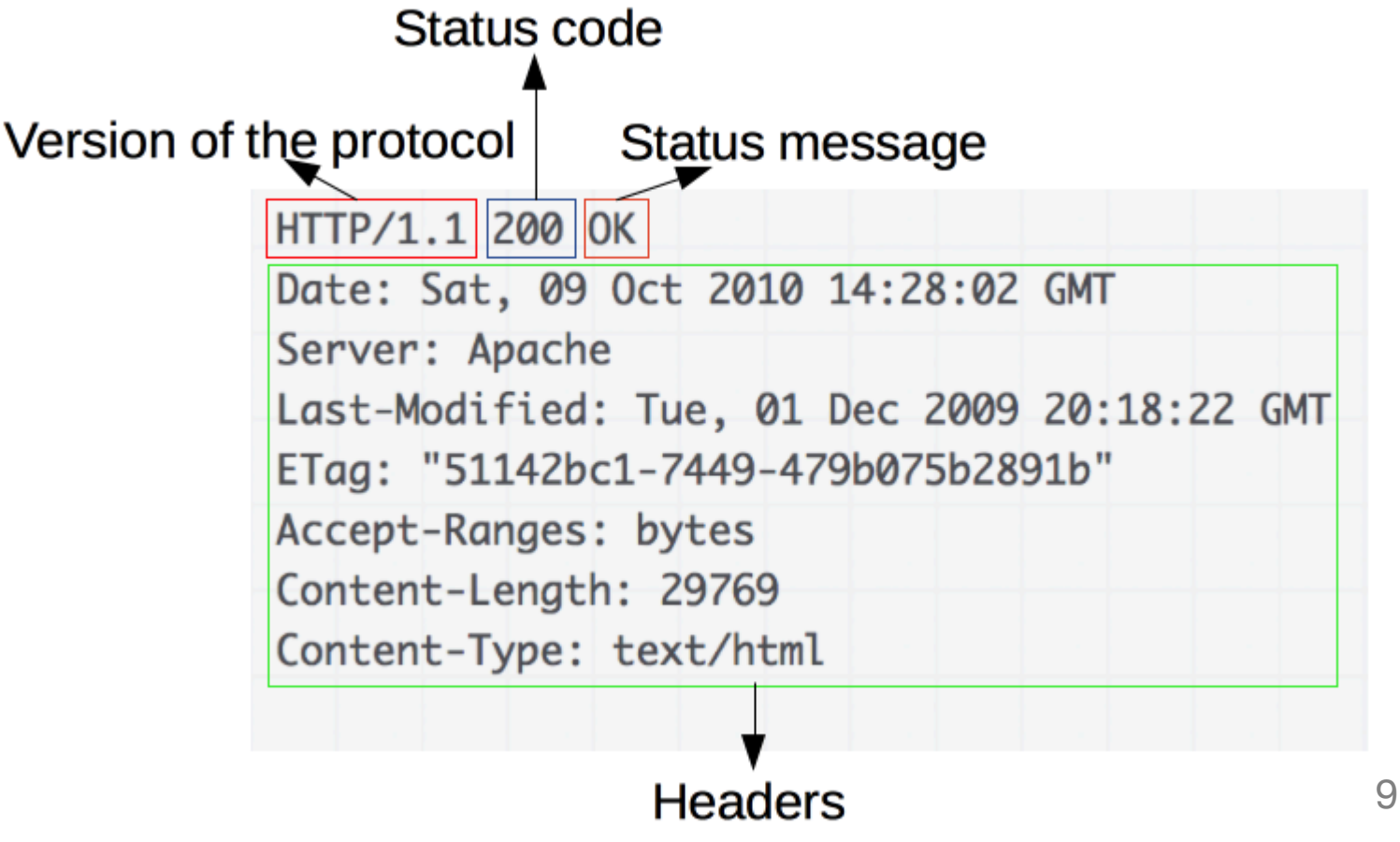
HTTP - Mensagens

- Requests



HTTP - Mensagens

- Response



HTTP - Métodos

- `GET`: requisita um recurso
- `HEAD`: idêntico ao `GET` mas sem corpo
- `POST`: usado para enviar um recurso
- `PUT`: usado para substituir o estado de um recurso
- `DELETE`: usado para remover um recurso
- `CONNECT`: estabele um tunel entre servidor e recurso
- `OPTIONS`: descreve uma comunicação
- `TRACE`: teste de lopp-back
- `PATCH`: usado para aplicar modificações parciais a um objeto

HTTP - Códigos Comuns

Categoria	Descrição
1xx: Informativo	Comunica informações no nível do protocolo de transferência.
2xx: Success	Indica que a solicitação do cliente foi aceita com sucesso.
3xx: Redirecionamento	Indica que o cliente deve executar alguma ação adicional para concluir sua solicitação.
4xx: Erro do cliente	Esta categoria de códigos de status de erro aponta o dedo para os clientes.
5xx: Erro do servidor	O servidor assume a responsabilidade por esses códigos de status de erro.

RESTFull - Métodos

Método	Num conjunto	Num objeto
POST	201 (criado)	404 (não encontrado), 409 (conflito)
GET	200 (OK)	200 (OK), 404 (não encontrado)
PUT	405 (método não permitido)	200 (OK), 204 (sem dados), 404 (não encontrado)
DELETE	405 (método não permitido)	200 (OK), 404 (não encontrado)

RESTFull com Spring Boot

Nossa agenda:

- Construir um projeto (com vcs)
- Definir um objeto básico
- Construir objeto exemplo com todos os métodos
- Testar

Objeto básico:

```
package main.java.com.example.restfull;

public class Curso {
    private String id;
    private String nome;

    public Curso(String id, String nome) {
        this.nome = nome;
        this.id = id;
    }

    public String getId () {
        return id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void setId(String id) {
        this.id = id;
    }
```

Mas seria legal

- Armazenar esse objeto num banco de dados, mesmo que em memória `H2`
  - Para isso usamos `@Entity`
  - O objeto precisa de um construtor padrão vazio ...

```
@Entity
public class Curso {
    @Id
    private String id;
    private String nome;

    public Curso() {

    }
...
```

Precisamos configurar o banco na aplicação:

- src/main/resources/application.properties

```
server.port=8080
spring.h2.console.enabled=true
spring.datasource.platform=h2
spring.datasource.url=jdbc:h2:mem:curso
```

E finalmente implementar um `Repository`

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface CursoRepo extends JpaRepository<Curso, String>{

}
```

Agora sim, construir métodos:

```
@RestController
public class CursoController {
    //associa automaticamente uma implementação
    @Autowired
    private CursoRepo repo;
```

Agora sim, construir métodos:

```
@RestController
public class CursoController {

    @Autowired
    private CursoRepo repo;

    @RequestMapping("/cursos")
    public List<Curso> obterCursos() {
        return repo.findAll();
    }
```

Agora sim, construir métodos:

```
@RestController
public class CursoController {

    @Autowired
    private CursoRepo repo;

    @RequestMapping("/cursos")
    public List<Curso> obterCursos() {
        return repo.findAll();
    }
    @RequestMapping("/curso")
    public Optional<Curso> obterCursosPorId(@RequestParam("key")
                                            String key) {
        return repo.findById(key);
    }
    @RequestMapping("/curso/{key}")
    public Optional<Curso> obterCursosPorId_(@PathVariable("key")
                                            String key) {
        return repo.findById(key);
    }
```

Para inserir novo, use POST:

- O método deve receber um `curso` no formato `JSON`
  - use `@RequestBody` para tranformar o JSON em objeto

```
@PostMapping("/curso")
public Curso novo(@RequestBody Curso curso) {
  repo.save(curso);
  return curso;
  }
```

Para atualizar, use PUT:

- O método deve receber um `curso` no formato `JSON`
  - aqui está explicitamente colocando o formato em `consumes`

```
@PutMapping(path = "/curso", consumes= {"application/json"})
public Curso clientsUpdate(@RequestBody Curso curso) {
  repo.save(curso);
  return curso;
}
```

Para remover, use DELETE (versão ruim):

- A exceção é disparada porque não é possível `serializar` null

```
@DeleteMapping("/curso/{id}")
@ResponseBody
public Curso remover(@PathVariable("id") String id) {
  Curso curso = repo.getOne(id);
  if (curso == null)
    return null; //dispara exceção

  repo.delete(curso);
  return curso; //dispara execeção
}
```

Para remover, use DELETE (melhorado):

- Use `ResponseEntity`. Ele lida com o null

```
@DeleteMapping("/curso/{id}")
@ResponseBody
public ResponseEntity<Void> remover(@PathVariable("id") String id
      Curso curso = repo.getOne(id);
      if (curso == null)
    return ResponseEntity.notFound().build();
  repo.delete(curso);
  return ResponseEntity.noContent().build();
}
```

Para testar:

> “
>
> Melhor do que ficar fazendo pelo navegador
>
> ”

- Postman
- Insomnia REST Client

Finalmente:

- RESTFull com banco de dados
- Faltam muitos detalhes, mais já tem como fazer:
    - Exercício: